МИКРОСХЕМА ИНТЕГРАЛЬНАЯ
1867ВЦ11Ф
Руководство пользователя

2016

Содержание

1 Введение

В настоящее время цифровая обработка сигналов (ЦОС) охватывает широкий спектр практических приложений. В качестве примеров можно назвать цифровую фильтрацию, кодирование речи, обработку изображений, быстрое преобразование Фурье (БПФ), телекоммуникацию и цифровую звукотехнику. Как в этих, так и в других областях, цифровая обработка сигналов имеет общие особенности:

- большой объем вычислений;
- работа в реальном времени;
- обработка оцифрованных данных;
- гибкость системы ЦОС.

Развитие техники интегральных схем позволило создать более быстрые микропроцессоры и микро-ЭВМ. В результате во многих областях, связанных с цифровой обработкой сигналов, матричные процессоры стали вытесняться разрядномодульными микропроцессорными секциями, а затем однокристальными микро-ЭВМ. Цена цифровых процессоров обработки сигналов (ПЦОС) резко снизилась, открыв дорогу для их самого широкого внедрения. Рост производительности СБИС расширил область применения ПЦОС, по традиции ограничивающуся телефонной связью, и ПЦОС стали применяться в графических системах и системах обработки изображений, а затем и в звукотехнике.

Последним достижением техники ЦОС стали однокристальные цифровые процессоры сигналов, примером которых служат процессоры серии TMS320.

В настоящее время отечественной промышленностью освоен серийный выпуск широкой номенклатуры 16-разрядных СБИС ПЦОС с фиксированной запятой: 1867ВМ1 (функциональный аналог TMS320C10), 1867ВМ2 (функциональный аналог TMS320C25), 1867ВЦ2Т (функциональный аналог TMS320C50), 1867ВЦ4Т (функциональный аналог TMS320C542), 1867ВЦ5Т (функциональный аналог TMS320F240). Однако необходимость решения сложных задач системного уровня, для которых характерно значительное увеличение динамического диапазона и высокая производительность, обусловила проектирование и освоение серийного производства 32-разрядных СБИС ПЦОС, способных оперировать с данными в формате как с фиксированной, так и с плавающей запятой; 1867ВЦ6Ф (функциональный аналог TMS320C30), 1867ВЦ3Ф (функциональный аналог TMS320C40).

Настоящий документ представляет собой руководство для следующей СБИС серии 1867 – 32-разрядного радиационно-стойкого однокристального ПЦОС 1867ВЦ11Ф с плавающей и фиксированной запятой, который является функциональным аналогом микросхемы 1867ВЦ6Ф, отличается от неё напряжением питания и повышенной стойкостью к воздействию СВВФ и имеет следующие характеристики:

Разрядность данных, бит:	
- плавающая запятая (ПЗ)	40
- фиксированная запятая (ФЗ)	32
Аппаратный умножитель, бит:	
- ПЗ	32 ×32
- ФЗ	24 × 24
Объем ПЗУ, бит	4К × 32
Объем ОЗУ, бит	2К × 32
Объем кэш ОЗУ команд, бит	64 × 32
Объем внешней адресуемой памяти, бит	16М × 32
Таймеры	2
Контроллер ПДП	1
Напряжение питания буферов ввода-вывода, В	3,3 ± 10 %
Напряжение питания ядра, В	3,3 ± 10 %
Тактовая частота	50 МГц
Время цикла	40 нс
Производительность:	
- ФЗ	25 MIPS
- ПЗ	50 MFLOPS
Количество команд	114
Количество выводов	180
Технология	КМОП КНИ
Параметры микросхемы к воздействию специальных факторов по ГОСТ РВ 20.39.414.2–98:	
- 7.$И_1$	$5У_с$ [1)]
- 7.$И_6$	$5У_с$ [2)]
- 7.$И_7$	$0,5×5У_с$
- 7.$С_1$, 7.$С_4$	$5У_с$
- 7.$К_1$	1К[3)]/2К[4)]
- 7.$К_4$	1К[3), 4)]
- 7.$К_{11}$, 7.$К_{12}$	80 МэВ×см^2/мг[2)]

[1)] По структурным повреждениям.
[2)] По катастрофическим отказам и тиристорному эффекту.
[3)] При совместном воздействии факторов с характеристиками 7.$К_1$ и 7.$К_4$.
[4)] При независимом воздействии факторов с характеристиками 7.$К_1$ и 7.$К_4$.

Критериями работоспособности микросхемы являются U_{OH}, U_{OL}, I_{OCC}, функциональный контроль.

Далее в настоящем руководстве подробно рассматриваются архитектура, регистры и память, периферийные устройства, адресация, форматы данных, система команд и другие вопросы, связанные с описанием структуры и функционирования ПЦОС.

2 Обзор архитектуры

Соединение технических и программных решений, требующихся для реализации арифметических алгоритмов, отразилось на архитектуре ПЦОС. Высокая производительность достигнута благодаря точности выполнения операций с плавающей запятой, большой памяти кристалла, высокой степени параллелизма и наличию контроллера прямого доступа к памяти ПДП (DMA).

Основные темы, представленные в разделе 2 технического описания:

- центральное процессорное устройство ЦПУ (CPU) (подраздел 2.1);
- умножитель чисел с фиксированной и плавающей запятой (Multiplier) (2.1.1);
- арифметико-логическое устройство АЛУ (ALU) для операций с фиксированной и плавающей запятой, целочисленных и логических (2.1.2);
- 32-разрядный циклический сдвигатель (Shifter);
- внутренние шины (CPU1/CPU2 и REG1/REG2);
- арифметические устройства вспомогательных регистров (ARAU0, ARAU1) (2.1.3);
- регистровый файл ЦПУ (R0-R7, AR0-AR7, IR0, IR1, Others registers) (2.1.4);
- организация памяти (подраздел 2.2);
- ОЗУ, ПЗУ и кэш (2.2.1);
- карты памяти (2.2.2);
- режимы адресации памяти (2.2.3);
- система команд (2.2.4);
- операции внутренней шины (подраздел 2.3);
- операции внешней шины (подраздел 2.4);
- периферия (подраздел 2.5);
- модули таймера (2.5.1);
- последовательные порты (2.5.2);
- прямой доступ к памяти ПДП (DMA) (подраздел 2.6);
- интеграция системы (подраздел 2.7).

2.1 Центральное процессорное устройство (ЦПУ)

ЦПУ ПЦОС имеет регистровую архитектуру. ЦПУ состоит из следующих компонентов:

- умножителя с фиксированной и плавающей запятой (Multiplier);
- АЛУ для выполнения арифметических действий: с фиксированной и плавающей запятой, целочисленных и логических операций (ALU);
- 32-разрядного циклического сдвигателя (Shifter), внутренней шины (CPU1/CPU2 и REG1/REG2);
- арифметического устройства вспомогательных регистров (ARAU0, ARAU1);
- регистрового блока ЦПУ (R0-R7, AR0-AR7, IR0, IR1, Others registers).

Структурная схема ИС 1867ВЦ11Ф приведена на рисунке 2.1.

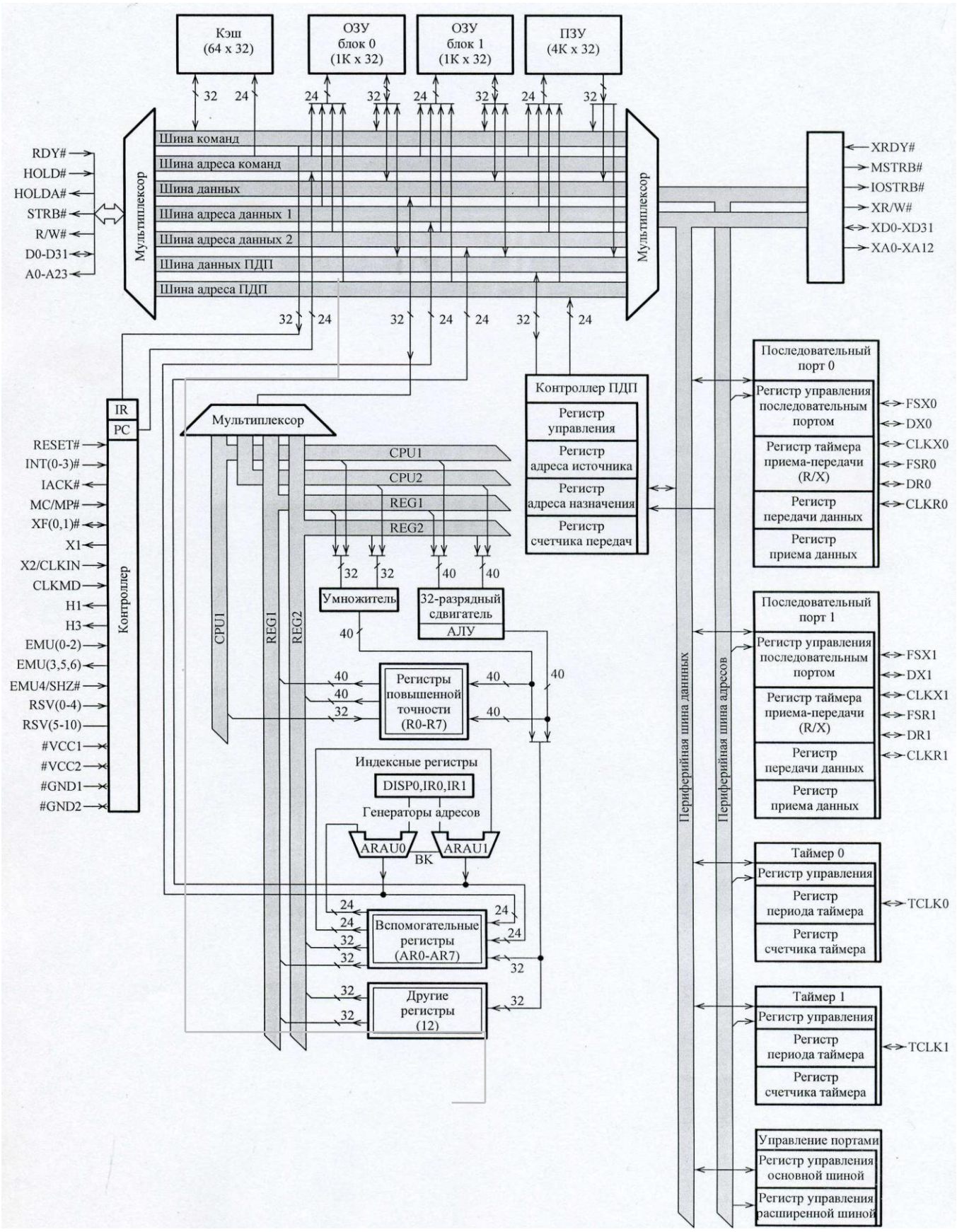


Рисунок 2.1 – Структурная схема ИС 1867ВЦ11Ф

2.1.1 Умножитель (Multiplier)

Умножитель выполняет одноцикловые умножения над 24-разрядными целыми значениями и 32-разрядными значениями с плавающей запятой. Реализация на ПЦОС арифметики с плавающей запятой допускает работу с такой же скоростью, как и для операций с фиксированной запятой: с 50 наносекундным циклом команд и высокую степень параллелизма. Для получения большей производительности операции АЛУ и умножителя могут быть выполнены за один цикл при использовании параллельных команд.

При выполнении умножения с плавающей запятой на вход подаются 32-разрядные значения с плавающей запятой, а на выходе получатся 40-разрядные значения с плавающей запятой. При выполнении целочисленного умножения на вход подаются 24-разрядные значения, а на выходе получается 32-разрядный результат. Более подробную информацию о форматах данных и операциях с плавающей запятой можно узнать в 4 разделе «Форматы данных и операции с плавающей запятой».

2.1.2 Арифметико-логическое устройство (АЛУ)

АЛУ выполняет одноцикловые операции с 24-разрядными целыми, 32-разрядными логическими и 40-разрядными значениями с плавающей запятой, включая одноцикловые целочисленные преобразования и преобразования значений с плавающей запятой. Результаты работы АЛУ всегда сохраняются в 32-разрядном целом или 40-разрядном формате с плавающей запятой. Циклический сдвигатель используется для сдвига до 32 бит влево или вправо за один цикл. Внутренние шины CPU1/CPU2 и REG1/REG2 переносят два операнда из памяти и два операнда из регистрового файла, таким образом, обеспечивая параллельное умножение и сложение/вычитание на четырех целочисленных или с плавающей запятой операндах в одном цикле.

2.1.3 Арифметические устройства вспомогательных регистров (ARAUn)

Два арифметических устройства вспомогательных регистров (ARAU0 и ARAU1) могут обрабатывать два адреса в одном цикле. ARAUn функционируют параллельно с умножителем и АЛУ. Они обеспечивают адресацию со смещениями, использование индексных регистров (IR0 и IR1), циклическую и битреверсную адресацию. Режимы адресации представлены в разделе 5 «Адресация».

2.1.4 Регистровый файл ЦПУ

ПЦОС имеет 28 регистров в многопортовом регистровом файле, который тесно связан с ЦПУ. Все эти регистры могут быть операндами умножителя и АЛУ и могут быть использованы как регистры общего назначения. Однако регистры также имеют некоторые специальные функции. Например, восемь регистров повышенной точности специально предназначены для хранения результатов повышенной точности с плавающей запятой. Восемь вспомогательных регистров поддерживают различные методы косвенной адресации и могут быть использованы как 32-разрядные целочисленные и логические регистры общего назначения. Оставшиеся регистры обеспечивают системные функции, такие как адресация, управление стеком, состояние процессора, прерывания и повторения блоков команд. В разделе 6 «Управление выполнением программы» представлена более детальная информация и примеры управления стеком и использования регистров.

Имена регистров и соответствующие функции приведены в таблице 2.1, где функции каждого регистра или группы регистров описаны кратко. В разделе 3 «Регистры ЦПУ, память и кэш» представлена более детальная информация о регистрах ЦПУ.

Таблица 2.1 – Регистры ЦПУ

Имя регистра	Функциональное назначение регистра
R0	Регистр повышенной точности 0
R1	Регистр повышенной точности 1
R2	Регистр повышенной точности 2
R3	Регистр повышенной точности 3
R4	Регистр повышенной точности 4
R5	Регистр повышенной точности 5
R6	Регистр повышенной точности 6
R7	Регистр повышенной точности 7
AR0	Вспомогательный регистр 0
AR1	Вспомогательный регистр 1
AR2	Вспомогательный регистр 2
AR3	Вспомогательный регистр 3
AR4	Вспомогательный регистр 4
AR5	Вспомогательный регистр 5
AR6	Вспомогательный регистр 6
AR7	Вспомогательный регистр 7
DP	Регистр указателя страницы данных
IR0	Индексный регистр 0
IR1	Индексный регистр 1
BK	Регистр размера блока
SP	Регистр указателя системного стека
ST	Регистр состояния
IE	Регистр разрешения прерываний ЦПУ/ПДП
IF	Регистр флагов прерываний ЦПУ
IOF	Регистр флагов ввода-вывода
RS	Регистр адреса начала повторений
RE	Регистр адреса конца повторений
RC	Регистр счетчика повторений
PC	Регистр счетчика команд

Регистры повышенной точности (R0-R7) имеют возможность хранения и выполнения операций с 32-разрядными целыми и 40-разрядными значениями с плавающей запятой. Любая команда, содержащая значения с плавающей запятой, использует разряды 39 – 0. Если операнды являются целыми со знаком или беззнаковыми, используются только разряды 31-0, разряды 39-32 остаются без изменений. Это верно для всех операций сдвига. В разделе 4 «Форматы данных и операции с плавающей запятой» указаны форматы регистров повышенной точности для значений с плавающей запятой и целочисленных значений.

32-разрядные вспомогательные регистры (AR0-AR7) могут быть доступны из ЦПУ, и модифицированы двумя арифметическими устройствами вспомогательных регистров (ARAUn). Основной функцией вспомогательного регистра является генерация 24-разрядных адресов. Они также могут быть использованы для выполнения различных функций, таких как циклические счетчики или как 32-разрядные регистры общего пользования, которые могут быть модифицированы умножителем и АЛУ. В разделе 5 «Адресация» дана подробная информация и примеры использования вспомогательных регистров для адресации. Указатель

страницы данных (DP) является 32-разрядным регистром. Восемь младших разрядов указателя страницы данных используются в режиме прямой адресации как указатели на адресуемую страницу данных. Длина страницы данных 64К слов, всего имеется 256 страниц.

32-разрядные индексные регистры (IR0 и IR1) используются арифметическими устройствами вспомогательных регистров (ARAUn) для вычисления индексированного адреса.

В разделе 6 «Управление выполнением программы» приведены примеры использования индексных регистров для адресации.

32-разрядный регистр размера блока (BK) используется ARAUn при циклической адресации для определения размера блока данных. Указатель системного стека (SP) – это 32-разрядный регистр, содержащий адрес вершины системного стека. SP всегда указывает на последний элемент, загруженный в стек. При проталкивании данных в стек происходит преддекремент, при выталкивании данных из стека – постдекремент указателя системного стека. SP манипулируется прерываниями, переходами, вызовами, возвратами и командами PUSH и POP. Для получения подробной информации об управлении системным стеком необходимо посмотреть подраздел 5.5 «Управление системным стеком и стеком пользователя».

Регистр состояния (ST) содержит глобальную информацию, относящуюся к состоянию ЦПУ. Обычно операции выставляют флаги условий регистра состояния в соответствии с результатом – нулевым, отрицательным и т. д., включая как операции загрузки и сохранения регистров, так и арифметические и логические функции. Когда регистр состояния загружен, то замена разряда на разряд выполняется над текущим содержимым операнда источника, несмотря на состояние любых разрядов в операнде источника. Следовательно, при следующей загрузке содержимое регистра состояния тождественно равно содержимому операнда источника. Это позволяет легко сохранять и восстанавливать регистр состояния. В таблице 3.2 перечислены и определены разряды регистра состояния. Регистр разрешения прерываний ЦПУ/ПДП – это 32-разрядный регистр. Битами разрешения прерывания ЦПУ являются разряды 10-0. Битами разрешения прерывания ПДП являются разряды 26-16. Единица в бите регистра разрешения прерываний разрешает соответствующие прерывания. Ноль запрещает соответствующие прерывания. В 3.1.8 даны определения разрядов регистра. Регистр флага прерываний ЦПУ (IF) является также 32-разрядным регистром (см. 3.1.9). Единица в бите регистра флага прерываний показывает, что соответствующие прерывания установлены. Ноль показывает, что соответствующие прерывания не установлены.

Регистр флагов ввода-вывода (IOF) управляет функцией специализированных внешних выводов XF0 и XF1. Эти выводы могут быть установлены в качестве входа или выхода, из них также может быть считана и записана информация. Смотри 3.1.10 для получения детальной информации.

Счетчик повторений (RC) – это 32-разрядный регистр, используемый для точного определения того, сколько раз должен быть повторен блок команд при выполнении повторений блока. При работе в режиме повторения 32-разрядный регистр начального адреса повторений (RS) содержит начальный адрес блока памяти программы, который будет повторяться, а 32-разрядный регистр адреса конца повторений (RE) – конечный адрес блока повторений.

Счетчик команд (PC) – это 32-разрядный регистр, содержащий адреса следующих выбираемых инструкций. Хотя PC не является частью регистрового файла ЦПУ, он может быть изменен командами, которые изменяют процесс выполнения программы.

2.2 Организация памяти

Общее адресное пространство ПЦОС – 16М (миллионов) 32-разрядных слов. Программа, данные и область ввода-вывода содержатся внутри этого 16М слов адресного пространства, обеспечивая, таким образом, хранение таблиц, коэффициентов, программы или данных либо в ОЗУ, либо в ПЗУ. В этом случае использование памяти может быть максимальным, и область памяти распределяется так, как это необходимо.

2.2.1 ОЗУ, ПЗУ и кэш

Блок ПЗУ – 4К × 32 бит. Каждый из блоков ОЗУ и ПЗУ поддерживает два обращения ЦПУ в одном цикле. Наличие раздельных шин команд, шин данных и шин ПДП обеспечивает параллельно: выборку программы, чтение и запись данных и работу ПДП. Например, ЦПУ может обращаться к двум значениям данных в одном блоке ОЗУ и выполнять выборку внешней программы параллельно с загрузкой ПДП другого блока ОЗУ – все внутри одного цикла. 64×32-разрядный кэш команд обеспечивает хранение часто повторяющихся фрагментов кода, что значительно уменьшает число необходимых внекристальных обращений. Эта операция допускается для кодов, сохраняемых вне кристалла в медленной, дешевой памяти. Внешние шины в этом случае также свободны для использования ПДП, выборки внешней памяти или для использования другими устройствами в системе. В разделе 3 «Регистры ЦПУ, память и кэш» приведена более подробная информация о памяти и командах кэш.

2.2.2 Карты памяти

Карта памяти зависит от того, работает ли процессор в режиме микропроцессора (MC/MP# или MCBL/MP# = 0) или в режиме микрокомпьютера (MC/MP# или MCBL/MP# = 1). Карты памяти для этих режимов очень похожи. Область от 800000h до 801FFFh картирована для расширенной шины. Если эта область доступна, то активен MSTRB#. Область адресов от 802000h до 803FFFh зарезервирована. Адреса от 804000h до 805FFFh картированы для расширенной шины. Если эта область доступна, то активен IOSTRB#. Адреса от 806000h до 807FFFh зарезервированы. Все картированные в памяти периферийные регистры размещены от 808000h до 8097FFh. В обоих режимах блок 0 ОЗУ размещен в адресах от 80900h до 809FFh и блок 1 ОЗУ расположен в адресах от 809C00h до 809FFFh. Для внешнего порта памяти допустимо размещение от 80A000h до 0FFFFFFh (STRB# активен).

В режиме микропроцессора 4К внутрикристального ПЗУ не картированы в карту памяти ПЦОС. Область от 0h до 0BFh содержит векторы прерываний, системные прерывания (векторы переходов) и зарезервированные адреса, каждый из которых доступен через порт внешней памяти (STRB# активен). Область от 0C0h до 7FFFFFh также доступна через порт внешней памяти. В режиме микрокомпьютера 4К внутрикристального ПЗУ картированы в адресах 0h - 0FFFh. 192 адреса (от 0h до BFh) внутри этого блока предназначены для векторов прерываний, векторов системных прерываний и зарезервированной области. Адреса от 1000h до 7FFFFFh доступны через порт внешней памяти (STRB# активен).

В подразделе 3.2 «Память» дана более подробная информация о картах памяти. Также приведена карта периферийной шины и адреса векторов сброса, прерываний и системных прерываний.

Будьте внимательны! Обращение к зарезервированной области памяти приведет к непредсказуемым результатам.

2.2.3 Режимы адресации памяти

ПЦОС содержит базовый набор команд общего назначения, таких как арифметические команды, которые предназначены для цифровой обработки сигналов и других интенсивных вычислительных применений. В разделе 5 «Адресация» дана более подробная информация об адресации.

ПЦОС имеет пять групп режимов адресации. Внутри группы могут быть использованы шесть типов адресации:

1) основные режимы адресации:

- регистровый. Операнд является регистром ЦПУ;
- короткая непосредственная. Операнд – 16-разрядное непосредственное значение;
- прямая. Операнд – содержимое 24-разрядного адреса;
- косвенная. Вспомогательный регистр указывает адрес операнда;

2) трехоперандовые режимы адресации:

- регистровый. Такой же, как для основных режимов адресации;
- косвенный. Такой же, как для основных режимов адресации;

3) режимы параллельной адресации:

- регистровый. Операнд – регистр повышенной точности;
- косвенный. Такой же, как для основных режимов адресации;

4) режим длинной непосредственной адресации:

- длинная непосредственная. Операнд – 24-разрядная непосредственная величина;

5) режимы адресации условных переходов:

- регистровый. Такой же, как для основных режимов адресации;
- относительно счетчика команд PC. 16-разрядов смещения прибавляется к PC.

2.2.4 Система команд

В таблице 2.2 приведен список команд ПЦОС, составленный в алфавитном порядке. Каждая строка таблицы содержит мнемонику, описание и работу команды. В разделе 10 «Команды языка ассемблера» приведен функциональный список команд и описание каждой команды.

Таблица 2.2 – Система команд

Мнемо-ника	Описание	Операция
1	2	3
ABSF	Абсолютное значение числа с плавающей запятой	|src| → Rn
ABSI	Абсолютное значение целого числа	|src| → Dreg
ADDC	Сложение целых с переносом	src + Dreg + C → Dreg
ADDC3	Сложение целых с переносом (3 операнда)	src + src2 + C → Dreg
ADDF	Сложение значений с плавающей запятой	src + Rn → Rn
ADDF3	Сложение значений с плавающей запятой (3 операнда)	src1 + src2 → Rn
ADDI	Сложение целых	src + Dreg → Dreg
ADDI3	Сложение целых (3 операнда)	src1 + src2 + → Dreg
AND	Поразрядное логическое И (AND)	Dreg AND src → Dreg

Продолжение таблицы 2.2

1	2	3
AND3	Поразрядное логическое И (AND) (3 операнда)	src1 AND src2 → Dreg
ANDN	Поразрядное логическое И (AND) с дополнением	Dreg AND src → Dreg
ANDN3	Поразрядное логическое И (AND с дополнением (3 операнда))	src1 AND src2 → Dreg
ASH	Арифметический сдвиг	Если count ≥ 0: (Сдвиг Dreg влево по счетчику) → Dreg. Иначе: (Сдвиг Dreg вправо по счетчику)→Dreg.
ASH3	Арифметический сдвиг (3 операнда)	Если count >= 0: (Сдвиг src влево по счетчику) →Dreg. Иначе: (Сдвиг src вправо по счетчику) →Dreg.
Bcond	Условный переход (стандартный)	Если cond = «истина»: Если Csrc – регистр, то Csrc → PC. Если Csrc – значение, то Csrc+PC → PC. Иначе: PC+1 → PC.
BcondD	Условный переход (с задержкой)	Если cond = «истина»: Если Csrc – регистр, то Csrc → PC. Если Csrc – значение, то Csrc+PC +3→ PC. Иначе: PC+1 → PC.
BR	Безусловный переход (стандартный)	Значение → PC
BRD	Безусловный переход (задержанный)	Значение → PC
CALL	Вызов подпрограммы	PC + 1 → TOS. Значение → PC.
CALLcond	Условный вызов подпрограммы	Если cond = «истина»: PC + 1 → TOS. Если Csrc регистр, Csrc → PC. Если Csrc значение, Csrc+PC → PC. Иначе, PC + 1 →PC.
CMPF	Сравнение значений с плавающей запятой	Устанавливает флаги в Rn – src
CMPF3	Сравнение значений с плавающей запятой (3 операнда)	Устанавливает флаги в src1 – src2
CMPI	Сравнение целых	Устанавливает флаги в Dreg – src
CMPI3	Сравнение целых	Устанавливает флаги в src1 – src2 (3 операнда)
DBcond	Декремент и условный переход (стандартный)	ARn – 1 → ARn Если cond = «истина» и ARn≥ 0: Если Csrc регистр, Csrc→PC. Если Csrc значение, Csrc+PC→ PC. Иначе, PC+1 → PC.
DBcondD	Декремент и условный переход (задержанный)	ARn – 1→ ARn Если cond = «истина» и ARn≥0: Если Csrc регистр, Csrc → PC. Если Csrc значение, Csrc+PC+3 → PC. Иначе, PC + 1 → PC.
FIX	Преобразование значений с плавающей запятой в целое	Fix (src) → Dreg
FLOAT	Преобразование целых значений в значения с плавающей запятой	Float (src) → Rn
IACK	Подтверждение прерывания	Пустое чтение src IACK защелкивается в 0, затем в 1.
IDLE	Ожидание прерывания	PC + 1 → PC. Ожидание следующих прерываний.
LDE	Загрузка значения экспоненты с плавающей запятой	src (exponent) → Rn (exponent)

Продолжение таблицы 2.2

1	2	3
LDF	Загрузка значения с плавающей запятой	src → Rn
LDFcond	Условная загрузка значения с плавающей запятой	Если cond = «истина», src → Rn. Иначе, Rn без изменений.
LDFI	Загрузка значения с плавающей запятой, блокировка. Сигнализирует об операции блокировки	src → Rn
LDI	Загрузка целого	src → Dreg
LDIcond	Условная загрузка целых	Если cond = «истина», src→ Dreg. Иначе, Dreg без изменений.
LDII	Загрузка целых, блокировка. Сигнализирует о команде блокировки	src → Dreg
LDM	Загрузка мантиссы с плавающей запятой	src (mantissa) → Rn (mantissa)
LSH	Логический сдвиг	Если count ≥ 0: (Dreg, сдвинутый влево по счетчику) → Dreg. Иначе: (Dreg, сдвинутый вправо по счетчику) → Dreg.
LSH3	Логический сдвиг (3 операнда)	Если count ≥ 0: (src, сдвинутый влево по счетчику) → Dreg. Иначе: (src, сдвинутый вправо по счетчику) → Dreg.
MPYF	Умножение значений с плавающей запятой	src × Rn → Rn
MPYF3	Умножение значений с плавающей запятой (3 операнда)	src1× src2 → Rn
MPYI	Умножение целых	src × Dreg → Dreg
MPYI3	Умножение целых (3 операнда)	src1× src2 → Dreg
NEGB	Отрицание целого с заемом	0 – src – C → Dreg
NEGF	Отрицание значения с плавающей запятой	0 – src → Rn
NEGI	Отрицание целого	0 – src → Dreg
NOP	Нет операции	Изменяется ARn, если требуется
NORM	Нормирование значений с плавающей запятой	Нормализованный (src) → Rn
NOT	Поразрядное логическое дополнение	src → Dreg
OR	Поразрядное логическое OR (ИЛИ)	Dreg OR src → Dreg
OR3	Поразрядное логическое – OR (ИЛИ) (3 операнда)	src1 OR src2 →Dreg
POP	Выталкивание целого из стека	*SP - - → Dreg
POPF	Выталкивание значений с плавающей запятой из стека	*SP - - → Rn
PUSH	Загрузка в стек целого	Sreg → * + + SP
PUSHF	Загрузка значений с плавающей запятой в стек	Rn → * + + SP
RETIcond	Условный выход из прерывания	Если cond = «истина» или опущено: *SP - - → PC 1 →ST(GIE). Иначе, продолжать.
RETScond	Условный выход из подпрограммы	Если cond = «истина» или опущено: *SP - - → PC. Иначе, продолжать.
RND	Округление значения с плавающей запятой	Round (src) → Rn
ROL	Циклический сдвиг влево	Dreg, сдвинутый влево на 1 бит → Dreg
ROLC	Циклический сдвиг влево через перенос	Dreg, сдвинутый влево на 1 бит через перенос → Dreg
ROR	Циклический сдвиг вправо	Dreg, сдвинутый вправо на 1 бит → Dreg
RORC	Циклический сдвиг вправо	Dreg, сдвинутый вправо на 1 бит через перенос → Dreg

Окончание таблицы 2.2

1	2	3
RPTB	Повторение блока команд	src → RE ; 1 → ST (RM); Следующий PC → RS.
RPTS	Повторение одной инструкции	src → RE; 1 → ST (RM); Следующий PC → RS; Следующий PC → RE.
SIGI	Сигнал блокировки сигнализирует об операции блокировки	Ожидание подтверждения блокировки. Очистка блокировки.
STF	Сохранение значения с плавающей запятой	Rn → Daddr.
STFI	Сохранение значения с плавающей запятой, блокировка	Rn → Daddr. Сигнализирует о завершении операции блокировки.
STI	Сохранение целого	Sreg → Daddr
STII	Сохранение целых, блокировка	Sreg → Daddr. Сигнализирует о завершении операции блокировки.
SUBB	Вычитание целых с заемом	Dreg – src – C → Dreg
SUBB3	Вычитание целых с заемом	src1 – src – C →Dreg
SUBB3	Вычитание целых с заемом (3 операнда)	src1 – src2 – C →Dreg
SUBC	Условное вычитание целых	Если Dreg – src ≥ 0: [(Dreg – src)<< 1] или 1 →Dreg. Иначе, Dreg << 1 →Dreg.
SUBF	Вычитание значений с плавающей запятой	Rn – src →Rn
SUBF3	Вычитание значений с плавающей запятой (3 операнда)	src1 – src2 →Rn
SUBI	Вычитание целых	Dreg – src →Dreg
SUBI3	Вычитание целых (3 операнда)	src1 – src2 → Dreg
SUBRB	Вычитание целых в обратном порядке с заемом	src – Dreg - C →Dreg
SUBRF	Вычитание значений с плавающей запятой в обратном порядке	src – Rn →Rn
SUBR	Вычитание целых в обратном порядке	src – Dreg →Dreg
SWI	Программное прерывание	Выполняет последовательно прерывания эмулятора
TRAPcond	Условное системное прерывание	Если cond = «истина» или опущено: Следующий PC → *++ SP; Вектор trap N →PC. Иначе, продолжать.
TSTB	Проверка битовых полей	Dreg AND src
TSTB3	Проверка битовых полей (3 операнда)	src1 AND src2
XOR	Поразрядное исключающее ИЛИ	Dreg XOR src →Dreg
XOR3	Поразрядное исключающее ИЛИ (3 операнда)	src1 XOR src2 →Dreg

Примечание – Применяемые условные обозначения:

- src – основные режимы адресации;
- src1 – трехоперандные режимы адресации;
- src2 – трехоперандные режимы адресации;
- Csrc – режимы адресации условных переходов;
- Sreg – адрес регистра (любой регистр);
- величина сдвига (основные режимы адресации);
- SP – указатель стека;
- GIE – регистр общего разрешения
- прерываний;
- RM – бит режима повторения;
- TOS – вершина стека;
- C – бит переноса;
- Dreg – адрес регистра (любой регистр);
- Rn – адрес регистра (R0-R7);
- Daddr – адрес назначения памяти;
- ARn – вспомогательный регистр n (AR0-AR7);
- Addr – 24-разрядный непосредственный адрес (метка);
- Cond – код условия;
- ST – регистр состояния;
- RE – регистр повторения прерываний;
- RS – регистр начала повторений;
- PC – счетчик команд.

Таблица 2.3 – Система параллельных команд

Мнемоника	Описание	Функция
1	2	3
Параллельные арифметические команды с сохранением		
ABSF ||STF	Абсолютное значение величины С плавающей запятой	|src2| → dst1 || src3 → dst2
ABSI ||STI	Абсолютное значение целого	|src2| → dst1 || src3 → dst2
ADDF3 ||STF	Сложение значений с плавающей запятой	src1 + src2 → dst1 || src3 → dst
ADDI3 ||STI	Сложение целых	src1 + src2 → dst1 || src3 → dst2
AND3 ||STI	Поразрядное логическое И (AND)	src1 AND src2 → dst1 || src → dst2
ASH3 ||STI	Арифметический сдвиг	Если count ≥ 0: src2 << count → dst1 || src3 → dst2. Иначе: src2 >> |count| → dst1 || src3 → dst2.
FIX ||STI	Преобразование значений с плавающей запятой в целое	Fix (src2) → dst1 ||src3 – dst2
FLOAT ||STF	Преобразование целого в значение С плавающей запятой	Float (src2) – dst1 || src3 → dst2
LDF ||STF	Загрузка значений с плавающей запятой	src2 → dst1 || src3 → dst2
LDI ||STI	Загрузка целых значений	src2 → dst1 || src3 → dst2
LSH3 ||STI	Логический сдвиг	Если count ≥ 0: src2 << count → dst1 || src3 → dst2. Иначе: src2 >> |count| → dst1 || src3 → dst2.
MPYF3 ||STF	Умножение значений с плавающей запятой	src1 × src2 → dst1 || src3 → dst2
MPYI3 ||STI	Умножение целых	src1 × src2 → dst1 || src3 → dst2
NEGF ||STF	Отрицание значения с плавающей запятой	0 – src2 → dst1 || src3 → dst2
NEGI ||STI	Отрицание целого	0 – src2 → dst1 || src3 → dst2
NOT ||STI	Дополнение	src1 → dst1 || src3 → dst2
OR3 ||STI	Поразрядное логическое ИЛИ (OR)	src1 OR src2 → dst1 || src3 → dst2
STF ||STF	Сохранение значений с плавающей запятой	src1 → dst1 || src3 → dst2
STI ||STI	Сохранение целых	src1 → dst1 || src3 → dst2
SUBF3 ||STF	Вычитание значений с плавающей запятой	src1 – src2 → dst1 || src3 → dst2
SUBI3 ||STI	Вычитание целых	src1 – src2 → dst1 || src3 → dst2
XOR3 ||STI	Поразрядное исключающее ИЛИ (XOR)	src1 XOR src2 → dst1 || src3 → dst2
Команды параллельной загрузки		
LDF ||LDF	Загрузка значений с плавающей запятой	src2 → dst1|| src4 → dst2

Окончание таблицы 2.3

1	2	3
LDI ||LDI	Загрузка целых	src2 → dst1 || src4 → dst2
Команды параллельного умножения и сложения/вычитания		
MPYF3 ||ADDF3	Умножение и сложение значений с плавающей запятой	op1 × op2 → op3 || op4 + op5 → op6
MPYF3 ||SUBF3	Умножение и вычитание значений с плавающей запятой	op1 × op2 → op3 || op4 – op5 → op6
MPYI3 ||ADDI3	Умножение и сложение целых	op1 × op2 → op3 || op4 + op5 → op6

Примечания

1 Применяемые условные обозначения:
- src1 – регистровая адресация (R0-R7);
- src3 – регистровая адресация (R0-R7);
- dst1 – регистровая адресация (R0-R7);
- op3 – регистровая адресация (R0 или R1);
- src2 – косвенная адресация (disp = 0, 1, R0, IR1);
- src4 – косвенная адресация (disp = 0, 1, IR0, IR1);
- dst2 – косвенная адресация (disp = 0, 1, IR0, IR1);
- op6 – регистровая адресация (R2 или R3).

2 Op1, op2, op4, op5 – два из этих операндов могут быть определены, используя регистровую адресацию, и два могут быть определены, используя косвенную.

2.3 Работа внутренней шины

Основой высокой производительности ПЦОС служит организация внутренних шин и возможность параллельной работы. Разделение шин команд (PADDR и PDATA), шин данных (DADDR1, DADDR2 и DDATA) и шин ПДП (DMAADDR и DMADATA) обеспечивает параллельную выборку программ, доступ к данным и обращение ПДП. Эти шины связывают все физические области (внутрикристальную память, внешнюю память и внутрикристальную периферию), поддерживаемые ПЦОС. Регистр команд (IR) связан с 32-разрядной шиной данных программы (PDATA). Эти шины могут выбирать однокомандное слово в каждом цикле.

24-разрядные адресные шины (DADDR1 и DADDR2) и 32-разрядная шина данных (DDATA) обеспечивают два обращения к данным памяти в каждом машинном цикле. Шина DDATA передает данные по шинам CPU1 и CPU2. Шины CPU1 и CPU2 могут передавать два операнда данных памяти в умножитель, АЛУ и в регистровый файл в каждом машинном цикле. Также в ЦПУ есть регистровые шины REG1 и REG2, которые могут передавать два значения данных из регистрового файла в умножитель и в АЛУ в каждом машинном цикле. Контроллер ПДП снабжен 24-разрядной адресной шиной (DMAADDR) и 32-разрядной шиной данных (DMADATA). Эти шины дают возможность ПДП выполнять обращение к памяти параллельно с обращением к памяти, имеющих место на шинах данных и команд.

2.4 Работа внешней шины

ПЦОС поддерживает два внешних интерфейса: основной и расширенный. Оба интерфейса состоят из 32-разрядной шины данных и набора управляющих сигналов. Основной интерфейс имеет 24-разрядную адресную шину, а расширенный – 13-разрядную адресную

шину. Обе шины могут быть использованы для адресации внешней памяти программ/данных или области ввода-вывода. Шины также имеют внешний сигнал RDY# для формирования состояния ожидания. Дополнительно состояния ожидания могут быть включены программно. В разделе 7 «Работа с внешней шиной» дана более подробная информация об операциях внешних шин.

2.4.1 Внешние прерывания

ПЦОС имеет четыре внешних прерывания (INT3# – INT0#), некоторое количество внутренних прерываний и немаскируемый внешний сигнал сброса RESET#. Они могут быть использованы для прерывания ЦПУ или ПДП. Когда ЦПУ обрабатывает прерывание, вывод IACK# может быть использован для сигнализации подтверждения внешнего прерывания. Подраздел 6.5 «Операция сброса» посвящен обработке RESET# и прерываний.

2.4.2 Сигнализация об инструкциях блокировки

Два внешних флага ввода-вывода XF0 и XF1 могут быть использованы как контакты ввода или вывода в управляющем программном обеспечении. Эти выводы также используются операциями блокировки ПЦОС. Группа команд блокировки обеспечивает мультипроцессорную связь, см. подраздел 6.4 «Операции блокировки» – примеры использования команд блокировки.

2.5 Периферия

Все периферийные устройства ПЦОС управляются регистрами, картированными в памяти, через специальную периферийную шину. Эта шина соединена с 32-разрядной шиной данных и 24-разрядной адресной шиной. Периферийная шина обеспечивает прямую связь с периферийными устройствами. Периферийные устройства ПЦОС включают два таймера и два последовательных порта. В разделе 8 «Периферийные устройства» приводится более подробная информация о периферийных устройствах.

2.5.1 Модули таймера

Два модуля таймера являются 32-разрядными универсальными счетчиками времени/числа событий с двумя режимами сигнализации и внешней или внутренней синхронизацией. Каждый таймер имеет контакт ввода-вывода, который может быть использован как вход синхронизации таймера или как выходной сигнал, управляемый таймером. Контакт может также быть использован как контакт ввода-вывода общего назначения.

2.5.2 Последовательные порты

Два последовательных порта полностью независимы. Они одинаковы и имеют комплементарный набор регистров управления, контролирующих каждый из портов. Каждый последовательный порт может быть установлен для пересылок 8-, 16-, 24- или 32-разрядными словами данных. Синхронизация для каждого последовательного порта может быть внутренней или внешней. Обеспечивается внутренняя генерация синхросигнала с делением частоты. Выводы последовательного порта можно представить как контакты ввода-вывода общего назначения. Последовательные порты могут также быть сконфигурированы как таймеры. Специальный режим квитирования передачи позволяет связать несколько ПЦОС через последовательные порты с гарантированной синхронизацией.

2.6 Прямой доступ к памяти (ПДП)

Контроллер прямого доступа к памяти (ПДП) на кристалле может считывать или записывать любую область памяти без взаимодействия с работой ЦПУ. Поэтому ПЦОС может работать с медленным внешним интерфейсом памяти и периферией без уменьшения производительности ЦПУ. Контроллер ПДП содержит свой собственный генератор адреса, регистры исходного адреса и назначенного адреса и счетчик переданных слов. Соответствующие шины адреса и данных ПДП позволяют минимизировать конфликты между ЦПУ и контроллером ПДП. Работа ПДП состоит из пересылок блока или одного слова в память или из памяти. В подразделе 8.3 «Контроллер ПДП» приведена более подробная информация о работе контроллера ПДП.

2.7 Интеграция системы

ПЦОС является высокопроизводительной ЦОС системой благодаря интеграции современных решений, мощному ЦПУ, большому объему памяти и двум эффективным интерфейсам шины для поддержания быстродействия. Периферийные устройства, такие как контроллер ПДП, два последовательных порта и два таймера находятся на кристалле. Однокристальная реализация ПЦОС позволяет значительно уменьшить размеры и цену разрабатываемых систем ЦОС.

3 Регистры ЦПУ, память и кэш

Регистровый файл ЦПУ содержит 28 регистров, которые могут использоваться умножителем и АЛУ. В регистровый файл включены вспомогательные регистры, регистры повышенной точности и индексные регистры. Регистры в регистровом файле ЦПУ поддерживают адресацию, операции с плавающей запятой, управление стеком, состояние процессора, повторение блока и прерывания.

ПЦОС имеет общее адресное пространство памяти в 16M 32-разрядных слов, включая области программ, данных и ввода-вывода.

Два блока ОЗУ по 1K × 32 разрядов каждый и блок ПЗУ размером 4K × 32 разрядов обеспечивают два обращения ЦПУ в одном цикле.

Карты памяти в режиме микрокомпьютера и микропроцессора аналогичны, за исключением того, что ПЗУ на кристалле не используется в режиме микропроцессора.

64 × 32-разрядный кэш команд хранит часто повторяющиеся фрагменты программы. Это сильно уменьшает количество необходимых обращений к памяти вне кристалла и позволяет хранить программу вне кристалла в медленной недорогой памяти. Три разряда регистра состояния ЦПУ обеспечивают управление очисткой, разрешением или «замораживанием» кэш.

В данном разделе подробно описаны каждый из регистров ЦПУ, карты памяти и команды кэш. Основные темы в данном разделе следующие:

1) регистровый файл ЦПУ (подраздел 3.1):

- регистры повышенной точности (R0-R7):
- вспомогательные регистры (AR0-AR7);

- индексные регистры (IR0, IR1);
- регистр размера блока (BK);
- указатель страницы данных (DP);
- указатель системного стека (SP);
- регистр состояния (ST);
- регистр разрешения прерываний ЦПУ/ПДП (IE);
- регистр флага прерываний ЦПУ (IF);
- регистр флага ввода-вывода (IOF);
- счетчик повторений (RC) и регистры повторений блока (RS, RE);
- счетчик команд (PC);

2) память (подраздел 3.2):
- карта памяти;
- карта периферийной шины;
- карта сброса/прерывания/системного прерывания;

3) кэш команд (подраздел 3.3):
- архитектура кэш;
- алгоритм кэш;
- управляющие разряды кэш.

3.1 Набор регистров ПЦОС

ПЦОС имеет 28 регистров в мультипортовом регистровом файле, который тесно связан с ЦПУ. Счетчик команд PC не включен в эти 28 регистров. Все эти регистры могут обрабатываться умножителем и АЛУ и могут использоваться как регистры общего назначения. Однако регистры имеют специальные функции, для которых они предназначены. Например, восемь регистров повышенной точности наиболее подходят для хранения результатов повышенной точности с плавающей запятой.

Восемь вспомогательных регистров обеспечивают различные режимы косвенной адресации и могут быть использованы как 32-разрядные целочисленные и логические и регистры общего назначения. Оставшиеся регистры обеспечивают такие системные функции, как адресация, управление стеком, состояние процессора, прерывания и повторение блока.

В разделе 5 «Адресация» дана более подробная информация и примеры использования регистров ЦПУ для адресации.

Сводный список набора регистров ПЦОС приведен в таблице 3.1

Таблица 3.1 – Набор регистров ПЦОС

Мнемоника	Название регистра
R0	Регистр повышенной точности 0
R1	Регистр повышенной точности 1
R2	Регистр повышенной точности 2
R3	Регистр повышенной точности 3
R4	Регистр повышенной точности 4
R5	Регистр повышенной точности 5
R6	Регистр повышенной точности 6
R7	Регистр повышенной точности 7
AR0	Вспомогательный регистр 0
AR1	Вспомогательный регистр 1
AR2	Вспомогательный регистр 2
AR3	Вспомогательный регистр 3
AR4	Вспомогательный регистр 4
AR5	Вспомогательный регистр 5
AR6	Вспомогательный регистр 6
AR7	Вспомогательный регистр 7
DP	Указатель страницы данных
IR0	Индексный регистр 0
IR1	Индексный регистр 1
BK	Регистр размера блока
SP	Указатель системного стека
ST	Регистр состояния
IE	Регистр разрешения прерываний ЦПУ/ПДП
IF	Флаги прерываний ЦПУ
IOF	Флаги ввода-вывода
RS	Регистр адреса начала повторений
RE	Регистр адреса конца повторений
RC	Счетчик повторений
PC	Программный счетчик

3.1.1 Регистры повышенной точности (R0-R7)

Восемь регистров повышенной точности (R0-R7) могут хранить и оперировать с 32-разрядными целыми и 40-разрядными значениями с плавающей запятой. Эти регистры состоят из двух различных областей. Разряды 39-32 регистров повышенной точности предназначены для хранения экспоненты (e) значения с плавающей запятой. Разряды 31-0 хранят мантиссу значения с плавающей запятой. Разряд 31 – это знаковый разряд (s), разряды 30-0 – это дробная часть (f). Любая команда, которая содержит операнд с плавающей запятой, использует разряды 0-39.

Для работы с целыми значениями разряды 31-0 регистров повышенной точности содержат целые значения (со знаком или без знака). Любая команда, которая содержит операнды с целыми значениями со знаком или без знака использует только разряды 31-0. Разряды 39-32 остаются без изменений. Это верно для любых операций сдвига.

3.1.2 Вспомогательные регистры (AR0-AR7)

Восемь 32-разрядных вспомогательных регистров (AR0-AR7) могут быть доступны для ЦПУ и могут быть модифицированы арифметическими устройствами вспомогательных

регистров ARAUn. Основная функция вспомогательных регистров – это генерация 24-разрядных адресов. Однако они также могут быть использованы для выполнения различных функций, таких как циклический счетчик для косвенной адресации или 32-разрядные регистры общего назначения, которые могут быть изменены умножителем и АЛУ.

В разделе 5 «Адресация» дана более подробная информация и примеры использования вспомогательных регистров для адресации.

3.1.3 Указатель страницы данных (DP)

Указатель страницы данных (DP) – это 32-разрядный регистр. Восемь младших значащих разрядов указателя страницы данных используются в режиме прямой адресации как указатель на страницу адресуемых данных. Страница данных имеет длину 64К слов, общее количество страниц – 256. Разряды 31-8 зарезервированы и будут всегда равны 0.

3.1.4 Индексные регистры (IR0, IR1)

32-разрядные индексные регистры (IR0, IR1) используются арифметическими устройствами вспомогательных регистров ARAUn для индексирования адресов. В разделе 5 «Адресация» дана подробная информация и примеры использования индексных регистров при адресации.

3.1.5 Регистр размера блока (BK)

32-разрядный регистр размера блока (BK) используется ARAUn при циклической адресации для точного определения размера блока данных (см. подраздел 5.3 «Циклическая адресация»).

3.1.6 Указатель системного стека (SP)

Указатель системного стека (SP) – это 32-разрядный регистр, содержащий адрес вершины системного стека. SP всегда указывает на следующий элемент, выталкиваемый из стека. SP манипулируется прерываниями, системными прерываниями, вызовами, возвратами и командами PUSH, PUSHF, POP, POPF. При выталкивании из стека и проталкивании данных в стек выполняется прединкремент и постдекремент указателя стека на всех 32 разрядах. Однако только 24 младших значащих разряда используются как адрес. В подразделе 5.5 «Управление системным стеком и стеком пользователя» дана информация об управлении системным стеком.

3.1.7 Регистр состояния (ST)

Регистр состояния (ST) содержит полную информацию, относящуюся к состоянию ЦПУ. Обычно операции устанавливают флаги в регистре состояния в соответствии с наличием нулевого, отрицательного результата и т. д. Сюда включаются операции загрузки и хранения регистра, а также арифметические и логические операции. Однако если регистр состояния загружен, содержимое операнда источника заменяет текущее содержимое разряда в разряд, независимо от состояния любого разряда в операнде источника. Поэтому следующая загрузка содержит регистр состояния, идентичный содержимому операнда источника. Это позволяет достаточно просто сохранять и восстанавливать регистр состояния. При системном сбросе в этот регистр записывается 0.

В таблице 3.2 приведены список разрядов регистра состояния, их имена и функции.

Таблица 3.2 – Описание разрядов регистра состояния

Разряд	Наименование	Функция
0*	C	Флаг переноса.
1*	V	Флаг переполнения.
2*	Z	Флаг нулевого значения.
3*	N	Флаг отрицательного значения.
4*	UF	Флаг потери значимости разрядов числа с плавающей запятой.
5*	LV	Фиксируемый флаг переполнения.
6*	LUF	Фиксируемый флаг потери значимости разрядов числа с плавающей запятой.
7	OVM	Флаг режима переполнения. Этот флаг действует только при целочисленных операциях. Если OVM=0, то режим переполнения отключен; целые результаты с переполнением не обрабатываются специальным методом. Если OVM=1: a) целые положительные результаты с переполнением устанавливаются в значение наибольшего положительного 32-разрядного числа в дополнительном коде (7FFFFFFFh); b) целые отрицательные результаты устанавливаются в наименьшее отрицательное 32-разрядное число в дополнительном коде (80000000h). Примечание – Функции регистров V и LV независимы от установки OVM.
8	RM	Флаг режима повторения. Если RM=1, то PC модифицируется или при повторении блока команд или в режиме одиночного повторения.
9	резервный	Считывается как 0.
10	CF	«Замораживание» кэш. Если CF=1, то кэш заморожен. Если кэш разрешен (CE=1), то выбор из кэш разрешен, но состояние кэш не изменяется. Эта функция может быть использована для сохранения часто используемых кодов резидентно в кэш. При сбросе в этот разряд заносится 0. Очистка кэш (CC=1) разрешается, если CF=0.
11	CE	Разрешение кэш. CE=1 разрешает кэш, позволяя использовать кэш в соответствии с алгоритмом кэш LRU (наименее недавно использованный). CE=0 запрещает кэш, кэш не изменяется. Не происходит выборка из кэш. Эта функция используется для отладки системы. При сбросе в этот разряд заносится 0. Очистка кэш (CC=1) разрешается, если CE=0.
12	CC	Очистка кэш. CC=1 запрещает все содержимое кэш. Этот разряд всегда очищается после того, как он записан, и всегда читается как 0. При сбросе в этот разряд – записывается 0.
13	GIE	Глобальное разрешение прерываний. Если GIE=1, то ЦПУ отвечает на разрешенное прерывание. Если GIE=0, то ЦПУ не отвечает на разрешенное прерывание.
14, 15	резервный	Считывается как 0.
16-31	резервный	Значение не определено.

* Семь флагов условий (разряды ST6-ST0) описываются более подробно в подразделе 10.2 «Коды условий и флаги».

3.1.8 Регистр разрешения прерываний ЦПУ/ПДП (IE)

Регистр разрешения прерываний ЦПУ/ПДП (IE) – это 32-разрядный регистр. Разряды разрешения прерываний ЦПУ: 10-0. Разряды разрешения прерываний ПДП: 26-16. Единица в разряде регистра разрешения прерываний ЦПУ/ПДП разрешает соответствующее прерывание. Ноль запрещает соответствующее прерывание. При сбросе в этот регистр записывается 0. В таблице 3.3 определены разряды регистра, имена разрядов регистра и функции разрядов.

Таблица 3.3 – Описание разрядов регистра IE

Разряд	Наименование	Функция
0	EINT0	Разрешение внешнего прерывания 0 (ЦПУ)
1	EINT1	Разрешение внешнего прерывания 1 (ЦПУ)
2	EINT2	Разрешение внешнего прерывания 2 (ЦПУ)
3	EINT3	Разрешение внешнего прерывания 3 (ЦПУ)
4	EXINT0	Разрешение прерывания передачи последовательного порта 0 (ЦПУ)
5	EXINT0	Разрешение прерывания приема последовательного порта 0 (ЦПУ)
6	EXINT0	Разрешение прерывания передачи последовательного порта 1 (ЦПУ)
7	EXINT0	Разрешение прерывания приема последовательного порта 1 (ЦПУ)
8	ETINT0	Разрешение прерывания таймера 0 (ЦПУ)
9	ETINT1	Разрешение прерывания таймера 1 (ЦПУ)
10	EDINT	Разрешение прерывания контроллера DMA (ЦПУ)
11-15	резерв	Значения не определены
16	EINT0	Разрешение внешнего прерывания 0 (ПДП)
17	EINT1	Разрешение внешнего прерывания 1 (ПДП)
18	EINT2	Разрешение внешнего прерывания 2 (ПДП)
19	EINT3	Разрешение внешнего прерывания 3 (ПДП)
20	EXINT0	Разрешение прерывания передачи последовательного порта 0 (ПДП)
21	ERINT0	Разрешение прерывания приема последовательного порта 0 (ПДП)
22	EXINT1	Разрешение прерывания передачи последовательного порта 1 (ПДП)
23	ERINT1	Разрешение прерывания при приеме последовательным портом 1 (ПДП)
24	ETINT0	Разрешение прерывания таймера 0 (ПДП)
25	ETINT1	Разрешение прерывания таймера 1 (ПДП)
26	EDINT	Разрешение прерывания контроллера ПДП (ПДП)
27-31	резерв	Неопределенные значения

3.1.9 Регистр флага прерываний ЦПУ (IF)

Единица в разряде регистра флага прерываний ЦПУ указывает на то, что соответствующее прерывание установлено. Единица в разряде регистра может быть уставлена программно, вызывая прерывание. 0 показывает, что соответствующее прерывание не установлено. Если 0 записан в разряд регистра флага прерываний, то соответствующее прерывание очищено. При сбросе в этот регистр записывается 0. В таблице 3.4 приведены список разрядов регистра флага прерываний ЦПУ, их имена и функции.

Таблица 3.4 – Описание разрядов регистра IF

Разряд	Имя	Функция
0	INT0	Флаг 0 внешнего прерывания
1	INT1	Флаг 1 внешнего прерывания
2	INT2	Флаг 2 внешнего прерывания
3	INT3	Флаг 3 внешнего прерывания
4	XINT0	Флаг прерывания при передаче последовательного порта 0
5	RINT0	Флаг прерывания при приеме последовательного порта 0
6	XINT1	Флаг прерывания при передаче последовательного порта 1
7	RINT1	Флаг прерывания при приеме последовательного порта 1
8	TINT0	Флаг прерывания таймера 0
9	TINT1	Флаг прерывания таймера 1
10	DINT0	Флаг прерывания канала ПДП
11-31	резерв	Значения не определены

3.1.10 Регистр флага ввода-вывода (IOF)

Регистр флага ввода-вывода (IOF) управляет функцией специальных внешних выводов XF0 и XF1. Эти выводы могут быть определены в качестве входа или выхода. По ним также могут быть считаны или переданы данные. При сбросе в регистр IOF заносится 0. Список разрядов, их имена и функции регистра приведены в таблице 3.5.

Таблица 3.5 – Описание разрядов регистра IOF

Разряд	Наименование	Функция
0	Зарезервированы	Читается как 0.
1	I/OXF0	Если I/OXF0=0, XF0 конфигурирован как контакт входа общего назначения. Если I/OXF0=1, XF0 конфигурирован как контакт выхода общего назначения.
2	OUTXF0	Вывод данных на XF0.
3	INXF0	Ввод данных на XF0. Запись не влияет.
4	Зарезервирован	Считывается как 0.
5	I/OXF1	Если I/OXF1=0, XF1 конфигурирован как контакт входа общего назначения. Если I/OXF1=1, XF1 конфигурирован как контакт выхода общего назначения.
6	OUTXF1	Вывод данных на XF1.
7	INXF1	Ввод данных на XF1. Запись не влияет.
8-31	Зарезервированы	Считываются как 0.

3.1.11 Счетчик повторений (RC) и регистры повторений блока (RS, RE)

Счетчик повторений (RC) – это 32-разрядный регистр, используемый для точного определения количества повторений блока кода при выполнении.

Регистр начального адреса повторений (RS) – это 32-разрядный регистр, содержащий начальный адрес повторяющегося блока памяти программы, при работе в режиме повторений.

32-разрядный регистр повторений и адреса (RE) содержит адрес конца повторяющегося блока памяти программы при работе в режиме повторений.

3.1.12 Счетчик команд (PC)

Счетчик команд (PC) – это 32-разрядный регистр, содержащий адрес следующей выполняемой команды. Счетчик команд не является частью регистрового файла, он является регистром, который может изменяться командами в процессе выполнения программы.

3.1.13 Резервные разряды и совместимость

Чтобы добиться совместимости с будущими схемами семейства микропроцессоров ПЦОС, резервные разряды, читающиеся как 0, должны быть записаны как 0. Резервные разряды, имеющие неопределенные значения, не должны изменять текущее значение. В остальных случаях пользователь должен поддерживать резервные разряды точно определенными.

3.2 Память

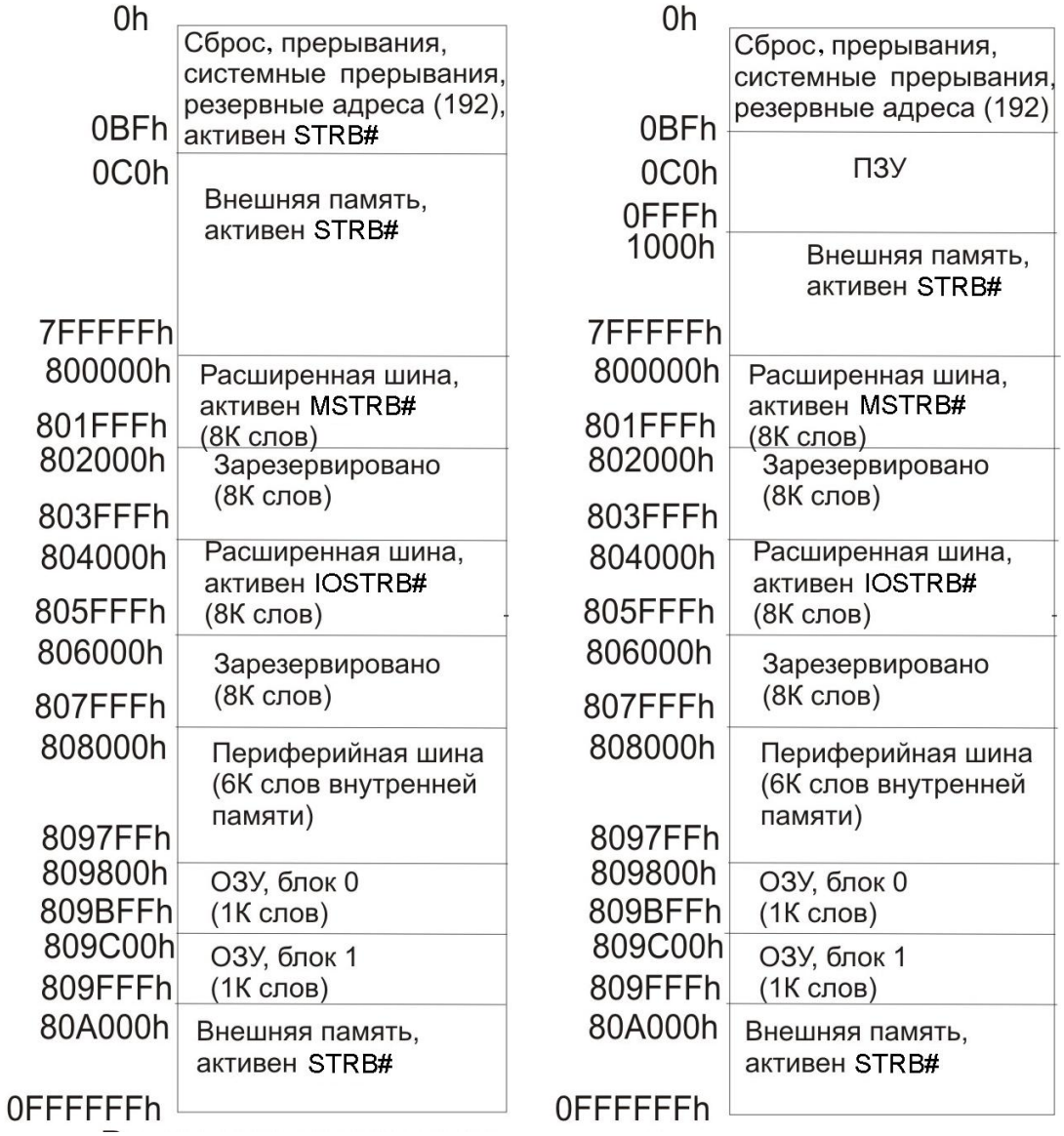
Общее адресное пространство памяти ПЦОС составляет 16M 32-разрядных слов, содержит область памяти программ данных и область ввода-вывода, что позволяет таблицам, коэффициентам, программным кодам или данным храниться либо в ОЗУ, либо в ПЗУ. В этом случае использование памяти может быть максимизировано и область памяти распределяется, как требуется.

ОЗУ имеет блоки 0 и 1, каждый из которых размером 1К×32 бит. Блок ПЗУ – 4К×32 бит. Каждый из блоков ОЗУ или ПЗУ может обеспечивать два обращения ЦПУ за один цикл. Отдельные шины команд и данных и ПДП обеспечивают параллельную выборку программ, чтение/запись данных и операции ПДП. Это подробно описано в разделе 9 «Работа конвейера».

3.2.1 Карта памяти ПЦОС

Карта памяти зависит от того, в каком режиме работает процессор: в режиме микропроцессора (MC/MP# или MCBL/MP# = 0) или в режиме микрокомпьютера (MC/MP# или MCBL/MP# = 1). Карта памяти для этих режимов представлены на рисунке 3.1.

Резервные пространства области памяти ПЦОС и резервные адреса периферийных шин не должны читаться и записываться пользователем. В этом случае может произойти останов работы ПЦОС и потребуется перезапуск системы.



Режим микропроцессора (MC/MP# или MCBL/MP# = 0)

Режим микрокомпьютера (MC/MP# или MCBL/MP# = 1)

Рисунок 3.1 – Карта памяти

3.2.2 Карта векторов сброса, прерывания, системного прерывания

Адреса для векторов возврата, прерывания и системного прерывания распределены от 0h до 3Fh. Векторы, хранящиеся в этих областях, являются начальными адресами программ сброса, прерывания и системного прерывания. Например, при сбросе содержимое области памяти 0h (вектор прерывания) загружается в PC, и выполнение начинается с этого адреса.

Системные прерывания 28-31 зарезервированы и не могут быть использованы!

3.2.3 Карта периферийной шины

Распределенные в памяти периферийные регистры расположены начиная с адреса 808000h. Каждое периферийное устройство занимает область в 16 слов карты памяти. Области от 808010h до 80801Fh и от 808070h до 8097FFh зарезервированы.

3.3 Кэш команд

64 × 32-разрядный кэш команд позволяет максимально ускорить выполнение программы при минимальных системных затратах, путем сохранения в кэш фрагментов программы, которые могут быть выбраны, когда необходим повторный доступ к этим фрагментам. Это значительно уменьшает число необходимых обращений к памяти вне кристалла и позволяет хранить программу вне кристалла в медленной недорогой памяти. Внешние шины в этом случае свободны от выборки программы, что может быть использовано ПДП и другими элементами системы.

Кэш может работать в полностью автоматическом режиме без вмешательства пользователя. Используется алгоритм изменения кэш LRU (Least Recently Used) – откачка наименее часто используемых фрагментов программы (см. 3.3.2).

3.3.1 Архитектура кэш

Кэш команд содержит 64 × 32-разрядных слов ОЗУ и разделен на два сегмента по 32 слова каждый. 19-разрядный регистр сегмента начального адреса (SSA) соединен с каждым сегментом. Каждому слову в кэш соответствует одноразрядный P (Present) флаг.

Когда ЦПУ запрашивает командное слово из внешней памяти, то проверяется, не содержится ли слово в кэш команд. 19 старших разрядов адреса команды используются для выбора сегмента, 5 младших разрядов определяют адрес слова команды внутри выбранного сегмента. 19 старших значащих разрядов адреса команды сравниваются с двумя регистрами начального адреса сегмента (SSA). Если соответствие найдено, то проверяется флаг P. Флаг P показывает, присутствует ли уже или нет слово внутри соответствующего сегмента памяти кэш. Если соответствие не найдено, то один из сегментов должен быть заменен новыми данными. Заменяемый сегмент в этом случае определяется алгоритмом LRU. Для этих целей существует стек LRU.

Стек LRU определяет, который из двух сегментов квалифицируется как наиболее давно использованный после каждого доступа к кэш; таким образом, стек содержит 0, 1 или 1, 0. Каждый раз при доступе к сегменту, номер сегмента удаляется из стека LRU и проталкивается в вершину стека LRU. Таким образом, число в вершине стека представляет собой номер последнего использованного сегмента, а число на дне стека – номер наиболее давно используемого сегмента.

При сбросе системы стек LRU инициализируется с установкой 0 в вершине, с 1 на дне стека, а все P флаги в кэш команд очищены.

Когда необходима замена, то для нее выбирается наиболее давно используемый сегмент. 32 флага P для заменяемого сегмента устанавливаются в 0, и в сегментный регистр SSA записываются 19 старших разрядов адреса команды.

3.3.2 Алгоритм кэш

Когда ПЦОС запрашивает командное слово из внешней памяти, то возможны два случая: кэш-попадание (команда найдена в кэш) или кэш-отсутствие (команда в кэш не найдена):

- первый случай – кэш-попадание. Требуемая команда содержится в кэш, и производятся следующие действия:

- командное слово считывается из кэш;

- номер сегмента, внутри которого содержится слово, удаляется из стека LRU и проталкивается на вершину стека LRU, т. о. перемещая номер другого сегмента на дно стека;

- второй случай – кэш-отсутствие. Команда не содержится в кэш.

Типы кэш-отсутствия:

- слово не найдено. Регистр адреса сегмента сравнивает адрес команды, но подходящий флаг не установлен. Следующие действия производятся параллельно:

1) командное слово считывается из памяти и копируется в кэш;

2) номер сегмента, внутри которого содержится слово, удаляется из стека LRU и проталкивается в вершину стека LRU, т. о. перемещая номер другого сегмента на дно стека;

3) устанавливается соответствующий флаг P;

- сегмент не найден. Никакой адрес сегмента не соответствует адресу команды. Следующие действия происходят параллельно:

1) наиболее редко используемый сегмент выбирается для замены. Флаги P для всех 32 слов очищены;

2) регистр SSA для выбранного сегмента загружается 19 старшими разрядами адреса запрошенного командного слова;

3) командное слово выбирается и копируется в кэш. Оно направляется в соответствующую позицию наиболее редко используемого сегмента. Флаг P для этого слова устанавливается в 1;

4) номер сегмента, содержащего слово, удаляется из стека LRU и проталкивается в вершину стека LRU, т. о. перемещая номер другого сегмента на дно стека.

Из кэш команд могут быть выбраны только команды. Все операции чтения/записи данных в память происходят без участия кэш. Выборка программы из внутренней памяти не изменяет кэш и не будет генерировать состояние попадания или отсутствия кэш.

Кэш команд – это блок памяти одиночного доступа. Пустая программная выборка (т. е. с последующим ветвлением) обрабатывается кэш как действительная выборка программы и может генерировать состояние попадания или отсутствия кэш.

Особое внимание необходимо при использовании самомодифицирующегося кода. Если команда находится в кэш и соответствующая область основной памяти изменена, то копия команды в кэш не изменяется.

Наиболее эффективное использование кэш может быть достигнуто с помощью выравнивания кода программы по границе адреса в 32 слова. Это может быть сделано, если использовать директиву ALIGN при написании программ на языке ассемблера.

3.3.3 Управляющие разряды кэш

Три управляющих разряда кэш размещены в регистре состояния ЦПУ: разряд очистки кэш (CC), разряд разрешения кэш (CE) и разряд «замораживания» кэш (CF).

Разряд очистки кэш (CC). При записи 1 в разряд очистки CC аннулируются все вводы в кэш. Все флаги P в кэш очищаются. Разряд CC всегда очищается после очистки кэш. Поэтому он всегда читается как 0. При сбросе кэш очищается и в этот разряд заносится 0.

Разряд разрешения кэш (CE). При записи 1 в этот разряд кэш разблокируется. Когда кэш разблокирован, то он используется в соответствии с вышеописанным алгоритмом кэш. При записи 0 в разряд разрешения кэш – кэш блокируется, дополнения или модификация не могут быть выполнены. Не выполняются дополнения регистра SSA, не изменяются флаги P (пока CC не равно 1) и не изменяется стек LRU. При записи 1 в CC, когда кэш блокирован, кэш будет очищен и, таким образом, очищены флаги P. Когда кэш блокирован, выборка из кэш производиться не будет. При сбросе в этот разряд заносится 0.

Разряд «замораживания» кэш (CF). Когда CF=1 – кэш заморожен. Если к тому же кэш разблокирован, то выбор из кэш разрешен, но состояние кэш не изменяется. Регистр SSA не обновляется, P флаги не изменяются (пока CC=0) и стек LRU не изменится. Эта функция может быть использована для хранения часто используемых кодов резидентно в кэш. Если кэш заморожен, то при занесении 1 в CC происходит очистка кэш и, следовательно, очистка флагов P. В результате сброса в этот разряд заносится 0.

Таблица 3.6 – Результат установки различных комбинаций разрядов CE и CF

CE	CF	Эффект
0	0	кэш не разрешен
0	1	кэш не разрешен
1	0	кэш разрешен и не заморожен
1	1	кэш разрешен и заморожен

4 Форматы данных и операции с плавающей запятой

Организация данных в архитектуре ПЦОС поддерживает три основных типа данных: целые, целые без знака и числа в формате с плавающей запятой.

Термины целые и целые без знака будут считаться эквивалентными. ПЦОС поддерживает короткие и с одинарной точностью форматы для целых со знаком и без знака. Он также поддерживает короткие, с одинарной точностью и с повышенной точностью форматы для значений с плавающей запятой.

Операции с плавающей запятой обеспечивают удобные и безошибочные вычисления.

Реализация арифметики с плавающей запятой на ПЦОС позволяет производить операции с плавающей запятой со скоростью целочисленных, в то же время, предотвращая проблемы с переполнением, выравниванием операндов и другие общие проблемы целочисленных операций.

В данном разделе подробно обсуждаются форматы данных и операции с плавающей запятой, поддерживаемые ПЦОС, рассматриваются следующие основные темы:

- целочисленные форматы (подраздел 4.1);
- целочисленные форматы целых без знака (подраздел 4.2);
- форматы значений с плавающей запятой (подраздел 4.3);
- умножение значений с плавающей запятой (подраздел 4.4);
- сложение и вычитание значений с плавающей запятой (подраздел 4.5);
- нормализация (подраздел 4.6);
- округление (подраздел 4.7);
- преобразование значений с плавающей запятой в целые (подраздел 4.8);
- преобразование целых значений в значения с плавающей запятой (подраздел 4.9).

4.1 Форматы целых

ПЦОС поддерживает два целочисленных формата: 16-разрядный короткий целый формат и 32-разрядный целый формат с одинарной точностью. Когда регистры повышенной точности применяются как целочисленные операнды, используются только разряды 31-0; разряды 39-32 остаются без изменения и не используются.

4.1.1 Короткий целочисленный формат

Короткий целый формат – это 16-разрядный в форме дополнения целый формат, используемый для непосредственных целых операндов. Для тех команд, которые содержат целые операнды, происходит расширение этого формата за счет знака до 32 разрядов (смотри рисунок 4.1). Диапазон целого числа Ni, представленный в коротком целом формате, равен:

$$-2^{15} \le Ni \le 2^{15} - 1 \qquad (4.1)$$

На рисунке 4.1 – s – знаковый разряд.

Рисунок 4.1 – Короткий целый формат и расширение знака короткого целого

4.1.2 Целый формат с одинарной точностью

В целом формате с одинарной точностью целые числа представлены в форме дополнения. Диапазон целого числа *Np*, представленного в целом формате с одинарной точностью:

$$-2^{31} \le Np \le 2^{31} - 1 \qquad (4.2)$$

На рисунке 4.2 приведен целый формат с одинарной точностью.

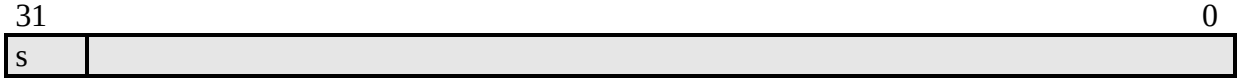


Рисунок 4.2 – Целый формат с одинарной точностью

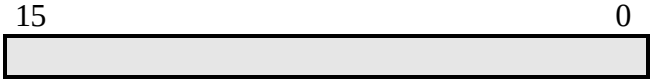
4.2 Форматы целых без знака

ПЦОС поддерживает два формата целых без знака: 16-разрядный короткий формат и 32-разрядный формат с одинарной точностью. В регистрах повышенной точности целые операнды без знака используют только разряды 31-0; разряды 39-32 остаются без изменений.

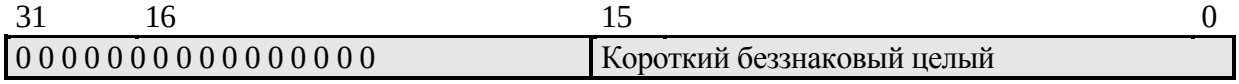
4.2.1 Короткий целочисленный формат без знака

На рисунке 4.3 приведен 16-разрядный короткий формат без знака, используемый для непосредственных целых операндов без знака. В тех командах, которые содержат целые без знака операнды, этот формат заполнен нулями до 32 разряда. Диапазон целого числа *Ni*:

$$0 \le Ni \le 2^{16} \qquad (4.3)$$



(a) Короткий целый формат без знака



(b) Заполнение нулями короткого целого формата без знака

Рисунок 4.3 – Короткий целочисленный формат без знака (a)
и с заполнением нулями (b)

4.2.2 Целочисленный формат с одинарной точностью без знака

В целочисленном формате с одинарной точностью без знака число представлено как 32-разрядное значение, как показано на рисунке 4.4. Диапазон целого числа *Np*:

$$0 \le \mathrm{Np} \le 2^{32} \qquad (4.4)$$

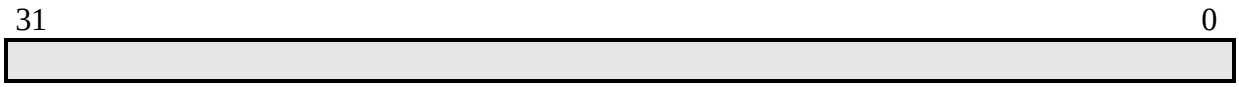


Рисунок 4.4 – Целочисленный формат с одинарной точностью без знака

4.3 Форматы числа с плавающей запятой

Все форматы чисел с плавающей запятой на ПЦОС состоят из трех полей: поля экспоненты (*e*), одного поля знакового разряда (*s*) и поля дробной части (*f*). Они представлены на рисунке 4.5.

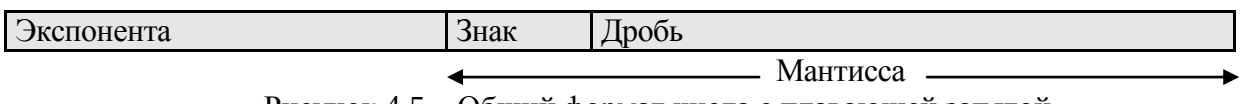


Рисунок 4.5 – Общий формат числа с плавающей запятой

Поле экспоненты – это число в форме дополнения до двух. Поле знака и поле дробной части могут рассматриваться как единая часть и определяются как поле мантиссы (man). Дробная часть в форме дополнения до двух объединяется со знаковым разрядом, который включает в себя старший значащий разряд для создания мантиссы. Мантисса используется для представления нормализованного числа в дополнительном коде. В нормализованном представлении включается старший незнаковый разряд, обеспечивая, таким образом, дополнительный разряд точности.

Основная формула (4.5) для вычисления значения числа с плавающей точкой дана ниже.

Значение числа с плавающей запятой:

$$x = s\bar{s}.f_2 \times 2^e \ , \tag{4.5}$$

где s – это значение одного бита, $\bar{s}$ – это инверсия значения одного бита, f – двоичное значение дробного поля и e – это десятичный эквивалент поля экспоненты.

Мантисса представляется нормализованным, дважды дополненным числом. В нормализованном представлении старший незнаковый разряд является неявным, что обеспечивает дополнительный разряд точности. Неявный знаковый разряд используется как:

- если s = 0, тогда два старших разряда мантиссы – 01;
- если s = 1, тогда два старших разряда мантиссы – 10;
- если единичный бит s равен 0, мантисса становится $01.f_2$,

где f_2 – двоичное представление дробного поля;

- если s = 1, мантисса становиться равной $10.f_2$,

где f_2 – двоичное представление дробного поля.

Например, если f_2=00000000001_2 и s=0, значение мантиссы (man) будет 01.00000000001_2. Если s = 1, значение man будет 10.00000000001_2.

Поле экспоненты – это число в форме дополнения до двух. По существу, поле экспоненты сдвигает двоичную запятую в мантиссе. Если экспонента положительна, двоичная запятая сдвигается вправо. Если экспонента отрицательна, двоичная запятая сдвигается влево.

Например, если man=01.00000000001_2 и e=11_{10}, тогда двоичная запятая переместится на одиннадцать разрядов вправо, полученное число 0100000000001_2, которое эквивалентно десятичному – 2049.

ПЦОС поддерживает три формата с плавающей запятой. Первый – короткий формат с плавающей запятой для непосредственных операндов с плавающей запятой, содержащих 4-разрядную экспоненту, 1 знаковый разряд и 11 разрядов для дробной части. Второй – это формат с одинарной точностью, содержащий 8-разрядную экспоненту, 1 знаковый разряд и 23-разряда для дробной части. Третий формат – это формат повышенной точности, содержащий 8-разрядную экспоненту, 1 знаковый разряд и 31 разряд для дробной части.

4.3.1 Короткий формат числа с плавающей запятой

В коротком формате с плавающей запятой число с плавающей запятой представляется 4-разрядным полем экспоненты (e) в дополнительном коде и 12-разрядным полем мантиссы (man) в дополнительном коде с неявным старшим незнаковым разрядом.

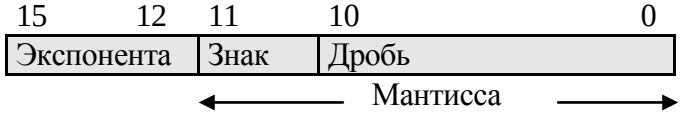


Рисунок 4.6 – Короткий формат числа с плавающей запятой

Для представления нуля в формате с плавающей запятой с одинарной точностью должны быть использованы следующие зарезервированные значения:

$e = -8$

$s = 0$

$f = 0$

Операции выполняются с неявной двоичной запятой между 10 и 11 разрядами. Число x с плавающей запятой в дополнительном коде в коротком формате с плавающей запятой представляется как:

$$x = 01.f_2 \times 2^e, \text{ если } s = 0 \quad (4.6)$$

$$x = 10.f_2 \times 2^e, \text{ если } s = 1 \quad (4.7)$$

$$x = 0, \text{ если } e = -8,\ s = 0,\ f = 0 \quad (4.8)$$

Следующие примеры иллюстрируют диапазон и точность короткого формата с плавающей запятой:

- наибольшее положительное: $x = (2 - 2^{-11}) \times 2^7 = 2.5594 \times 10^{-2}$;
- наименьшее положительное: $x = 1 \times 2^{-7} = 7.8125 \times 10^{-3}$;
- наибольшее отрицательное: $x = (-1 - 2^{-11}) \times 2^{-7} = -7.8163 \times 10^{-3}$;
- наименьшее отрицательное: $x = -2 \times 2^7 = -2.5600 \times 10^2$.

4.3.2 Формат числа с плавающей запятой с одинарной точностью

В формате с одинарной точностью число с плавающей запятой представлено 8-разрядным полем экспоненты (e) и 24-разрядным полем мантиссы (man) в дополнительном коде с неявным старшим незнаковым разрядом.

Операции выполняются с неявной двоичной запятой между 23 и 22 разрядами. Когда неявный старший незнаковый разряд определен, он размещается непосредственно слева от двоичной запятой. Число x с плавающей запятой представляется как

$$x = 01.f \times 2^e, \text{ если } s = 0 \quad (4.9)$$

$$x = 10.f \times 2^e, \text{ если } s = 1 \quad (4.10)$$

$$x = 0, \text{ если } e = -128 \quad (4.11)$$

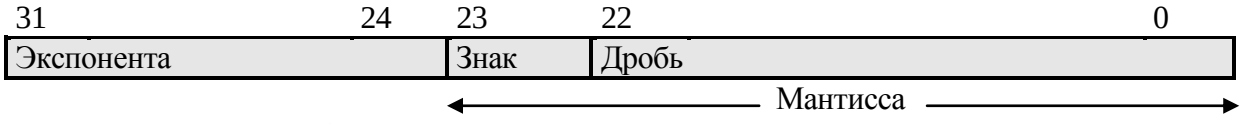


Рисунок 4.7 – Формат числа с плавающей запятой с одинарной точностью

Для представления нуля в формате с плавающей запятой с одинарной точностью должны быть использованы следующие зарезервированные значения:

$e = -128$

$s = 0$

$f = 0$

Следующие примеры иллюстрируют диапазон и точность формата с плавающей запятой с одинарной точностью:

- наибольшее положительное: $x = (2 - 2^{-23}) \times 2^{127} = 3.4028234 \times 10^{38}$;
- наименьшее положительное: $x = 1 \times 2^{-127} = 5.8774717 \times 10^{-39}$;
- наибольшее отрицательное: $x = (-1 - 2^{-23}) \times 2^{-127} = -5.8774724 \times 10^{-39}$;
- наименьшее отрицательное: $x = -2 \times 2^{127} = -3.4028236 \times 10^{38}$.

4.3.3 Формат числа с плавающей запятой с повышенной точностью

В формате с повышенной точностью число с плавающей запятой представляется 8-разрядным полем экспоненты (*e*) и 32-разрядным полем мантиссы (*man*) с неявным старшим незнаковым разрядом.

Операции выполняются с неявной двоичной запятой между 31 и 30 разрядами. Когда неявный старший незнаковый разряд определен, он размещается непосредственно слева от двоичной запятой. Число *x* с плавающей запятой устанавливается как:

$$x = 01.f \times 2^e, \text{ если } s = 0 \quad (4.12)$$

$$x = 10.f \times 2^e, \text{ если } s = 1 \quad (4.13)$$

$$x = 0, \text{ если } e = -128,\ s = 0,\ f = 0 \quad (4.14)$$

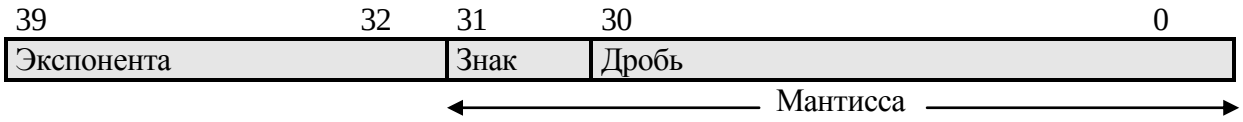


Рисунок 4.8 – Формат числа с плавающей запятой с повышенной точностью

Для представления нуля в формате с плавающей запятой с повышенной точностью должны использоваться следующие зарезервированные значения:

$e = -128$

$s = 0$

$f = 0$

Следующие примеры иллюстрируют диапазон и точность формата с повышенной точностью с плавающей запятой:

- наибольшее положительное: $x = (2 - 2^{-31}) \times 2^{127} = 3.4028236683 \times 10^{38}$;
- наименьшее положительное: $x = 1 \times 2^{-127} = 5.8774717541 \times 10^{-39}$;
- наибольшее отрицательное: $x = (-1 - 2^{-31}) \times 2^{-127} = -5.8774717569 \times 10^{-39}$;
- наименьшее отрицательное: $x = -2 \times 2^{127} = -3.4028236691 \times 10^{38}$.

4.3.4 Преобразования между форматами с плавающей запятой

Операции с плавающей запятой предполагают различные форматы для ввода и вывода. Эти форматы часто требуют преобразования из одного формата с плавающей запятой в другой (т. е. короткий формат с плавающей запятой в формат с плавающей запятой с повышенной точностью). Преобразования формата происходят автоматически аппаратно, без дополнительных затрат как часть операций с плавающей запятой. Четыре типа преобразования с примерами показаны на рисунках 4.9 – 4.12 (*s* – знаковый разряд экспоненты). Когда число в формате с плавающей запятой нулевого значения преобразуется в формат с большей точностью, оно всегда преобразуется в действительное представление нуля в данном формате.

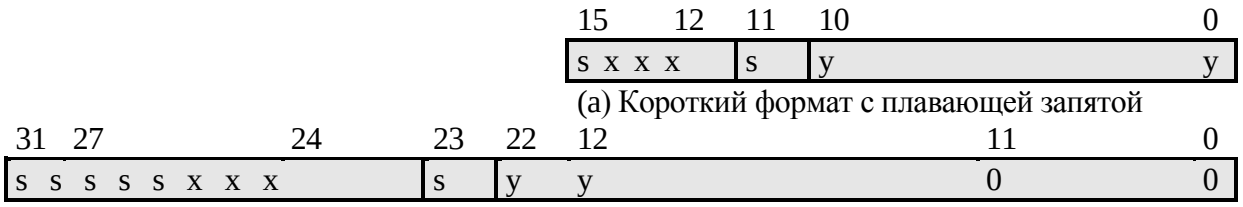


Рисунок 4.9 – Преобразование короткого формата с плавающей запятой в формат с плавающей запятой одинарной точности

При выполнении преобразования из короткого формата в формат с одинарной точностью поле экспоненты дополняется знаковым разрядом, а крайние справа 12 разрядов дробного поля заполняются нулями.

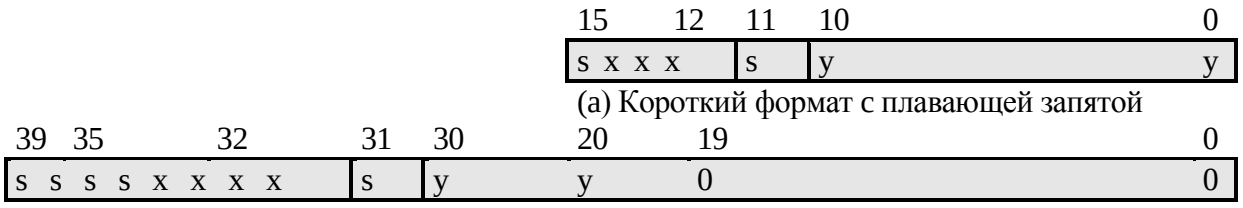


Рисунок 4.10 – Преобразование короткого формата с плавающей запятой в формат с плавающей запятой повышенной точности

При преобразовании из короткого формата в формат с повышенной точностью поле экспоненты дополняется знаковым разрядом, а крайние справа 20 разрядов дробного поля заполняются нулями.

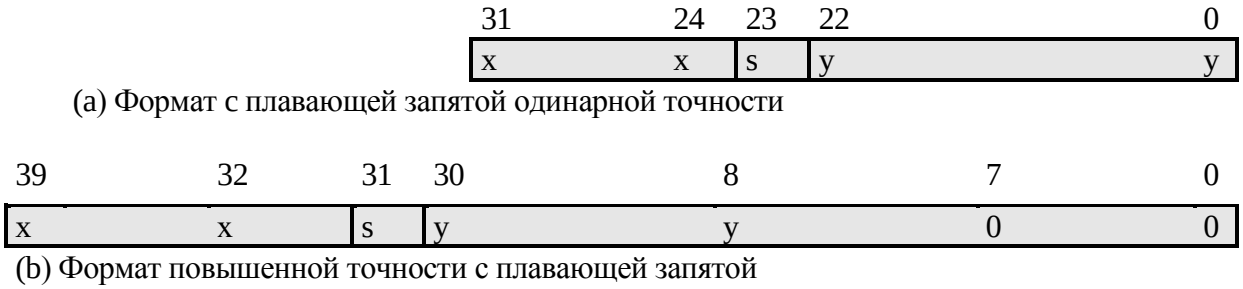


Рисунок 4.11 – Преобразование формата с плавающей запятой одинарной точности в формат с плавающей запятой повышенной точности

Если выполняется преобразование из формата с одинарной точностью в формат с повышенной точностью, то крайние справа 8 разрядов дробного поля заполняются нулями.

Рисунок 4.12 – Преобразование формата с плавающей запятой повышенной точности в формат с плавающей запятой одинарной точности

Если выполняется преобразование формата с повышенной точностью в формат с одинарной точностью, то крайние справа 8 разрядов дробного поля урезаются.

4.4 Умножение значений с плавающей запятой

Число с плавающей запятой может быть записано в формате с плавающей запятой согласно формуле:

$$a = a(man) \times 2^{a(exp)}, \quad (4.15)$$

где a(man) – мантисса и a(exp) – экспонента.

Результатом умножения чисел a и b является число c, определяемое следующим образом:

$$c = a \times b = a(man) \times b(man) \times 2^{(a(exp) + b(exp))} \quad (4.16)$$

$$c(man) = a(man) \times b(man); \quad c(exp) = a(exp) + b(exp) \quad (4.17)$$

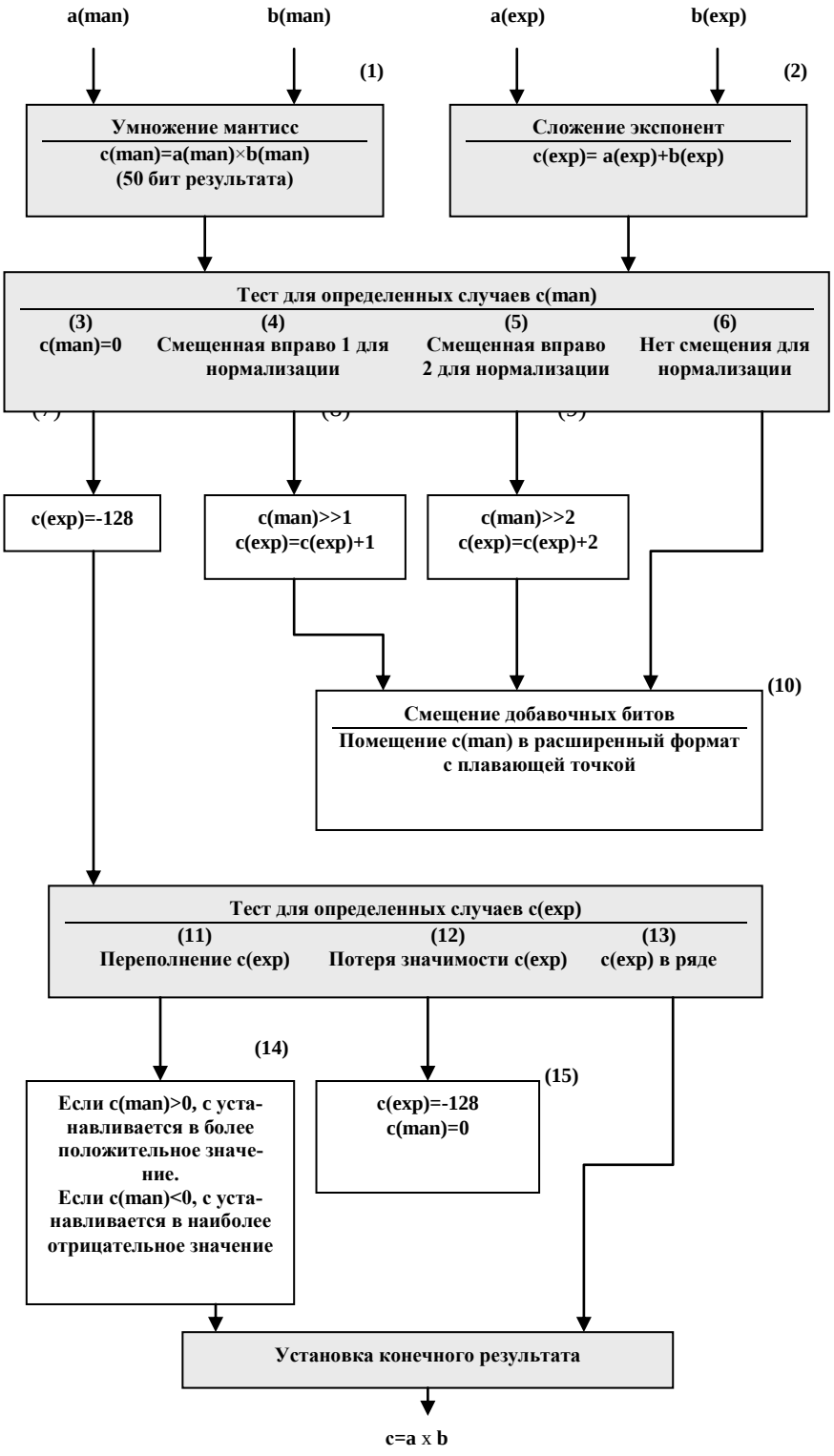


Рисунок 4.13 – Блок-схема для умножения с плавающей точкой

Когда выполняется умножение с плавающей запятой, предполагается, что исходные операнды всегда будут в формате с плавающей запятой с одинарной точностью. Если исходные операнды в коротком формате, они расширяются до формата с плавающей запятой с одинарной точностью. Если исходные операнды в формате с плавающей запятой с повышенной точностью, они сокращаются до формата с одинарной точностью. Эти преобра-

зования производятся автоматически аппаратно, без дополнительных затрат. Все результаты умножений с плавающей запятой предполагают формат с повышенной точностью. Эти умножения происходят в одном цикле.

На 1 этапе 24-разрядные мантиссы операндов источника умножаются, и на выходе получается 50-разрядный результат c(man). (Отметим, что входные и выходные данные всегда представлены нормализованными числами). На этапе 2 складываются экспоненты, получается c(exp). Этапы с 3 по 6 используются для проверки специальных случаев. На этапе 3 проверяется на равенство нулю мантисса c(man), которая представлена в формате с повышенной точностью. Если c(man) равна нулю, то на этапе 7 устанавливается c(exp) = -128, и обеспечивается, таким образом, представление нуля.

Этапы 4 и 5 нормализуют результат. Если необходим сдвиг вправо на один разряд, то на этапе 8 c(man) сдвигается вправо на один разряд и к c(exp) прибавляется единица. Если же необходим сдвиг вправо на два разряда, то на этапе 9 c(man) сдвигается вправо на два разряда и к c(exp) добавляется два. Этап 6 происходит, когда результат нормализован.

На этапе 10 c(man) приводится к формату с плавающей запятой повышенной точности. Этапы с 11 по 13 предназначены для проверки специальных случаев с c(exp). На этапе 14, если c(exp) переполнена в положительном направлении (этап 11), то c(exp) присваивается наибольшее положительное значение в формате с повышенной точностью. Если c(exp) переполнена в отрицательном направлении, то c(exp) присваивается наименьшее отрицательное значение в формате с повышенной точностью. Если произошла потеря значимости c(exp) (исчезновение значащих разрядов) (этап 12), то c присваивается значение ноль (этап 15), т. е. c(man) = 0 и c(exp) =-128.

Следующие примеры иллюстрируют как выполняется умножение с плавающей запятой на ПЦОС. Для этих примеров неявный старший незнаковый разряд задан явно.

Пример 4.1 – Умножение с плавающей запятой (обе мантиссы равны -2.0)

Дано:

$a = -2.0 \times 2^{a(exp)} = 10.00000000000000000000000 \times 2^{a(exp)}$

$b = -2.0 \times 2^{b(exp)} = 10.00000000000000000000000 \times 2^{b(exp)}$,

где a и b представлены в двоичном виде в соответствии с нормализованным форматом с плавающей запятой с одинарной точностью. Тогда:

$$10.00000000000000000000000 \times 2^{a(exp)}$$
$$\times\ 10.00000000000000000000000 \times 2^{b(exp)}$$
$$= 0100.00000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

Для приведения этого числа в соответствующий нормализованный формат необходимо сдвинуть мантиссу на два разряда вправо и прибавить двойку к экспоненте. Получается:

$$10.00000000000000000000000 \times 2^{a(exp)}$$
$$\times\ 10.00000000000000000000000 \times 2^{b(exp)}$$
$$= 01.00000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp)+2)}$$

При умножении с плавающей запятой экспонента результата может переполняться.

Это может произойти, когда экспонента изменяется при выполнении нормализации.

Пример 4.2 – Умножение с плавающей запятой (обе мантиссы равны 1.5)

Дано:

$a = 1.5 \times 2^{a(exp)} = 01.10000000000000000000000 \times 2^{a(exp)}$

$b = 1.5 \times 2^{b(exp)} = 01.10000000000000000000 \times 2^{b(exp)}$,

где a и b оба представлены в двоичном виде в соответствии с форматом с плавающей запятой повышенной точности. Тогда:

$$01.10000000000000000000 \times 2^{a(exp)}$$
$$\times 01.10000000000000000000 \times 2^{b(exp)}$$
$$= 0010.01000000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

Для приведения этого числа в соответствующий нормализованный формат, необходимо сдвинуть мантиссу на один разряд вправо и прибавить единицу к экспоненте.

$$01.10000000000000000000 \times 2^{a(exp)}$$
$$\times 01.10000000000000000000 \times 2^{b(exp)}$$
$$= 01.00100000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp)+1)}$$

Пример 4.3 – Умножение с плавающей запятой (обе мантиссы равны 1.0)

Дано:

$a = 1.0 \times 2^{a(exp)}) = 01.00000000000000000000 \times 2^{a(exp)}$

$b = 1.0 \times 2^{b(exp)} = 01.00000000000000000000 \times 2^{b(exp)}$,

где a и b оба представлены в двоичном виде соответственно формату с плавающей запятой одинарной точности. Тогда:

$$01.00000000000000000000 \times 2^{a(exp)}$$
$$\times 01.00000000000000000000 \times 2^{b(exp)}$$
$$= 0001.00000000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

Это число представлено в нормализованном формате. Следовательно, нет необходимости сдвигать мантиссу или изменять экспоненту.

В этих примерах рассмотрены случаи, где результатом действий над двумя нормализованными числами может быть нормализованное число со сдвигом на ноль, один или два.

Пример 4.4 – Умножение с плавающей запятой между положительным и отрицательным числами

Дано:

$a = 1.0 \times 2^{a(exp)} = 01.00000000000000000000 \times 2^{a(exp)}$

$b = 2.0 \times 2^{b(exp)} = 10.00000000000000000000 \times 2^{b(exp)}$

Тогда:

$$01.00000000000000000000 \times 2^{a(exp)}$$
$$\times 10.00000000000000000000 \times 2^{b(exp)}$$
$$= 1110.00000000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

Результатом будет $c = -2.0 \times 2^{(a(exp)+b(exp))}$

Пример 4.5 – Умножение с плавающей запятой на ноль

Любое умножение с плавающей запятой на ноль дает ноль ($f = 0$, $s = 0$, $exp = -128$).

4.5 Сложение и вычитание с плавающей запятой

Блок-схема для сложения с плавающей точкой представлена на рисунке 4.14. При сложении и вычитании с плавающей запятой два числа с плавающей запятой a и b должны быть определены как:

$$a = a(man) \times 2^{a(exp)} \quad (4.18)$$

$$b = b(man) \times 2^{b(exp)} \quad (4.19)$$

Сумма или разность a и b должна быть определена как:

$$c = a \pm b = \quad (4.20)$$

$$= (a(man) \pm (b(man) \times 2^{-(a(exp) - b(exp))})) \times 2^{a(exp)}, \quad \text{если } a(exp) \geq b(exp) \quad (4.21)$$

$$= ((a(man) \times 2^{-(b(exp) - a(exp))}) \pm b(man)) \times 2^{b(exp)}, \quad \text{если } a(exp) < b(exp) \quad (4.22)$$

Так как эта блок-схема предполагает данные со знаком, она так же подходит для вычитания с плавающей запятой. Для этого примера предполагается, что a(exp) ± b(exp). На этапе 1 исходные экспоненты сравниваются и c(exp) присваивается наибольшее значение исходной экспоненты. На этапе 2 d присваивается значение разности экспонент. На этапе 3 мантисса с наименьшей экспонентой, в этом случае a(man), сдвигается вправо на d разрядов с целью выравнивания мантисс. После того как мантиссы выровнены, они складываются (этап 4).

Этапы с 5 по 7 – проверка для специальных случаев c(man). Если c(man) равно нулю (этап 5), то c(exp) присваивается наибольшее отрицательное значение (этап 8) для получения корректного представления нуля. Если происходит переполнение c(man) (этап 6), то на этапе 9 c(man) сдвигается вправо на один разряд и к c(exp) добавляется единица. На этапе 10 результат нормализуется. Этапы 11 и 12 – для тестирования c(exp) в специальных случаях.

Если происходит переполнение c(exp), то c присваивается наибольшее положительное значение с повышенной точностью, если c(exp) положительно, в противном случае c(exp) присваивается наименьшее отрицательное значение с повышенной точностью.

Следующие примеры описывают операции сложения и вычитания с плавающей запятой. Предполагается, что данные находятся в формате с плавающей запятой с повышенной точностью.

Пример 4.6 – Сложение с плавающей запятой

Если будут складываться два нормализованных числа, дано:

$a = 1.5 = 01.10000000000000000000000000000000 \times 2^{0}$

$b = 0.5 = 01.00000000000000000000000000000000 \times 2^{-1}$

Необходимо сдвинуть b вправо на один разряд так, чтобы a и b имели одинаковые экспоненты. В результате получится:

$b = 0.5 = 00.10000000000000000000000000000000 \times 2^{0}$

Тогда:

$$01.10000000000000000000000000000000 \times 2^{0}$$
$$+ 00.10000000000000000000000000000000 \times 2^{0}$$
$$= 010.00000000000000000000000000000000 \times 2^{0}$$

Как и в случае умножения, необходимо сдвинуть точку на один разряд влево и прибавить единицу к экспоненте. В результате получится:

$$01.10000000000000000000000000000000 \times 2^{0}$$
$$+ 00.10000000000000000000000000000000 \times 2^{0}$$
$$= 01.00000000000000000000000000000000 \times 2^{1}$$

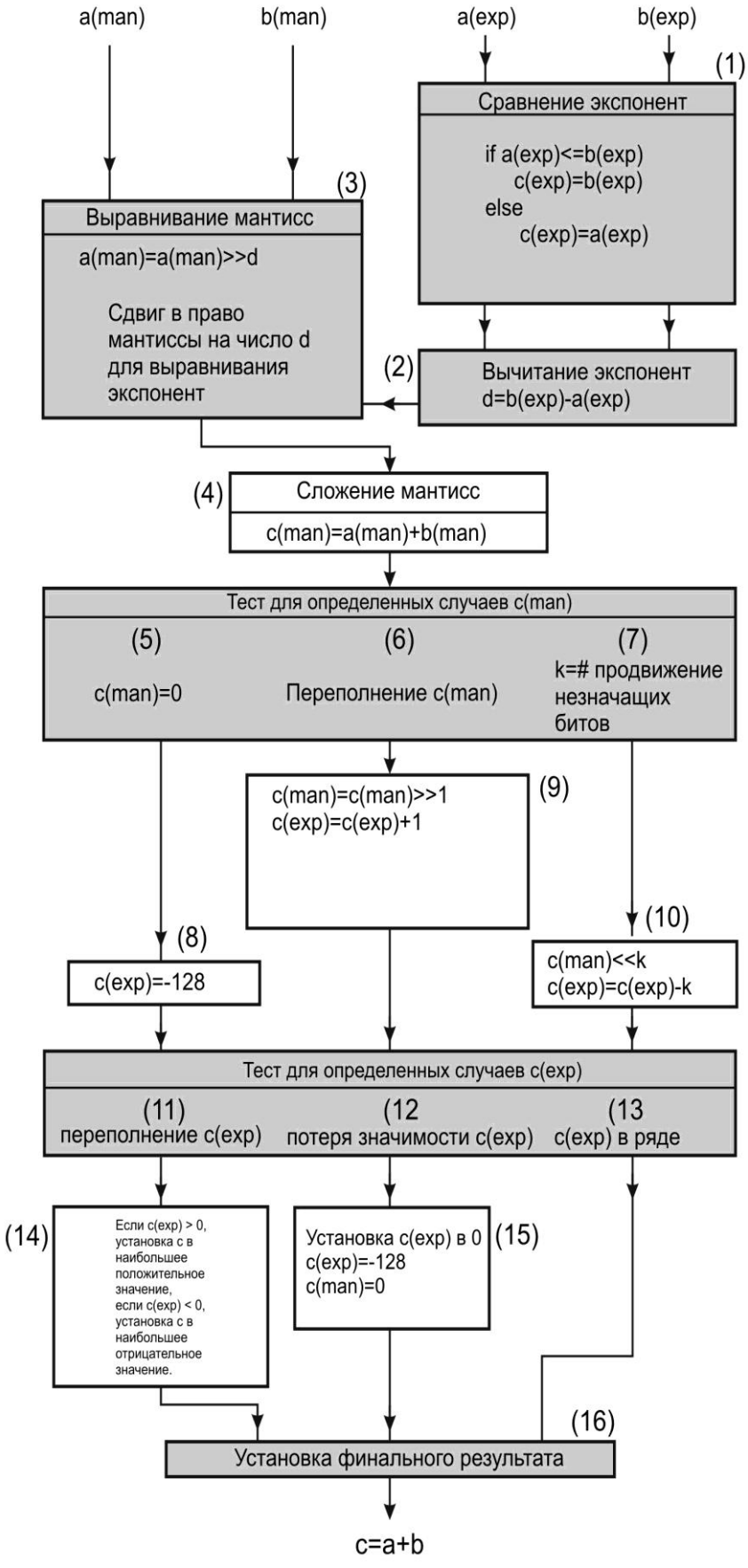


Рисунок 4.14 – Блок-схема для сложения с плавающей точкой

Пример 4.7 – Вычитание с плавающей запятой

Дано:

$a = 01.00000000000000000000000000000001 \times 2^0$

$b = 01.00000000000000000000000000000000 \times 2^0$

Должна быть выполнена операция a – b. Мантиссы уже выравнены, так как два числа имеют одинаковые экспоненты. В результате происходит большая потеря точности верхних разрядов как показано ниже:

$01.00000000000000000000000000000001 \times 2^0$

$- 01.00000000000000000000000000000000 \times 2^0$

$= 00.00000000000000000000000000000001 \times 2^0$

Результат должен быть нормализован. В этом случае требуется сдвиг влево на 31.

Соответственно изменяется экспонента результата. В результате получится:

$01.00000000000000000000000000000001 \times 2^0$

$- 01.00000000000000000000000000000000 \times 2^0$

$= 01.00000000000000000000000000000000 \times 2^{-31}$

Пример 4.8 – Сложение с плавающей запятой со сдвигом на 32 разряда

Этот пример иллюстрирует ситуацию, где полный сдвиг на 32 разряда необходим для нормализации результата. Дано:

$a = 01.11111111111111111111111111111111 \times 2^{127}$

$b = 10.00000000000000000000000000000000 \times 2^{127}$

Должна быть выполнена операция a + b.

$01.11111111111111111111111111111111 \times 2^{127}$

$+ 10.00000000000000000000000000000000 \times 2^{127}$

$= 11.11111111111111111111111111111111 \times 2^{127}$

Для нормализации результата требуется сдвиг влево на 32 разряда и вычитание 32 из экспоненты. В результате получится:

$01.11111111111111111111111111111111 \times 2^{127}$

$+ 10.00000000000000000000000000000000 \times 2^{127}$

$= 10.00000000000000000000000000000000 \times 2^{95}$

Пример 4.9 – Сложение/вычитание с плавающей запятой и ноль

Когда выполняется сложение и вычитание чисел с нулем в формате с плавающей запятой, справедливы следующие соотношения:

$a \pm 0 = a \ (a \neq 0)$

$0 \pm 0 = 0$

$0 - a = -a \ (a \neq 0)$

4.6 Использование команды NORM для нормализации

Команда NORM берет число с плавающей запятой с повышенной точностью, если оно не нормализовано, то нормализует его. Так как в предположении число не нормализовано,

также предполагается не пустым старший незнаковый разряд. Команда NORM выполняет следующие три этапа:

- 1 этап – размещает старший незнаковый разряд числа с плавающей запятой;
- 2 этап – сдвигает влево число для нормализации;
- 3 этап – устанавливает экспоненту.

Данное значение a с плавающей запятой с повышенной точностью не нормализовано, нормализация показана на рисунке 4.15.

Пример 4.10 – Команда NORM. Форматы данных и операции с плавающей запятой

Предположим, что регистр повышенной точности содержит значения:

man = 00000000000000000001000000000001, exp = 0

Когда выполняется нормализация числа (предполагается, что число не нормализовано), то подразумевается, что с двоичной точкой число имеет вид:

man = 0.00000000000000000001000000000001, exp = 0

Знак этого числа расширен на один разряд, так что мантисса содержит 33 разряда.

man = 00.00000000000000000001000000000001, exp = 0

Результат после перемещения старшего незнакового разряда и выполнения сдвига определяется следующим образом:

man = 01.00000000001000000000000000000000, exp = -19

После удаления резервного разряда формируется окончательное 32-разрядное значение нормализованного числа с плавающей запятой:

man = 00000000001000000000000000000000, exp = -19

Команда NORM полезна для вычисления количества начальных нулей или начальных единиц в 32-разрядном слове. Если начальное значение экспоненты равно нулю, абсолютное значение конечного результата экспоненты является количеством начальных единиц или нулей. Эта команда также используется для обработки ненормализованных чисел с плавающей запятой.

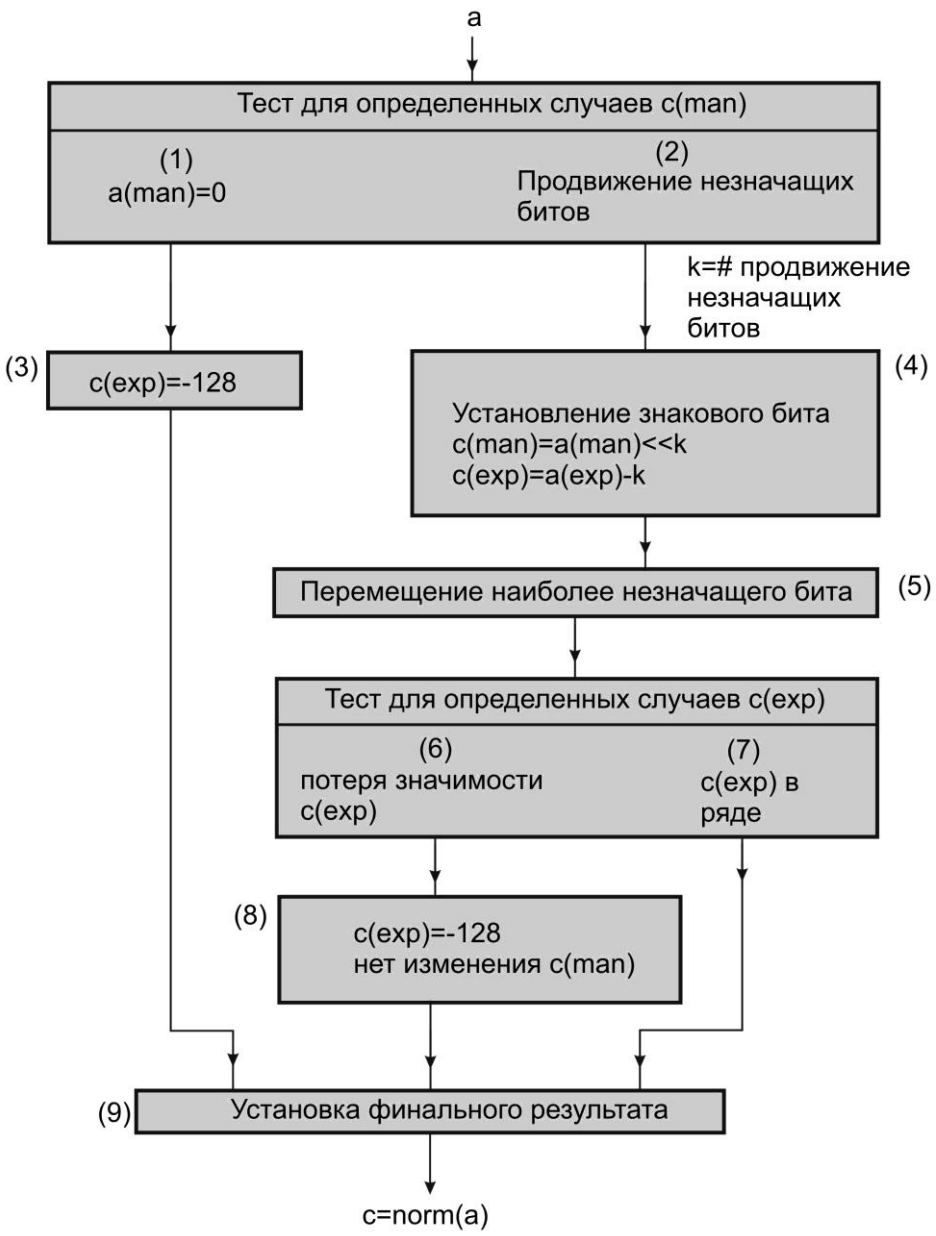


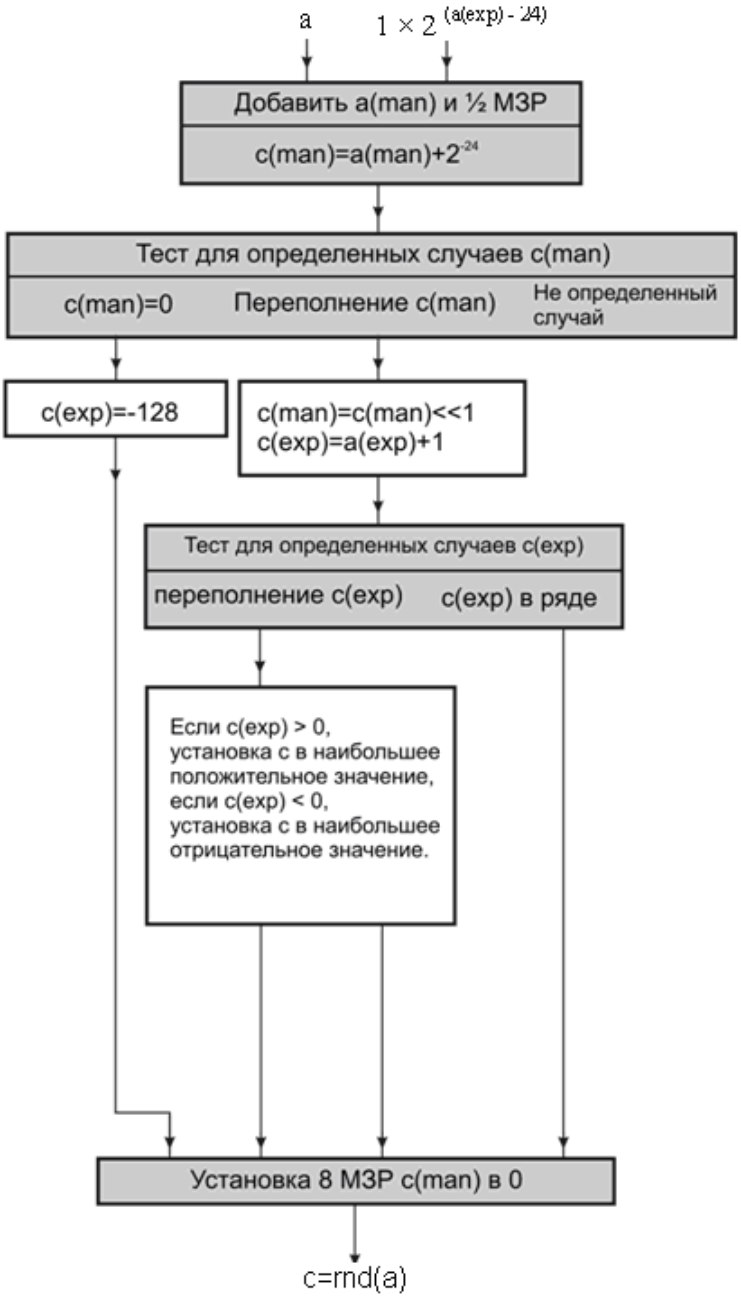
Рисунок 4.15 – Блок-схема выполнения команды NORM

4.7 Округление: команда RND

RND команда округляет числа из формата с плавающей запятой повышенной точности в формат с плавающей запятой одинарной точности. Операция округления аналогична сложению с плавающей запятой. При округлении данного числа a, сначала выполняется следующая операция:

$$c = a(man) \times 2^{a(exp)} + (1 \times 2^{(a(exp) - 24)}) \qquad (4.23)$$

Затем выполняется преобразование из формата с плавающей запятой с повышенной точностью в формат с плавающей запятой с одинарной точностью.



МЗР – младший значащий разряд

Рисунок 4.16 – Блок-схема для округления числа с плавающей запятой с помощью команды RND

4.8 Преобразование значения с плавающей запятой в целое

Блок-схема для преобразования значения с плавающей запятой в целое показана на рисунке 4.17.

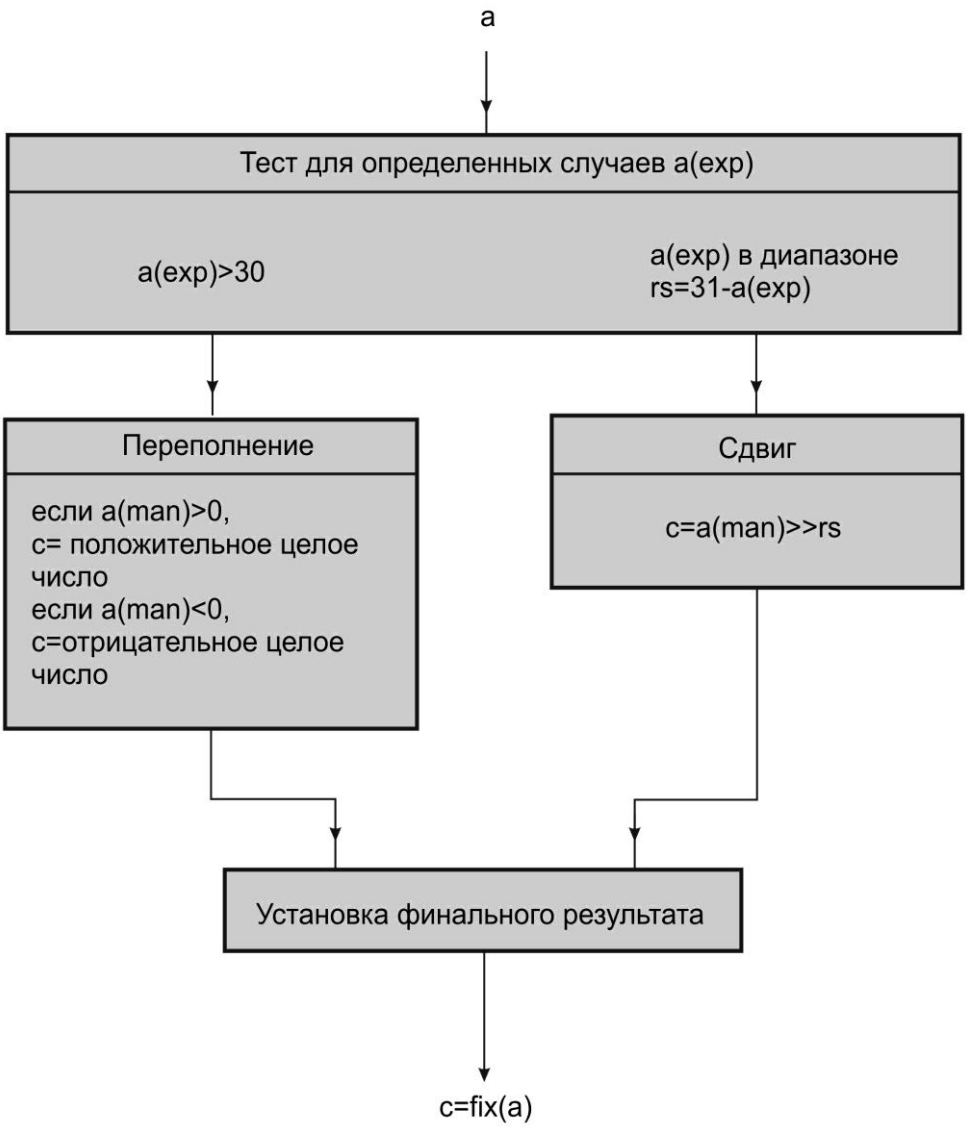


Рисунок 4.17 – Блок-схема преобразования значения с плавающей запятой в целое с помощью команды FIX

Преобразование значения с плавающей запятой в целое, используя команду FIX, позволяет выполнять преобразование чисел в формате с плавающей запятой с одинарной точностью в целые с одинарной точностью в одном цикле. Преобразование значения с плавающей запятой *a* в целое будет обозначаться *fix(a).* Преобразование не приведет к переполнению, если преобразуемое число *a* находится в диапазоне:

$$-2^{31} \le a \le 2^{31} - 1 \qquad (4.24)$$

Сначала необходимо определить что:

$$a(\text{exp}) \le 30 \qquad (4.25)$$

Если это не выполняется, то происходит переполнение. Если переполнение произошло в положительном направлении, на выходе будет наибольшее положительное целое. Если переполнение произошло в отрицательном направлении, на выходе будет получено наименьшее отрицательное число. Если a(exp) попадает в разрешенный диапазон, то для a(man), включая неявный разряд, производится расширение знака и сдвиг вправо (rs) на величину:

$$\text{rs} = 31 - \text{a(exp)} \qquad (4.26)$$

Этот сдвиг вправо (rs) сдвигает разряды в соответствии с дробной частью мантиссы. Например:

если $0 \le a < 1$, тогда fix(a) = 0;

если $-1 \le a < 0$, тогда fix(a) = -1.

4.9 Преобразование целого в значение с плавающей запятой командой FLOAT

Преобразование целого значения в значение с плавающей запятой, используя команду FLOAT, позволяет осуществить преобразование целого с одинарной точностью в число в формате с плавающей запятой повышенной точности.

Блок-схема этого преобразования дана на рисунке 4.18.

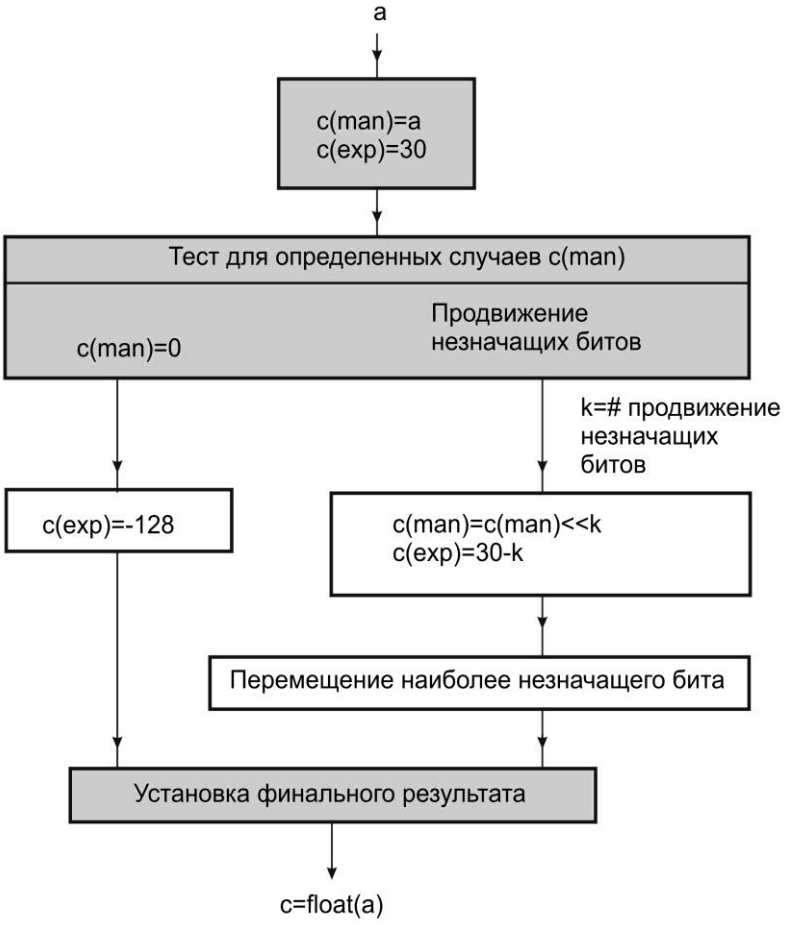


Рисунок 4.18 – Блок-схема преобразования целого числа в число с плавающей запятой с помощью команды FLOAT

5 Адресация

ПЦОС поддерживает пять групп режимов адресации. Внутри группы могут быть использованы шесть типов адресации, которые обеспечивают доступ к данным в памяти, регистрам и командным словам. В этом разделе подробно рассматриваются операции, кодирование и применение режимов адресации, описано управление системными стеком, очередями и исключениями из очереди в памяти.

В данном разделе рассмотрены следующие основные темы:

- типы адресации (подраздел 5.1):
    - регистровый;
    - прямой;
    - косвенный;
    - короткий непосредственный;
    - длинный непосредственный;
    - относительный (относительно PC);
- группы режимов адресации (подраздел 5.2):
    - основные режимы адресации;
    - трехоперандные режимы адресации;
    - параллельные режимы адресации;
    - режим длинной непосредственной адресации;
    - режимы адресации условных переходов;
- циклическая адресация (подраздел 5.3);
- битреверсная адресация (подраздел 5.4);
- управление стеком системы (подраздел 5.5).

5.1 Типы адресации

Шесть типов адресации делают возможным доступ к данным в памяти, регистрам и командным словам:

- регистровый;
- прямой;
- косвенный;
- короткий непосредственный;
- длинный непосредственный;
- относительный (относительно PC).

Некоторые типы адресации присутствуют в одних командах и отсутствуют в других.

По этой причине типы адресации используются в пяти различных группах режимов адресации:

1) основные режимы адресации (G):
    - регистровый;
    - прямой;
    - косвенный;
    - короткий непосредственный;
2) трехоперандные режимы адресации (T):
    - регистровый;

- косвенный;

3) режимы параллельной адресации (P):

- регистровый;
- косвенный;

4) режим длинной непосредственной адресации:

- длинной непосредственной;

5) режимы адресации условных переходов (B):

- регистровый;
- относительный.

В первую очередь будут рассмотрены шесть типов адресации, затем пять групп режимов адресации.

5.1.1 Регистровая адресация

В регистровой адресации операнд содержится в регистре ЦПУ как показано в примере: ABSF R1; R1 = | R1 |

Синтаксис для регистров ЦПУ, синтаксис ассемблера и назначение этих регистров приведены в таблице 5.1.

Таблица 5.1 – Регистры ЦПУ, их назначение и синтаксис ассемблера

Адрес регистра ЦПУ	Синтаксис ассемблера	Назначение
00h	R0	Регистр повышенной точности 0
01h	R1	Регистр повышенной точности 1
02h	R2	Регистр повышенной точности 2
03h	R3	Регистр повышенной точности 3
04h	R4	Регистр повышенной точности 4
05h	R5	Регистр повышенной точности 5
06h	R6	Регистр повышенной точности 6
07h	R7	Регистр повышенной точности 7
08h	AR0	Вспомогательный регистр 0
09h	AR1	Вспомогательный регистр 1
0Ah	AR2	Вспомогательный регистр 2
0Bh	AR3	Вспомогательный регистр 3
0Ch	AR4	Вспомогательный регистр 4
0Dh	AR5	Вспомогательный регистр 5
0Eh	AR6	Вспомогательный регистр 6
0FH	AR7	Вспомогательный регистр 7
10h	DP	Указатель страницы данных
11h	IR0	Индексный регистр 0
12h	IR1	Индексный регистр 1
13h	BK	Регистр размера блока
14h	SP	Указатель системного стека
15h	ST	Регистр состояния
16h	IE	Разрешение прерываний ЦПУ/ПДП
17h	IF	Флаги прерываний ЦПУ
18h	IOF	Флаги ввода-вывода
19h	RS	Регистр адреса начала повторения
1Ah	RE	Регистр адреса конца повторения
1Bh	RC	Счетчик повторений

5.1.2 Прямая адресация

В прямой адресации адрес данных формируется путем объединения восьми младших разрядов указателя страницы данных (DP) с 16 младшими разрядами командного слова (expr). В результате программист имеет возможность обращаться к большому адресному пространству (256 страниц по 64К слов на каждой) и нет необходимости изменять указатель страницы. На рисунке 5.1 приведено формирование адреса данных.

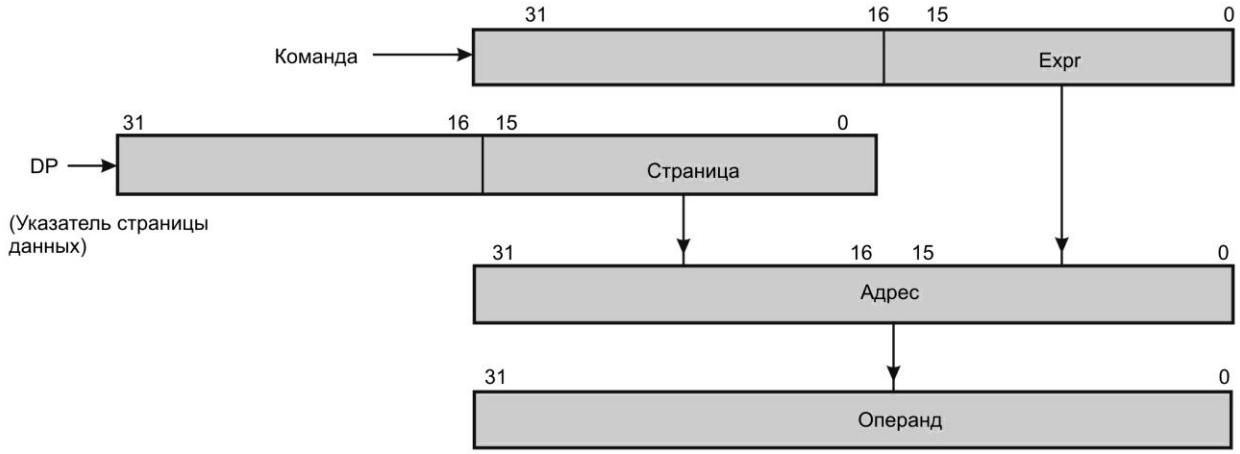


Рисунок 5.1 – Прямая адресация

Синтаксис и операции при прямой адресации приведены ниже.

Синтаксис: @expr

Операция: адрес = DP объединен с expr

В примере 5.1 приводится пример команды с данными до и после выполнения команды.

Пример 5.1 – Прямая адресация

ADDI @OBCDEh, R7

Перед выполнением команды:	После выполнения команды:
DP = 8Ah	DP = 8Ah
R7 = 0h	R7 = 12345678h
Данные в 8ABCDEh = 12345678h	Данные в 8ABCDEh = 12345678h

5.1.3 Косвенная адресация

К косвенной адресации обращаются для определения адреса операнда в памяти, используя вспомогательный регистр, а также дополнительное смещение и индексные регистры. В косвенной адресации используются только 24 младших разряда вспомогательных регистров и индексных регистров. Арифметические операции над содержимым регистров (24 младшими разрядами без знака) выполняются арифметическими устройствами вспомогательных регистров (ARAUn). Старшие восемь разрядов не изменяются.

Гибкость косвенной адресации возможна потому, что ARAUn на ПЦОС использованы для изменения вспомогательных регистров параллельно с операциями внутри главного ЦПУ. Косвенная адресация определяется пятиразрядным полем в командном слове, определенным как поле mod. Смещение – это либо восьмиразрядное целое без знака, содержащееся в

командном слове или неявное смещение на один. Два индексных регистра IR0 и IR1 также могут быть использованы в косвенной адресации. В некоторых случаях дополнительно можно использовать циклическую или битреверсную адресацию. Механизм формирования адресов в циклической адресации рассмотрен в подразделе 5.3, битреверсной – в подразделе 5.4.

В таблице 5.2 представлены разные виды косвенной адресации с соответствующим каждому виду полем модификации (mod), синтаксисом ассемблера, операцией и функцией.

Далее приведены 18 примеров, показывающих операцию для каждого вида косвенной адресации.

Пример 5.2 – Косвенная адресация через вспомогательный регистр

Адрес операнда, который будет выбран, содержится во вспомогательном регистре (ARn).

Операция: адрес операнда = ARn

Синтаксис: * ARn

Поле модификации: 11000

Рисунок 5.2 – Косвенная адресация через вспомогательный регистр

Пример 5.3 – Косвенная адресация с предварительным добавлением смещения

Выбираемый адрес операнда – это сумма содержимого вспомогательного регистра (ARn) и смещения (disp). Смещение – это либо восьмиразрядное целое без знака, содержащееся в командном слове, либо неявное значение единицы.

Операция**:** адрес операнда = ARn + disp

Синтаксис ассемблера: * + ARn (disp)

Поле модификации: 00000

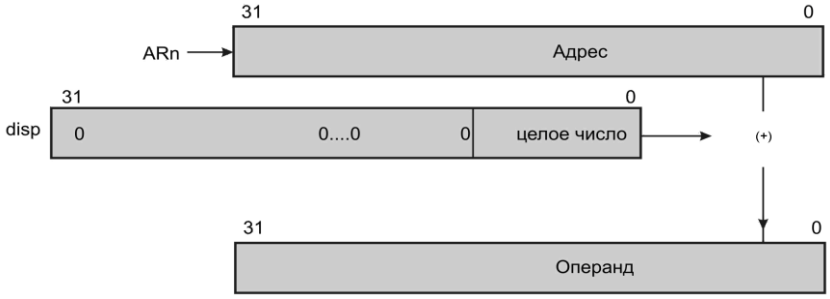


Рисунок 5.3 – Косвенная адресация с предварительным добавлением смещения

Пример 5.4 – Косвенная адресация с предварительным вычитанием смещения

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра минус смещение (disp). Смещение – это или восьмиразрядное целое, содержащееся в командном слове, или неявное значение единицы.

Операция: адрес операнда = ARn - disp

Синтаксис ассемблера: * - ARn (disp)

Поле модификации: 00001

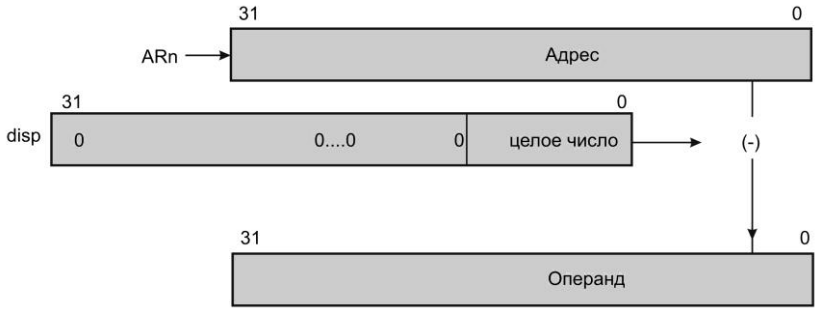


Рисунок 5.4 – Косвенная адресация с предварительным вычитанием смещения

Пример 5.5 – Косвенная адресация с предварительным добавлением смещения и модификацией

Адрес выбранного операнда – это сумма вспомогательного регистра (ARn) и смещения (disp). Смещение – это каждый восьмой бит беззнакового целого, содержащегося в инструкционном слове, или подразумевается значение 1. После этого данные выбираются, вспомогательный регистр обновляется со сгенерированным адресом.

Операция**:** адрес операнда = ARn + disp

ARn = ARn + disp

Синтаксис ассемблера: *++ARn (disp)

Модификационное поле: 00010

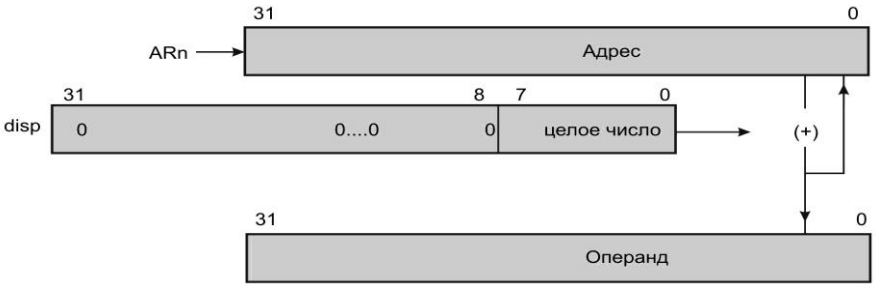


Рисунок 5.5 – Косвенная адресация с предварительным добавлением смещения и модификацией

Пример 5.6 – Косвенная адресация с предварительным вычитанием смещения и модификацией

Выбираемый адрес операнда – это разность между содержимым вспомогательного регистра (ARn) и смещения (disp). Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы. После выборки данных вспомогательный регистр загружается сгенерированным адресом.

Операция: адрес операнда = ARn – disp,

ARn = ARn + disp

Синтаксис ассемблера: * -- ARn (disp)

Поле модификации: 00011

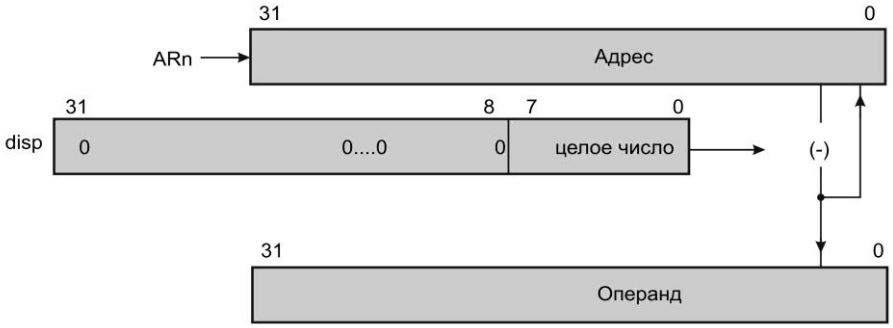


Рисунок 5.6 – Косвенная адресация с предварительным вычитанием смещения и модификацией

Пример 5.7 – Косвенная адресация с последующим добавлением смещения и модификацией

Выбранный адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выбора операнда смещение (disp) добавляется к содержимому вспомогательного регистра. Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы.

Операция: адрес операнда = ARn,
ARn = ARn + disp

Синтаксис ассемблера: *ARn ++ (disp)

Поле модификации: 00100

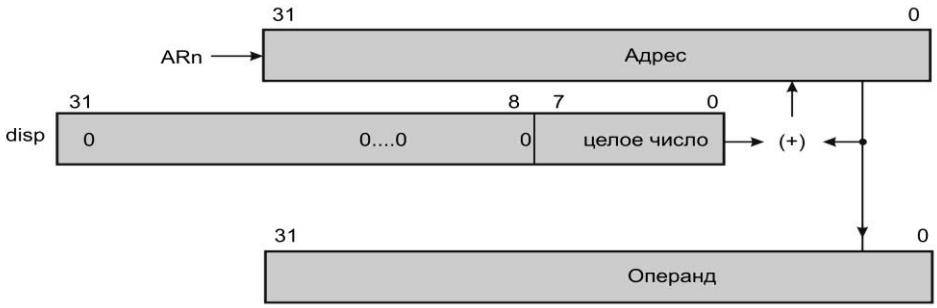


Рисунок 5.7 – Косвенная адресация с последующим добавлением смещения и модификацией

Пример 5.8 – Косвенная адресация с последующим вычитанием смещения и модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда смещение (disp) – вычитается из содержимого вспомогательного регистра. Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы.

Операция: адрес операнда = ARn,
ARn = ARn - disp

Синтаксис ассемблера: * ARn -- (disp)

Поле модификации: 00101

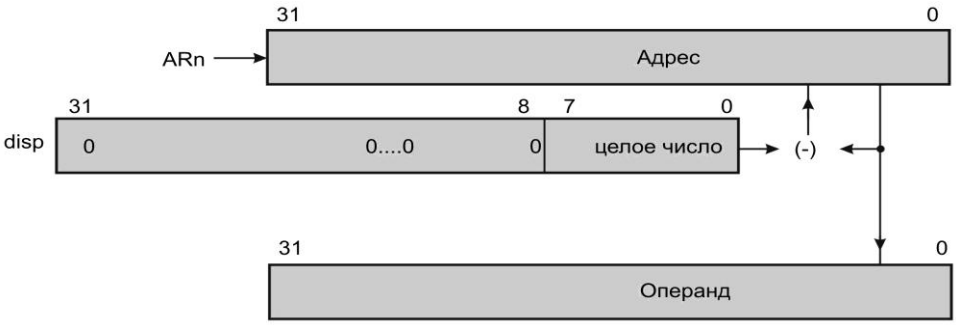


Рисунок 5.8 – Косвенная адресация с последующим вычитанием смещения и модификацией

Пример 5.9 – Косвенная адресация с последующим сложением смещения и циклической модификацией

Выбираемый операнд соответствует содержимому вспомогательного регистра (ARn). После выбора операнда смещение добавляется к содержимому вспомогательного регистра, использующего циклическую адресацию. Результат используется для изменения вспомогательного регистра. Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы.

Операция: адрес операнда = ARn,

ARn = circ(ARn + disp)

Синтаксис ассемблера: * ARn ++ (disp) %

Поле модификации: 00110

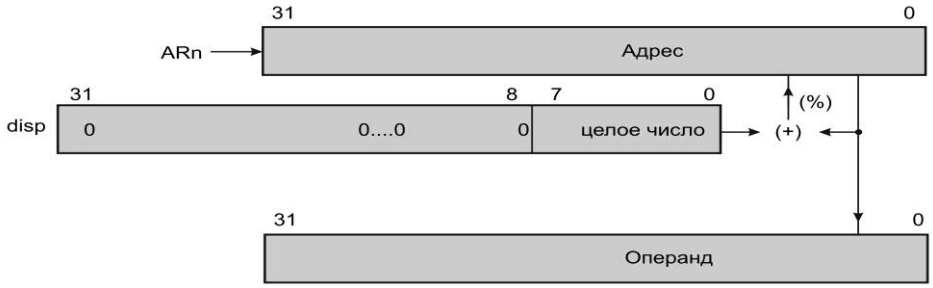


Рисунок 5.9 – Косвенная адресация с последующим сложением смещения и циклической модификацией

Пример 5.10 – Косвенная адресация с последующим вычитанием смещения и циклической модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выбора операнда смещение (disp) вычитается из содержимого вспомогательного регистра, используя циклическую адресацию. Результат используется для изменения вспомогательного регистра. Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы.

Операция: адрес операнда = ARn,

ARn = circ(ARn - disp)

Синтаксис ассемблера: * ARn -- (disp) %

Поле модификации: 00111

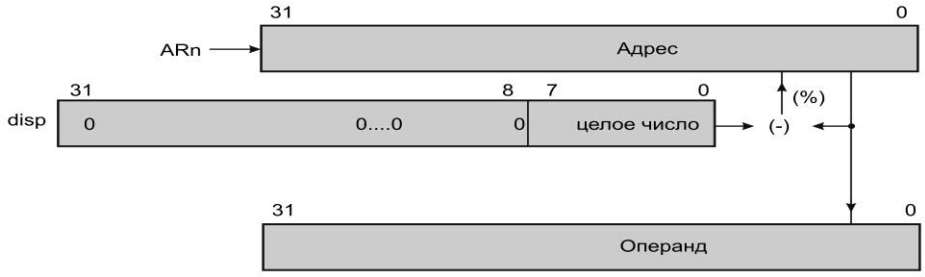


Рисунок 5.10 – Косвенная адресация с последующим вычитанием смещения и циклической модификацией

Пример 5.11 – Косвенная адресация с предварительным добавлением индекса

Выбираемый адрес операнда – это сумма содержимого вспомогательного регистра (ARn) и индексного регистра (IR0 или IR1).

Операция: адрес операнда = ARn + IRm

Синтаксис ассемблера: * + ARn (IRm)

Поле модификации: 01000, если m=0,

10000, если m=1.

Рисунок 5.11 – Косвенная адресация с предварительным добавлением индекса

Пример 5.12 – Косвенная адресация с предварительным вычитанием индекса

Выбираемый адрес операнда – это разность между содержимым вспомогательного регистра (ARn) и индексного регистра (IR0 или IR1).

Операция**:** адрес операнда = ARn - IRm

Синтаксис ассемблера: * - ARn (IRm)

Поле модификации: 01001, если m=0,

10001, если m=1.

Рисунок 5.12 – Косвенная адресация с предварительным вычитанием индекса

Пример 5.13 – Косвенная адресация с предварительным сложением индекса и модификацией

Выбираемый адрес операнда – это сумма содержимого вспомогательного регистра (ARn) и индексного регистра (IR0 или IR1). После выборки данных содержимое вспомогательного регистра заменяется сгенерированным адресом.

Операция: адрес операнда = ARn + IRm,

ARn = ARn + IRm

Синтаксис ассемблера**:** * ++ ARn (IRm)

Поле модификации: 01010, если m=0,

10010, если m=1.

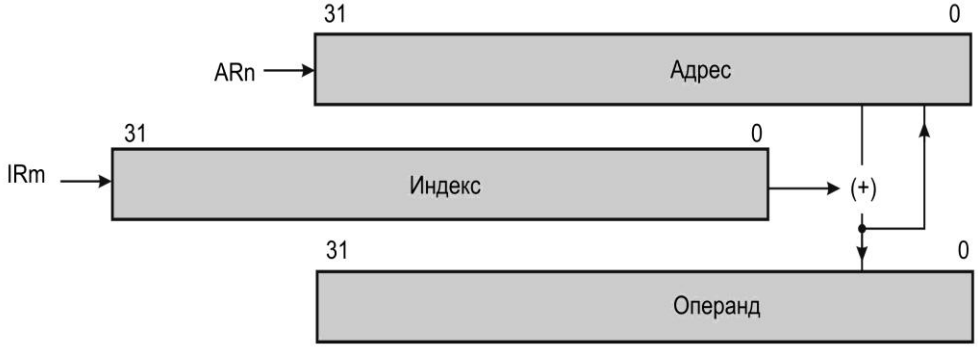


Рисунок 5.13 – Косвенная адресация с предварительным сложением индекса и модификацией

Пример 5.14 – Косвенная адресация с предварительным вычитанием индекса и модификацией

Выбираемый адрес операнда – это разность между содержимым вспомогательного регистра (ARn) и индексного регистра (IRm). Результирующий адрес становится новым содержимым вспомогательного регистра.

Операция**:** адрес операнда = ARn - IRm,

ARn = ARn - IRm

Синтаксис ассемблера: * -- ARn (IRm)

Поле модификации: 01011, если m=0,

10011, если m=1.

Рисунок 5.14 – Косвенная адресация с предварительным вычитанием индекса и модификацией

Пример 5.15 – Косвенная адресация с последующим прибавлением индекса и модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда содержимое индексного регистра (IR0 или IR1) прибавляется к содержимому вспомогательного регистра.

Операция**:** адрес операнда = ARn,
ARn = ARn + IRm

Синтаксис ассемблера**:** * ARn ++ (IRm)

Поле модификации: 01100, если m=0,
10100, если m=1.

Рисунок 5.15 – Косвенная адресация с последующим прибавлением индекса и модификацией

Пример 5.16 – Косвенная адресация с последующим вычитанием индекса и модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда содержимое индексного регистра (IR0 или IR1) вычитается из вспомогательного регистра.

Операция: адрес операнда = ARn,
ARn = ARn – IRm

Синтаксис ассемблера: * ARn -- (IRm)

Поле модификации: 01101, если m=0,
10101, если m=1.

Рисунок 5.16 – Косвенная адресация с последующим вычитанием индекса и модификацией

Пример 5.17 – Косвенная адресация с последующим сложением с индексом и циклической модификацией

Адрес выбранного операнда – это содержимое вспомогательного регистра (ARn). После выбора операнда, индексный регистр (IR0 или IR1) добавляется к вспомогательному регистру. Это значение преобразуется через круговую адресацию и перезаписывается содержимым вспомогательного регистра.

Операция**:** адрес операнда = ARn,

ARn = circ(ARn + IRm)

Синтаксис ассемблера: *ARn ++ (IRm) %

Поле модификации: 01110, если m=0,

10110, если m=1.

Рисунок 5.17 – Косвенная адресация с последующим сложением с индексом и циклической модификацией

Пример 5.18 – Косвенная адресация с последующим вычитанием индекса и циклической модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда содержимое индексного регистра (IR0 или IR1) вычитается из содержимого вспомогательного регистра. Это значение вычисляется, используя циклическую адресацию, и замещает содержимое вспомогательного регистра.

Операция: адрес операнда = ARn, ARn = circ(ARn - IRm)

Синтаксис ассемблера: * ARn -- (IRm)%

Поле модификации: 01111 , если m=0,

10111 , если m=1.

Рисунок 5.18 – Косвенная адресация с последующим вычитанием индекса и циклической модификацией

Пример 5.19 – Косвенная адресация с последующим прибавлением индекса и битреверсной модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда содержимое индексного регистра (IR0) складывается с содержимым вспомогательного регистра. Сложение выполняется с обратным распростране-

нием переноса, который может быть использован для выдачи битреверсного адреса (B). Это значение замещает содержимое вспомогательного регистра.

Операция: адрес операнда = ARn,

ARn = B(ARn + IR0)

Синтаксис ассемблера: * ARn ++ (IR0)B

Поле модификации: 11001

Рисунок 5.19 – Косвенная адресация с последующим прибавлением индекса и битреверсной модификацией

5.1.4 Короткая непосредственная адресация

В короткой непосредственной адресации операнд – это 16-разрядная непосредственная величина, содержащаяся в 16 младших значащих разрядах командного слова (expr). В зависимости от типа данных, требуемых для команды, короткий непосредственный операнд может быть целым в дополнительном коде, целым без знака или числом с плавающей запятой.

Синтаксис для этого режима приведен ниже.

Синтаксис: expr

В примере 5.20 приведен пример команды с данными до и после команды.

Пример 5.20 – Короткая непосредственная адресация

SUBI 1, R0

До команды:	После команды:
R0 = 0h	R0 = 0FFFFFFFFh

5.1.5 Длинная непосредственная адресация

В длинной непосредственной адресации операнд – это 24-разрядная непосредственная величина, содержащаяся в 24 младших разрядах командного слова (expr). Синтаксис для этого режима приведен ниже.

Синтаксис: expr

В примере 5.21 дан пример команды с данными до и после выполнения команды.

Пример 5.21 – Длинная непосредственная адресация

BR 8000h

До команды:	После команды:
PC = 0h	PC = 8000h

5.1.6 Относительная адресация (относительно PC)

Относительная адресация используется для ветвления. Ассемблер использует src (метку или адрес), определенный пользователем и генерирует смещение.

Если переход стандартный, то смещение равно: метка − (PC + 1).

Если переход задержанный, то смещение равно: метка − (PC + 3).

Смещение хранится как 16-разрядное целое со знаком в младших разрядах командного слова. Смещение добавляется к значению PC, если условие выполнено.

Синтаксис: expr

В примере 5.22 представлена команда с данными до и после выполнения команды.

Пример 5.22 – Адресация относительно PC

BU NEWPC; PC=1001h, NEWPC=1005h, смещение=3

До команды:	После команды:
PC=1001h	PC=1005h

5.2 Группы режимов адресации

Шесть типов адресации, приведенные в разделе 5.1, используются для формирования следующих пяти групп режимов адресации:

- основные режимы адресации (G);
- трехоперандные режимы адресации (T);
- параллельные режимы адресации (P);
- режим длинной непосредственной адресации;
- режимы адресации условных переходов (B).

5.2.1 Основные режимы адресации

Инструкции, которые используют режимы основной адресации, являются командами общего назначения, такими как ADDI, MPYF и LSH. Такие команды обычно представляются в форме:

dst «операция» src → dst, где dst – операнд назначения, src – операнд источника, «операция» – операция, которая будет выполнена, используя основные режимы адресации для определения указанных операндов. Разряды 31-29 – нули, определяющие команды основного режима адресации. Разряды 22 и 21 определяют поле основного режима адресации (G), которое определяет назначение разрядов с 15 по 0 для адресации операнда src.

Возможные состояния разрядов 22 и 21 (поле G) следующие:

0 0 – регистровая (все регистры ЦПУ, если не определены иначе);

0 1 – прямая;

1 0 – косвенная;

1 1 – непосредственная.

Если поля src и dst соответствуют спецификациям регистра, то значения в этих полях соответствуют адресам регистра ЦПУ как определено таблицей 5.1. Для режимов основной адресации допустимы следующие значения ARn: (ARn, $0 \le n \le 7$).

31 29	28 23	22	21	20 16	15 14 13 12 11 10 9 8 7 6 5	4 0
0 0 0	операция	0	0	dst	0 0 0 0 0 0 0 0 0 0 0	src
0 0 0	операция	0	1	dst	прямая	
0 0 0	операция	1	0	dst	modn ARn	disp
0 0 0	операция	1	1	dst	непосредственная	
		G	G	Назначение	Исходные операнды	

Рисунок 5.20 – Основные режимы адресации

Таблица 5.2 – Кодирование для основных режимов адресации

Поле модификации	Синтаксис	Операция	Описание
Косвенная адресация с заменой			
00000	*+ ARn(disp)	addr = ARn + disp	Сложение с предварительным смещением
00001	*- ARn(disp)	addr = ARn - disp	Вычитание с предварительным смещением
00010	* ++ ARn(disp)	addr = ARn + disp ARn = ARn + disp	Сложение и модификация с предварительным смещением
00011	* -- ARn(disp)	addr = ARn - disp ARn = ARn - disp	Вычитание и модификация с предварительным смещением
00100	* ARn ++(disp)	addr = ARn ARn = ARn + disp	Сложение и модификация с последующим смещением
00101	* ARn -- (disp)	addr = ARn ARn = ARn - disp	Вычитание и модификация с последующим смещением
00110	* ARn ++ disp)%	addr = ARn ARn= circ(ARn+disp)	Сложение и циклическая модификация с последующим смещением
00111	* ARn -- (disp)%	addr = ARn ARn =circ(ARn-disp)	Вычитание и циклическая модификация с последующим смещением
Косвенная адресация с индексным регистром (IR0)			
01000	* + ARn(IR0)	addr = ARn + IR0	Сложение с предварительным индексированием (IR0)
01001	* - ARn(IR0)	addr = ARn + IR0	Вычитание с предварительным индексированием (IR0)
01010	* ++ ARn(IR0)	addr = ARn + IR0 ARn =ARn + IR0	Сложение и модификация с предварительным индексированием (IR0)
01011	*-- ARn(IR0)	addr = Arn – IR0 Arn = Arn – IR0	Вычитание и модификация с предварительным индексированием (IR0)
01100	* ARn ++ (IR0)	addr = ARn ARn = ARn + IR0	Сложение и модификация с последующим индексированием (IR0)
01101	* ARn -- (IR0)	addr = ARn ARn = ARn - IR0	Вычитание и модификация с последующим индексированием (IR0)
01110	* ARn ++ (IR0)%	addr = ARn ARn = irc(ARn+IR0)	Сложение и циклическая модификация с последующим индексированием
01111	* ARn -- (IR0)%	addr = ARn ARn = circ(ARn-IR0)	Вычитание и циклическая модификация с последующим индексированием
Косвенная адресация с индексным регистром (IR1)			
10000	* + ARn(IR1)	addr = ARn + IR1	Сложение с предварительным индексированием (IR1)
10001	* - ARn(IR1)	addr = ARn - IR1	Вычитание с предварительным индексированием (IR1)
10010	* ++ ARn(IR1)	addr = ARn + IR1 ARn = ARn + IR1	Вычитание и модификация с предварительным индексированием (IR1)
10011	* -- ARn(IR1)	addr = ARn - IR1 ARn = ARn - IR1	Вычитание и модификация с предварительным индексированием (IR1)
10100	* ARn ++ (IR1)	addr = ARn ARn = ARn + IR1	Сложение и модификация с последующим индексированием (IR1)
10101	* ARn -- (IR1)	addr = ARn ARn = ARn - IR1	Вычитание и модификация с последующим индексированием (IR1)
10110	* ARn ++ (IR1)%	addr = ARn ARn = irc(ARn+IR1)	Сложение и циклическая модификация с последующим индексированием (IR1)
10111	* ARn -- (IR1)%	addr = ARn ARn = circ(ARn-IR1)	Вычитание и циклическая модификация с последующим индексированием (IR1)
Косвенная адресация (специальные случаи)			
11000	* ARn	addr = ARn	Косвенная
11001	* ARn ++ (IR0)B	addr = ARn ARn = B(ARn + IR0)	Сложение и битреверсная модификация с последующим индексированием (IR0)

5.2.2 Трехоперандные режимы адресации

Команды, использующие трехоперандные режимы адресации, такие как ADDI3, LSH3, CMPF3 или XOR3 обычно представляются в форме:

SRC1 «операция» SRC2 → dst,

где dst – операнд назначения, SRC1 и SRC2 – операнды источника, «операция» определяет выполняемую операцию.

Примечание – Цифра «3» может быть опущена в трехоперандных командах.

В разрядах 31-29 установлено значение 001, обозначающее команды трехоперандного режима адресации. Разряды 22 и 21 определяют поле (T) трехоперандного режима адресации, которое определяет как интерпретируются разряды 15-0 для адресации операндов SRC. Разряды 15-8 используются для определения адреса SRC1 и разряды 7-0 для определения адреса SRC2.

Варианты для разрядов 22 и 21 следующие:

T (22 21)	SRC1	SRC2
0 0	Регистровый	Регистровый
0 1	Косвенный	Регистровый
1 0	Регистровый	Косвенный
1 1	Косвенный	Косвенный

Если поля SRC1 и SRC2 используют одинаковые вспомогательные регистры, то оба адреса сгенерированы правильно. Однако, только значение, созданное полем SRC1, сохраняется в определенном вспомогательном регистре. Ассемблер выдает предупреждение, если это условие определено пользователем.

Допустимы следующие значения ARn и ARm:

ARn, $0 \le n \le 7$

ARm, $0 \le m \le 7$

Сокращения «modm» или «modn» обозначают поле модификации, соответствующее ARn или ARm. В таблице 5.2 приведена полная информация.

В косвенной адресации трехоперандного режима адресации, допустимо смещение 0 или 1 (если используется смещение) и могут быть использованы индексные регистры (IR0 и IR1). Смещение 1 – неявное и не закодировано явно в командном слове.

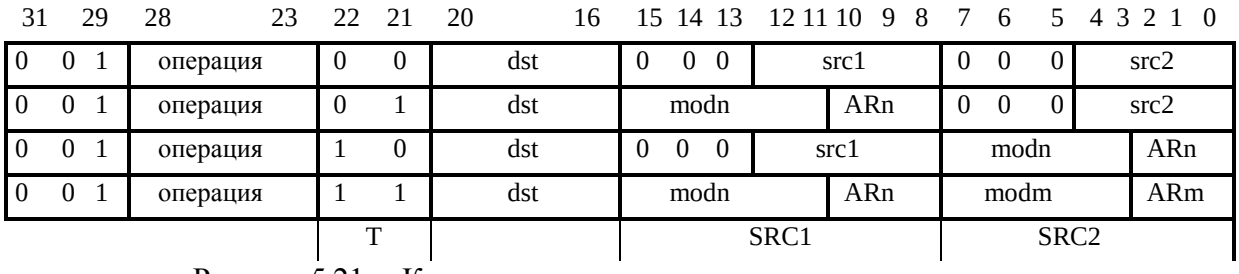


Рисунок 5.21 – Кодирование для трехоперандных режимов адресации

5.2.3 Параллельные режимы адресации

Команды, использующие параллельную адресацию (обозначены || (две вертикальные линии)), обеспечивают максимально возможный параллелизм. Операнды назначения обозначены d1 и d2, обозначающие dst1 и dst2, соответственно. Операнды источника, обозначенные

src1 и src2, используют регистры повышенной точности. Выполняемые параллельные операции обозначены как «операция».

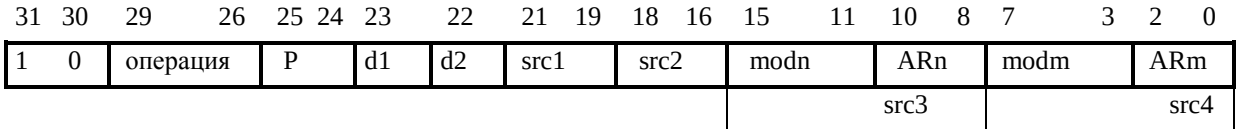


Рисунок 5.22 – Кодирование для параллельных режимов адресации

Поле параллельного режима адресации (P) определено по использованию операндов, т. е. являются они операндами источника или назначения. Связь между полем P и операндами подробно представлена в описании отдельных параллельных команд (см. раздел 10 «Команды языка ассемблера»). Однако операнды всегда кодируются одинаково. В разрядах 31 и 30 установлено значение 10, обозначающее команды параллельного режима адресации. Разряды 21-0 определяют поле параллельного режима адресации (P), которое определяет как разряды 21-0 интерпретируются для адресации операндов src. Разряды 21-19 используются для определения адреса src1, разряды 18-16 определяют адрес src2, разряды 15-8 – адрес src3 и разряды 7-0 – адрес src4. Запись «modn» и «modm» обозначает, какое поле модификации соответствует какому ARn или ARm (вспомогательный регистр) полю.

Операнды параллельной адресации приведены ниже:

src1, $0 \leq$ src1 ≤ 7 (регистры повышенной точности R0-R7);

src2, $0 \leq$ src2 ≤ 7 (регистры повышенной точности R0-R7);

d1, если 0, dst1 это R0; если 1, dst1 это R1;

d2, если 0, dst2 это R2; если 1, dst2 это R3;

P, $0 \leq P \leq 3$;

src3 – косвенная (disp = 0, 1, IR0, IR1);

src4 – косвенная (disp = 0, 1, IR0, IR1).

Как в трехоперандном режиме адресации, косвенная адресация в параллельном режиме адресации допустима для смещения 1 или 0 и при использовании индексных регистров (IR0 и IR1). Замена 1 неявная и не записывается явно в командном слове.

Если поля src1 и src2 используют одинаковые вспомогательные регистры, то оба адреса сгенерированы правильно, но только значение, созданное полем src3, хранится в определенном вспомогательном регистре. Ассемблер выдает предупреждение, если это условие определено пользователем.

5.2.4 Режим длинной непосредственной адресации

Режим длинной непосредственной адресации, см. рисунок 5.23, используется для кодирования инструкций управления программой (BR, BRd и CALL), для которых полезно иметь 24-разрядный адрес, содержащийся в командном слове. Безусловные переходы BR (стандартный) и BRD (задержанный) используют режим длинной непосредственной адресации. В разрядах 31-26 установлено значение 011000, обозначающее команды режима длинной непосредственной адресации. Выбор 24 разряда определяет способ ветвления: D=0 для стандартного перехода или D=1 для задержанного перехода.

Длинный непосредственный операнд – это 24-разрядный src.

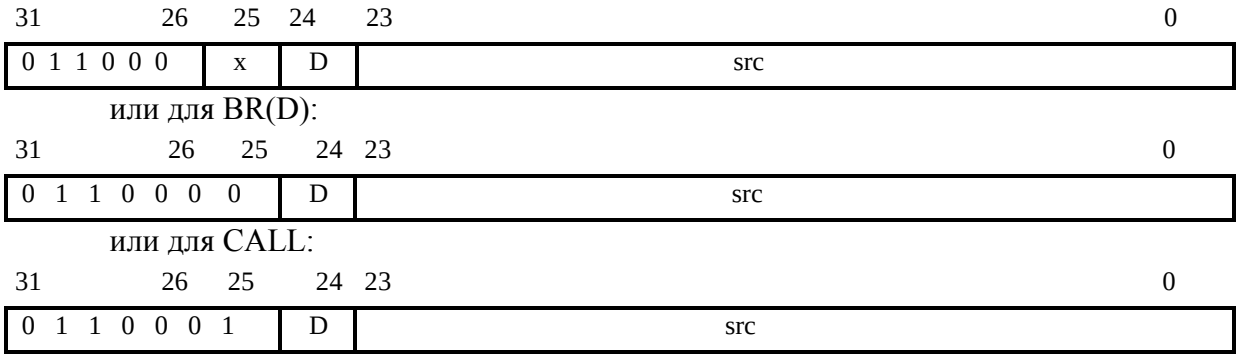


Рисунок 5.23 – Кодирование для режима длинной непосредственной адресации

5.2.5 Режимы адресации условных переходов

Команды, использующие режимы адресации условных переходов (Bcond, BcondD, CALLcond, DBcond и DBcondD), могут выполнять различные условные операции. В разрядах 31-27 установлено значение 01101, обозначающее команды режима адресации условных переходов; разряд 26 установлен в 0 или 1, первый выбирает DBcond, другой – Bcond. Выбор 25 разряда определяет режим адресации условных переходов (B).

На рисунке 5.24 показано кодирование для режима адресации условных переходов.

DBcond(D):

31 26	25	24 22	21	20 16	15 5	4 0
0 1 1 0 1 1	B	ARn	D	cond	0 0 0 0 0 0 0 0 0 0 0	src reg
0 1 1 0 1 1	B	ARn	D	cond	непосредственная (относительно PC)	

Bcond(D):

31 26	25	24 22	21	20 16	15 5	4 0
0 1 1 0 1 0	B	0 0 0	D	cond	0 0 0 0 0 0 0 0 0 0 0	src reg
0 1 1 0 1 0	B	0 0 0	D	cond	непосредственная (относительно PC)	

CALLcond(D):

31 26	25	24 22	21	2016	15 5	4 0
0 1 1 1 0 0	B	0 0 0	0	cond	0 0 0 0 0 0 0 0 0 0 0	src reg
0 1 1 1 0 0	B	0 0 0	0	cond	непосредственная (относительно PC)	

Рисунок 5.24 – Кодирование для режимов адресации условных переходов

Если B=0, то используется регистровая адресация, если B=1, то используется PC – относительная адресация. Выбор 21 разряда устанавливает способ ветвления: D=0 для стандартного перехода или D=1 для задержанного перехода. Поле условия (cond) определяет условие, которое проверяется для определения того, какое действие необходимо выполнить, т. е. должно ли происходить ветвление.

5.3 Циклическая адресация

Многие алгоритмы, такие как свертка и корреляция, требуют организации циклического буфера в памяти. В процессе свертки и корреляции циклический буфер используется для реализации скользящего окна, которое содержит самые последние обрабатываемые значения. По мере того, как вводятся новые данные, они стирают предыдущую информацию.

Ключом для реализации циклического буфера является реализация циклического режима адресации. В этом подразделе описан циклический режим адресации ПЦОС.

Регистр размера блока (BK) определяет размер циклического буфера. Дно циклического буфера определяется как первый разряд (считая от старшего значащего разряда к младшему) в младших 16 разрядах регистра BK плюс выбираемый пользователем вспомогательный регистр (ARn). Если размещение первого единичного разряда определено как разряд N, адрес вершины буфера именуется действительным основанием (EB) и соответствует разрядам с 31 до (N+1) регистра ARn с нулевыми разрядами с N до 0 EB.

Циклический буфер размером R должен начинаться на границе K разряда (т. е. K младших значащих разрядов циклического буфера должны быть 0), где K – наименьшее целое, удовлетворяющее соотношению $2 > R$. Значение R также должно быть загружено в регистр BK. Например, 31-словный (31-word) циклический буфер должен начинаться с адреса, чьи 5 младших значащих разрядов = 0 (т. е. XXXXXXXXXXXXXXXXXXXXXXXXX00000) и значение 31 должно быть загружено в регистр BK.

В циклической адресации «index» относится к N младшим значащим разрядам выбранного вспомогательного регистра, и «step» – это величина, которая будет добавлена к вспомогательному регистру или вычтена из него. При использовании циклической адресации необходимо придерживаться следующих двух правил:

- используемый шаг должен быть меньше или равен размеру блока;
- сначала адресуется циклическая очередь, вспомогательный регистр должен указывать на элемент в циклической очереди.

Алгоритм для циклической адресации следующий:

Если $0 \leq$ index + step < BK:

index = index + step.

Если index + step $\geq$ BK:

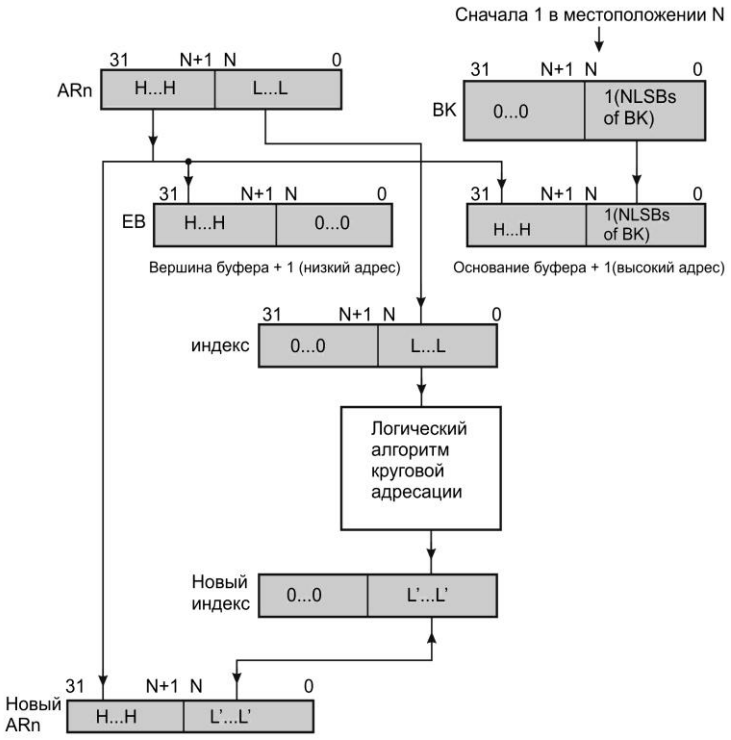
index = index + step - BK.

Если index + step < 0:

index = index + step + BK.

Циклическая адресация особенно полезна для реализации КИХ-фильтров (фильтров с конечной импульсной характеристикой).

Блок-схема циклической адресации представлена на рисунке 5.25.



Обозначения:

ARn – вспомогательный регистр n;

BK – регистр размера блока;

EB – действительное основание;

H – старшие разряды;

L – младшие разряды;

L` – новые младшие разряды;

LSB – младший значащий разряд;

N – значение разряда.

Рисунок 5.25 – Блок-схема циклической адресации

На рисунке 5.26 показана реализация циклического буфера. Это иллюстрирует взаимосвязь сгенерированных величин и элементов в циклическом буфере.

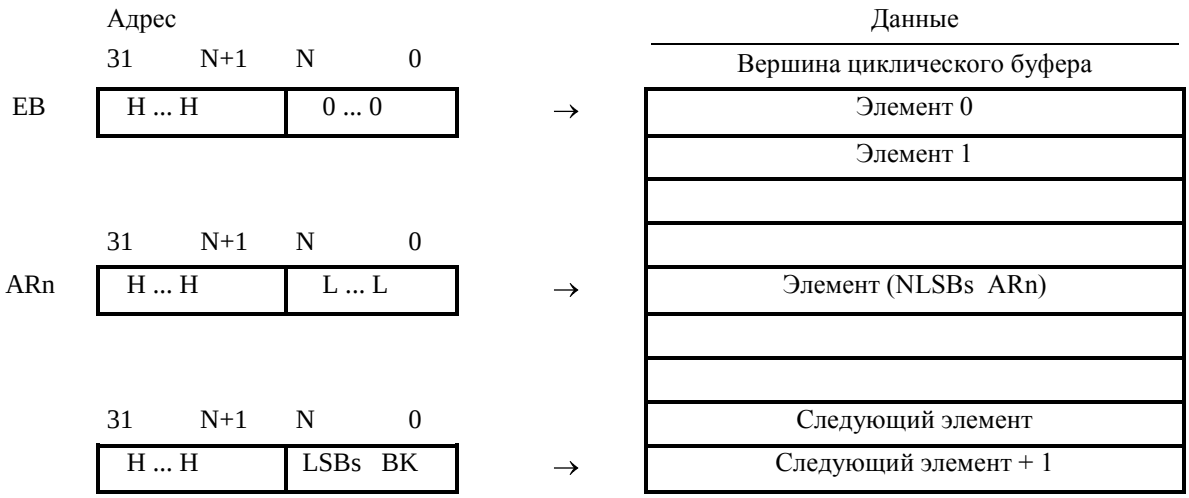


Рисунок 5.26 – Реализация циклического буфера

На рисунке 5.27 приведен пример операции при циклической адресации. Предполагается, что все AR по четыре разряда, AR0=0000 и BK=0110 (размер блока = 6). Этот пример показывает последовательность модификаций и результирующее значение AR0. Он также

иллюстрирует то, как указатель продвигается по очереди с разным шагом (как с увеличением, так и с уменьшением).

```
*AR0 ++ (5) %    ;  AR0 = 0 (нулевое значение)
*AR0 ++ (2) %    ;  AR0 = 5 (первое значение)
*AR0 -- (3) %    ;  AR0 = 1 (второе значение)
*AR0++ (6) %     ;  AR0 = 4 (третье значение)
*AR0-- %         ;  AR0 = 4 (четвертое значение)
*AR0             ;  AR0 = 3 (пятое значение)
```

Рисунок 5.27 – Пример циклической адресации

5.4 Битреверсная адресация

Битреверсная адресация в ПЦОС улучшает характеристики скорости выполнения и использования памяти программ для алгоритмов БПФ, которые используют различные основания. Базовый адрес битреверсной адресации должен быть размещен на границе размера таблицы. Например, если IR0 = 2^{n-1}, n младших разрядов базового адреса должны быть равны нулю. Базовый адрес данных в памяти должен быть на границе 2^n. Один вспомогательный регистр указывает на физический адрес значения данных. IR0 определяет половину размера БПФ; например, значение, содержащееся в IR0 должно быть равно 2^{n-1}, где n – целое и размер БПФ = 2^n. При добавлении содержимого IR0 к вспомогательному регистру, используя битреверсную адресацию, генерируется адрес в битреверсной форме.

Рисунок 5.28 показывает последовательность модификаций AR2 и результирующие значения AR2.

```
* AR2++(IR0)B    ; AR2 = 0110 0000 (нулевое значение)
 * AR2++(IR0)B   ; AR2 = 0110 1000 (первое значение)
 * AR2++(IR0)B   ; AR2 = 0110 0100 (второе значение)
 * AR2++(IR0)B   ; AR2 = 0110 1100 (третье значение)
 * AR2++(IR0)B   ; AR2 = 0110 0010 (четвертое значение)
 * AR2++(IR0)B   ; AR2 = 0110 1010 (пятое значение)
 * AR2++(IR0)B   ; AR2 = 0110 0110 (шестое значение)
 * AR2           ; AR2 = 0110 1110 (седьмое значение)
```

Рисунок 5.28 – Пример битреверсной адресации

Чтобы проиллюстрировать этот вид адресации, были взяты восьмиразрядные вспомогательные регистры. Например, AR2 содержит значение 0110 0000 (96). Это базовый адрес данных в памяти. Регистр IR0 содержит значение 0000 1000 (8).

В таблице 5.3 приведены отношения шагов индекса и четырех младших разрядов AR2.

Как видно, можно найти четыре младших значащих разряда реверсированием набора разрядов шагов.

Таблица 5.3 – Шаг индекса и битреверсная адресация

Шаг	Набор разрядов	Битреверсный набор	Битреверсный шаг
0	0000	0000	0
1	0001	1000	8
2	0010	0100	4
3	0011	1100	12
4	0100	0010	2
5	0101	1010	10
6	0110	0110	6
7	0111	1110	14
8	1000	0001	1
9	1001	1001	9
10	1010	0101	5
11	1011	1101	13
12	1100	0011	3
13	1101	1011	11
14	1110	0111	7
15	1111	1111	15

5.5 Управление системным стеком и стеком пользователя

ПЦОС обеспечивает специальный указатель системного стека (SP) для построения стеков в памяти. Вспомогательный регистр также может быть использован для построения множества общих линейных списков. Далее обсуждается реализация следующих способов линейных списков:

- стек – линейный список, для которого все вставки и удаления выполняются в одном конце списка;

- очередь – линейный список, для которого все вставки выполняются в одном конце списка и все удаления выполняются в другом конце списка;

- удаление – линейный список очереди с двумя концами, для которого вставки и удаления могут выполняться в любом из концов списка;

- указатель системного стека (SP) – это 32-разрядный регистр, который содержит адрес вершины системного стека. Системный стек заполняется от младших адресов памяти к старшим. SP всегда указывает на следующий элемент, подталкиваемый в стек.

При проталкивании данных в стек выполняется прединкремент, а при выталкивании – постдекремент указателя системного стека.

Счетчик команд проталкивает данные в системный стек во время вызовов подпрограмм, прерываний и системных прерываний. Стек может выполнять проталкивание и выталкивание данных, используя команды PUSH, POP, PUSHF и POPF.

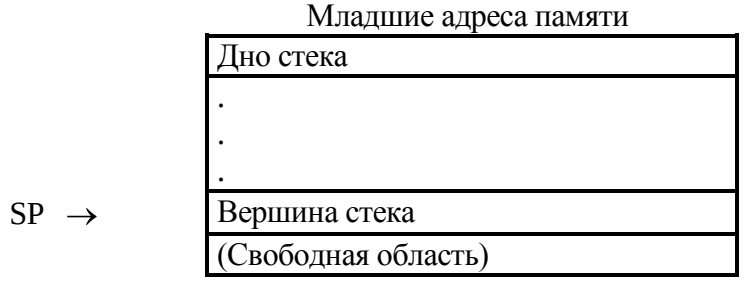


Рисунок 5.29 – Конфигурация системного стека

5.5.1 Стеки

Стеки могут быть построены от младших адресов памяти к старшим или от старших к младшим. Стеки могут быть построены, используя режимы прединкремента/декремента и постинкремента/декремента изменения вспомогательных регистров (AR). Изменение стека от старших адресов памяти к младшим может быть выполнено двумя путями:

- 1 вариант: записывает данные в память, используя ***--**ARn для проталкивания данных в стек, и считывает из памяти, используя ***ARn ++** для выталкивания данных из стека;

- 2 вариант: записывает данные в память, используя ***ARn --** для проталкивания данных в стек, и считывает из памяти, используя ***++** ARn для выталкивания данных из стека.

Рисунок 5.30 иллюстрирует два этих случая. Различие только в том, что при использовании варианта 1, AR всегда указывает на вершину стека, а в варианте 2 – AR всегда указывает на следующую свободную область в стеке.

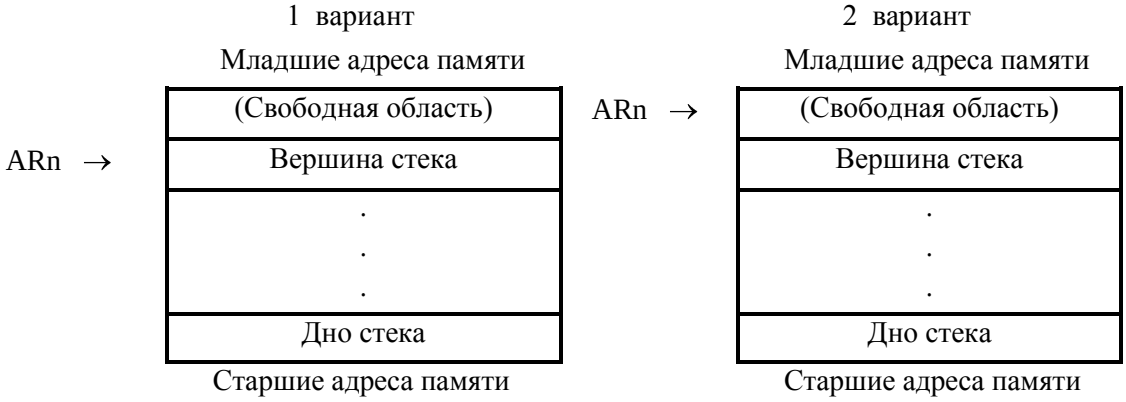


Рисунок 5.30 – Два варианта реализации стека

Изменение стека от младших адресов памяти к старшим может быть реализовано двумя путями:

- 3 вариант: записывает данные в память, используя ***++** ARn для проталкивания данных в стек, и считывает из памяти, используя *ARn -- для выталкивания данных из стека;

- 4 вариант: записывает данные в память, используя *ARn ++ для проталкивания данных в стек, и считывает из памяти, используя ***** -- ARn для выталкивания данных из стека.

На рисунке 5.31 показаны эти два случая.

Случай 3: AR всегда указывает на вершину стека.

Случай 4: AR всегда указывает на следующую свободную область в стеке.

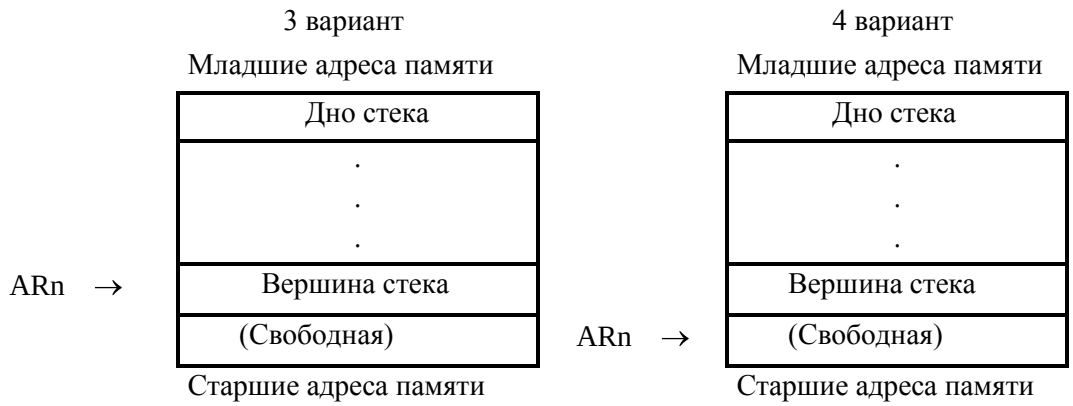


Рисунок 5.31 – Реализация стеков от младших адресов памяти к старшим

5.5.2 Очереди

Очередь реализована как FIFO. Реализация очередей основана на манипуляции вспомогательными регистрами. Для очередей используются два вспомогательных регистра, один – для обозначения передней стороны очереди, с которой данные выталкиваются из стека (или выводятся из очереди), другой – для обозначения обратной стороны очереди, где данные проталкиваются в стек. При соответствующем управлении вспомогательными регистрами, очередь может быть циклической. Очередь является циклической, если обратный указатель применяется для указания на начало памяти очереди после того, как он указывал на конец памяти очереди.

6 Управление выполнением программы

ПЦОС обеспечивает полное множество гибких и мощных конструкций, которые обеспечивают управление программным и аппаратным обеспечением выполнения программы. Программное управление обеспечивает повторения, ветвления, вызовы, системные прерывания и возвраты. Аппаратное управление обеспечивает выполнение операции, сброс и прерывания. Т. к. программирование включает множество различных конструкций, можно выбрать одну, наиболее подходящую для конкретного случая.

Несколько команд блокировки обеспечивают гибкую поддержку мультипроцессорного режима работы через использование внешних сигналов, являющихся мощным средством синхронизации. Они также гарантируют целостность связи и результата в высокоскоростных операциях.

ПЦОС поддерживает сигнал немаскируемого внешнего сброса и некоторое количество внутренних и внешних прерываний. Эти функции могут быть запрограммированы для специального применения.

В данном разделе рассматриваются следующие основные темы:

- режимы повторений (подраздел 6.1);
- инициализация (операции) (6.1.1 – 6.1.3);
- задержанные переходы (подраздел 6.2);
- вызовы, системные прерывания и возвраты (подраздел 6.3);
- операции блокировки (подраздел 6.4);
- операция сброса (подраздел 6.5);
- прерывания (подраздел 6.6).

6.1 Режимы повторений

Режимы повторений ПЦОС могут применяться для реализации циклов без дополнительных потерь. Для многих алгоритмов наибольшее время тратится на реализацию внутреннего ядра программы. Использование режимов повторений позволяет максимально сократить время выполнения критичных по времени секций программы.

ПЦОС обеспечивает две команды для реализации циклов без дополнительных потерь:

- RPTB – повторение блока кода;
- RPTS – повторение одной команды.

RPTB позволяет выполнять блок кода определенное количество раз.

RPTS обеспечивает повторение одной команды определенное количество раз и уменьшает нагрузку шины посредством выбора команды только один раз.

Три регистра (RS, RE и RC) связаны с установкой счетчика программ, если установка происходит в режиме повторения. В таблице 6.1 описаны эти регистры.

Таблица 6.1 – Регистры режима повторения

Регистр	Функция
Регистр начального адреса повторений RS	Содержит адрес первой команды повторяемого блока команд.
Регистр конечного адреса повторений RE	Содержит адрес последней команды повторяемого блока кода.
Регистр счета повторений RC	Содержит значение, на 1 меньшее числа повторений блока кода. Например, для выполнения блока N раз, необходимо загрузить N-1 в RC.

6.1.1 Инициализация режима повторений

Существует два разряда, которые очень важны для операций RPTB и RPTS, это разряды RM и S.

RM (флаг режима повторений) – разряд в регистре состояния, который определяет, работает ли процессор в режиме повторений. Если RM=0, выборка в режиме повторений не производится, если RM=1, выборка в режиме повторений производится.

Разряд S скрыт от пользователя, но он необходим для полного описания операций RPTB и RPTS. Если S=0, то ЦПУ не выполняет выборки в режиме одного повторения. Если S=1 и RM=1, то ЦПУ выполняет выборки в режиме одного повторения.

Для корректной работы режимов повторений требуется, чтобы все вышеописанные регистры и поля регистра состояния были правильно инициализированы.

Команды RPTB и RPTS выполняют эту инициализацию несколько различно (см. 6.1.2 и 6.1.3).

6.1.2 Инициализация RPTB

Когда выполняется RPTB src, выполняются следующие операции

Операция 1: PC + 1 → RS

Операция 2: src → RE

Операция 3: 1 → Разряд RM регистра состояния

Операция 4: 0 → S разряд.

Операция 1 загружает начальный адрес блока в RS. Операция 2 загружает src в RE (конечный адрес блока). Операнд src – это 24-разрядная величина, содержащаяся в командном слове. Операция 3 устанавливает регистр состояния для указания режима повторения операции. Операция 4 показывает, что это режим повторения блока.

Следующая требующаяся информация – это количество повторений блока.

Значение определяется инициализацией регистра RC (счетчик повторений). Так как выполнение RPTB не загружает RC, этот регистр должен загружаться пользователем явно. Типичная установка операции повторения блока показана ниже:

```
LDI   15, RC   ;  15 → RC
RPTB  LOOP ;  LOOP → RE, PC + 1 → RS, 1 → RM, 0 → S
```

Режимы повторений повторяют блок кодов команд, по крайней мере, один раз в обычной операции. Счетчик операций будет загружаться числом, на 1 меньшим, чем число повторений блока, так что значение 0 в RC повторяет блок кода один раз. Все повторения блоков, начатые RPTB, могут быть прерваны.

6.1.3 Инициализация RPTS

Во время выполнения RPTS src производится следующая последовательность операций.

Операция 1: PC + 1 → RS

Операция 2: PC + 1 → RE

Операция 3: 1 → Разряд RM регистра состояния

Операция 4: 1 → Бит S

Операция 5: src → RC

Команда RPTS загружает все регистры и разряды режима, необходимые для операции повторения одной команды. Операция 1 загружает начальный адрес блока в RS. Операция 2 загружает конечный адрес в RE (конечный адрес блока).

Так как это повторение одной команды, то начальный и конечный адреса совпадают. Операция 3 устанавливает регистр состояния для указания режима повторения операции. Операция 4 указывает, что это режим повторения одной команды.

Операнд src загружается в RC (операция 5), и следующая команда выполняется src+1 раз.

Повторения одной команды, указанной RPTS, не являются прерываемыми, так как RPTS выбирает командное слово только один раз и хранит его в регистре команд для многократного использования. Прерывание может привести к потере командного слова. Повторная выборка командного слова из регистра команд уменьшает количество обращений к памяти и, в результате, действует как программный кэш на одно слово. Если необходимо иметь одну команду, которую можно и повторять, и прерывать, то можно использовать команду RPTB.

6.1.4 Работа режима повторения

Информация в регистрах режима повторения и соответствующих разрядах управления используется для контроля изменений PC, когда производится выборка в режиме повторения. Режимы повторения сравнивают содержимое регистра RE с программным счетчиком (PC). Если они соответствуют друг другу и счетчик повторений неотрицательный, то счетчик повторений уменьшается, PC загружается с начального адреса повторений и обработка продолжается. Выборка и соответствующие разряды состояния при необходимости изменяются.

Примечание – Счетчик повторений (RC) никогда не изменяется, если RM равно нулю. Число повторений максимально, если RC=080000000h, что соответствует числу повторений 080000001h.

Подробно алгоритм изменения PC показан в примере 6.1.

Пример 6.1 – Алгоритм управления режимом повторения

```
if RM=1                    ; Если в режиме повторения (RPTB или RPTS)
if S=1                     ; Если RPTS
if первый раз              ; Если это первая выборка
выбор команды из памяти    ; Выборка команды из памяти
else                       ; Если не первая выборка
выбор команды из IR        ; Выборка команды из IR
RC-1 → RC                  ; Уменьшение RC
if  RC < 0                 ; Если RC отрицательный
                           ; Завершение режима повторений одной команды
0 → ST(RM)                 ; Выключить разряд режима повторений
0 → S                      ; Очистить S
```

```
PC+1 → PC                ; Увеличение PC
else if S==0             ; Если RPTB
выбор команды из памяти  ; Выборка команды из памяти
if PC=RE                 ; Если это конец блока
RC - 1 → RC              ; Уменьшение RC
if RC ≥ 0                ; Если RC не отрицательный
RS → PC                  ; Устанавливает PC в начало блока
else if RC < 0           ; Если RC отрицательно
0 → ST(RM)               ; Выключить разряды режима повторений
0 → S                    ; Очистить S
PC+1 → PC                ; Увеличение PC
```

Пример 6.2 показывает типичную установку блока повторений.

Пример 6.2 – Выполнение RPTB

```
LDI      15, RC          ; Загружает 15 в счетчик повторений
PTB      ENDLOP          ; Выполняет блок кода
STLOOP                   ; От STLOOP до ENDLOP 16 раз
          .
          .
          .
ENDLOP
```

Использование этого режима изменения PC позволяет проанализировать, что произойдет в случае ветвлений внутри блока. Предполагается, что следующее значение PC будет либо PC+1, либо содержимым регистра RS. Таким образом, этот метод повторения блока допустим для любого количества ветвлений внутри блока повторений. Выполнение может идти в любом месте внутри пользовательской программы через прерывания, вызовы подпрограмм и т. д. Для надлежащей модификации счетчика цикла должна быть выбрана последняя команда цикла.

Повторение цикла может быть остановлено до завершения посредством занесения 0 в счетчик повторения или занесением 0 в разряд RM регистра состояния.

Так как режим повторения блока изменяет счетчик программ, другие команды не могут изменять счетчик программ в то же самое время.

Необходимо придерживаться двух правил:

- следующая команда в блоке (или только одна команда в блоке единичного размера) не может быть: Bcond, BR, Dbcond, CALL, CALLcond, TRAPcond, RETIcond, RETScond, IDLE, RPTB или RPTS. Пример 6.3 иллюстрирует некорректное размещение стандартного перехода;

- ни одна из следующих четырех команд не может быть в конце блока (или только одна команда в блоке единичного размера): BcondD, BRD или DBcondD.

Пример 6.4 показывает некорректное размещение задержанного ветвления.

Если какое либо из этих правил нарушено, PC будет не определен!

Пример 6.3 – Некорректное размещение стандартного перехода

```
LDI    15, RC            ; Загрузка 15 в счетчик повторений
RPTB   ENDLOP            ; Выполнение блока кода
STLOOP                   ; от STLOOP до ENDLOP 16 раз
```

```
.
.
.
ENDLOP   BR   OOPS      ; Этот переход нарушает правило 1
```

Пример 6.4 – Некорректное размещение задержанного перехода

```
LDI    15, RC              ; Загрузка счетчика повторений с 15
RPTB      ENDLOP           ; Выполнение блока кода
STLOOP                     ; от STLOOP до ENDLOP 16 раз
BRD    OOPS                ; Этот переход нарушает правило 2
ADDF
MPYF
ENDLOP   SUBF
```

Повторения блоков (RPTB) могут быть вложенными. Так как все управление определено регистрами RS, RE, RC и ST, допустимо хранение и восстановление этих регистров для обеспечения их вложенности. Разряд RM в регистре состояния может быть использован для определения активности режима повторения блока. Например, если составляется программа обслуживания прерывания, требующая использование RPTB, возможно, что прерывание, связанное с программой, может происходить во время другого повторения блока. Программа обработки прерывания может проверять разряд RM для определения того, является ли активным режим повторения. Если этот разряд установлен, программа прерывания сохраняет RS, RE, RC и ST. Затем программа прерывания может выполнять повторение блока. Перед возвратом в прерванную программу, программа прерывания восстанавливает регистры RS, RE, RC и ST. Если разряд RM не установлен, то сохранять и восстанавливать эти регистры нет необходимости.

6.2 Задержанные переходы

В ПЦОС имеются два основных типа ветвлений: стандартные переходы и задержанные переходы. Стандартные переходы освобождают конвейер до выполнения перехода, чтобы гарантировать корректное управление программным счетчиком. В результате ветвление на ПЦОС выполняется за четыре цикла. В этот класс включены вызовы, возвраты и системные прерывания.

Задержанные переходы не освобождают конвейер, но также гарантируют, что следующие три команды будут выполнены до того, как программный счетчик будет изменен ветвлением. Результатом является переход, требующий только одного цикла, таким образом, скорость работы задержанного перехода очень близка к оптимальному режиму повторения блоков на ПЦОС. Однако, в отличие от режимов повторения блоков, задержанные переходы могут быть использованы не только для организации цикла. Каждый задержанный переход имеет в соответствие стандартный переход, который применяется, когда задержанный переход не может быть использован. Задержанные переходы в ПЦОС выполняются командами BcondD, BRD и DBcondD.

Условные задержанные переходы используют условия, которые возникают в конце команды, непосредственно предшествующей задержанному переходу. Флаги условий не зависят от команд, следующих за задержанным переходом. Флаги условий устанавливаются предыдущей командой тогда, когда регистром назначения этой команды является один из регистров повышенной точности (R7-R0) или при выполнении команд сравнения (CMPF, SMPF3, CMPI, CMPI3, TSTB или TSTB3). Задержанные переходы гарантируют, что три следующие команды будут выполнены, независимо от других конфликтов конвейера.

Когда задержанные переходы выбраны, они остаются задержанными до тех пор, пока три следующие команды не будут выполнены. Ни одна из трех команд, следующих за задержанным переходом, не может быть Bcond, BcondB, BR, BRD, DBcond, DBcondB, CALL, CALLcond, TRAPcond, RETIcond, RETScond, RPTB, RPTS или IDLE (см. пример 6.5).

Задержанные переходы запрещают прерывания до тех пор, пока три команды, следующие за задержанным переходом, не будут выполнены. Это не зависит от того, произошло ли ветвление.

При некорректном использовании задержанных переходов PC будет не определен!

Пример 6.5 – Некорректное размещение задержанного перехода

```
B1:   BD    L1
      NOP
      NOP
B2:   B     L2    ; Этот переход размещен некорректно
      NOP
      NOP
      NOP
```

6.3 Вызовы, системные прерывания и возвраты

Вызовы и системные прерывания обеспечивают выполнение подпрограммы или функции до возврата в вызывавшую программу.

Команды CALL, CALLcond и TRAPcond сохраняют значение PC в стеке до изменения содержимого PC, таким образом, стек обеспечивает возврат при использовании команд RETScond или RETIcond.

Команда CALL помещает значение следующего PC в стек и помещает операнд src в PC, src – 24-разрядное непосредственное значение. Команда CALLcond подобна команде CALL, за исключением того, что:

- она выполняется тогда, когда определенное условие есть «истина» (20 условий – включая безусловные – приведены в подразделе 10.2);
- src является либо относительным (относительно PC) смещением, либо определено регистровым режимом адресации.

Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7-R0), либо когда выполняется одна из команд сравнения CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3.

Команда TRAPcond также выполняется тогда, когда определенное условие есть «истина» (условия те же, что и для CALLcond). При выполнении команды прерывания запрещены записью 0 в разряд GIE ST, следующее значение PC сохраняется в стеке, и вектор выбирается одним из адресов 20h - 3Fh и загружается в PC.

Соответствующий адрес определяется номером системного прерывания в команде TRAP. Использование RETIcond для возврата опять разрешает прерывания.

RETScond возвращает выполнение от одной из вышеперечисленных команд, загружая вершину стека в PC. Для выполнения команды необходимо, чтобы определенное условие было «истина». Условия те же, что и для CALLcond.

RETIcond возвращает из системного прерывания или вызова аналогично команде RETScond и, кроме того, устанавливает бит GIE в регистре состояния, таким образом, разрешая прерывания, разряды, разрешения которых установлены в 1. Условия те же, что и для CALLcond.

На самом деле, вызовы и системные прерывания завершаются практически одинаково (т. е. подфункция вызывается, выполняется, затем управление возвращается к вызывающей функции). Системные прерывания имеют следующие преимущества:

- прерывания автоматически запрещаются при выполнении системного прерывания. Это позволяет выполнять критичную программу без риска прерываний, поэтому для разрешения прерываний используется команда RETIcond;

- можно использовать системные прерывания для косвенного вызова функций. Это практически незаменимо, когда ядро программы содержит основные подфункции для использования в приложениях. В этом случае функции в ядре могут быть изменены и переадресованы без перекомпиляции каждой прикладной программы.

6.4 Операции блокировки

Одна из наиболее общих мультипроцессорных конфигураций – разделение основной памяти многими процессорами. Для того чтобы разрешить многим процессорам доступ к основной памяти и совместное использование данных последовательным методом, необходимы некоторые виды арбитража или квитирования установления связи. Обеспечению требований по арбитражу служат операции блокировки ПЦОС.

ПЦОС обеспечивает гибкие средства поддержки мультипроцессорной конфигурации с помощью пяти команд, относящихся к операциям блокировки. Используя внешние сигналы, эти команды обеспечивают механизм общей синхронизации. Они также гарантируют целостность связей и результата при высокоскоростных операциях.

Операции блокировки используют два внешних вывода флага – XF0 и XF1.

XF0 должен быть определен как выход и XF1 – как вход. Когда флаги представлены этим методом, XF0 сигнализирует запрос операции блокировки, а XF1 действует как сигнал подтверждения для запрошенной операции блокировки. В этом режиме XF1 и XF0 имеют активный низкий уровень.

Внешняя синхронизация для блокированных загрузок и записей в памяти такая же, как стандартная загрузка и запись в памяти. Блокированные загрузки и записи в памяти могут

быть продлены подобно стандартным доступам с использованием соответствующего сигнала готовности (RDY# int или XRDY# int).

Команды LDFI и LDII выполняют следующие действия:

- одновременная установка XF0 в 0 и начало цикла чтения. Временная диаграмма XF0 подобна адресной шине во время выполнения цикла чтения;

- выполняется команда LDF или LDI и продляется цикл считывания до тех пор, пока XF1 не станет нулем и будет выдан сигнал готовности (RDY# int или XRDY# int);

- снимается установка XF0 в 0 и завершается цикл чтения.

Операция чтение/запись идентична любому другому циклу чтение/запись, исключая специальное использование XF0 и XF1. Операнд src для LDFI и LDII всегда является прямым или косвенным адресом памяти. XF0 устанавливается в 0, если src расположен вне кристалла (т. е. STRB#, MSTRB# или IOSTRB# активны) или src – одно из внутрикристальных периферийных устройств. При обращении к внутрикристальной памяти XF0 не установлен, и выполнение операции аналогично выполнению LDF или LDI во внутренней памяти.

Команды STFI и STII выполняют следующие операции:

- одновременно устанавливают XF0 в 1 и запускают цикл записи. Временная диаграмма XF0 подобна диаграмме адресной шины во время выполнения цикла записи;

- выполняются команды STF и STI и цикл записи продляется до тех пор, пока не будет сигнализирована готовность (RDY# int или XRDY# int).

Как в случае для LDFI и LDII, dst операций STFI и STII влияет на XF0. Если dst размещен вне кристалла (STRB#, MSTRB# или IOSTRB# активны) или dst – это одна из периферий на кристалле, XF0 устанавливается в 1. При обращении к памяти на кристалле XF0 не установлен, и выполнение операции аналогично выполнению STF или STI во внутренней памяти.

Функции команды SIGI следующие:

- устанавливает XF0 в 0;

- удерживается до тех пор, пока XF1 не установится в 0;

- устанавливает XF0 в 1 и завершает операцию.

Несмотря на то, что команды LDFI, LDII и SIGI ожидают установки XF1, они могут быть прерваны. LDFI и LDII для прерывания требуют сигнала готовности (RDY# int и XRDY# int). Так как прерывание имеет место на границах цикла шины (см. подраздел 6.6 «Прерывания»), прерывание может иметь место некоторое время спустя установки разрешения. Это позволяет применять механизм защиты от тупиковых условий, используя прерывание блокированной загрузки, которая продолжается слишком долго. На выходе из прерывания выполняется следующая команда. Команды STFI и STII не могут быть прерваны. Так как команды STFI и STII завершаются при сигнализации готовности, задержка до возникновения прерывания может быть такой же, как и для любых других команд.

Операции блокировки могут быть использованы для выполнения ждущего цикла (цикла ожидания сигнала занятости), для управления мультипроцессорным счетчиком, для реализации простого механизма семафора или для обеспечения синхронизации двух ПЦОС. Следующие примеры иллюстрируют применение команд операций блокировки.

Пример 6.6 показывает реализацию ждущего цикла. Если размещение LOCK является блокировкой для критической секции программы и ненулевое состояние означает, что блокировка занята, то может быть использован алгоритм для ждущего цикла.

Пример 6.6 – Ждущий цикл

```
        LDI   1,R0              ; Заносит 1 в R0
L1:     LDII  @LOCK, R1         ; Начало операции блокировки
                                ; Содержимое LOCK → R1
        STII  R0, @LOCK         ; Заносит R0 (=1) в LOCK, XF0=1
                                ; Конец операции блокировки
        BNZ   L1                ; Сохраняет пока LOCK=0
```

Пример 6.7 показывает, как положение COUNT может содержать значение счетчика количество специальных операций, которые необходимо выполнить. Эти операции могут быть выполнены любым процессором в системе. Если значение счетчика равно нулю, процессор может ожидать до тех пор, пока значение счетчика не станет ненулевым до начала обработки. Алгоритм для изменения COUNT показан в примере 6.7.

Пример 6.7 – Мультипроцессорный счетчик операций

```
CT:   OR    4, IOF              ; XF0=1
                                ; Конец операции блокировки
      LDII  @COUNT, R1          ; Начало операции блокировки
                                ; Содержимое COUNT → R1
      BZ    CT                  ; Если COUNT=0, продолжает попытки
      SUBI  1, R1               ; Уменьшение R1 (=COUNT)
      STII  R1, @COUNT          ; Изменяет COUNT, XF0=1
                                ; Конец операции блокировки
```

Иногда нескольким процессорам необходимо иметь доступ к общим данным или другим общим ресурсам. Часть кода, которая должна обеспечивать доступ к общим данным, называется критической секцией.

Чтобы упростить программирование для критических секций, используют семафоры. Семафоры – это переменные, которые могут принимать только неотрицательные значения. Две простых неделимых операции определены для семафоров (S – семафор).

```
V(S):          S + 1 → S
P(S):   P:     Если    (S==0), перейти к P
               Иначе   S - 1 → S
```

Неделимость V(S) и P(S) означает, что когда эти процессы обращаются и модифицируют семафор S, они только выполняют доступ и модификацию S.

Для ввода критической секции операция P выполняется на общем семафоре, устанавливая S (S инициализируется в 1). Первый процессор, выполняющий P(S), будет иметь возможность вводить эти критические секции. Все другие процессоры блокированы, потому что S стало равным 0. После прохождения этой критической секции процессор выполняет V(S), таким образом, разрешая другому процессору успешно выполнить P(S).

Коды ПЦОС для V(S) приведены в примере 6.8, и коды для P(S) приведены в примере 6.9.

Пример 6.8 – Реализация V(S)

```
V:    LDII   @S, R0     ; Начало блокированного чтения (XF0 = 0)
                        ; Содержимое S → R0
      ADDI   1, R0      ; Инкремент R0 (=S)
      STII   R0, @S     ; Изменение S, конец блокировки (XF0=0)
```

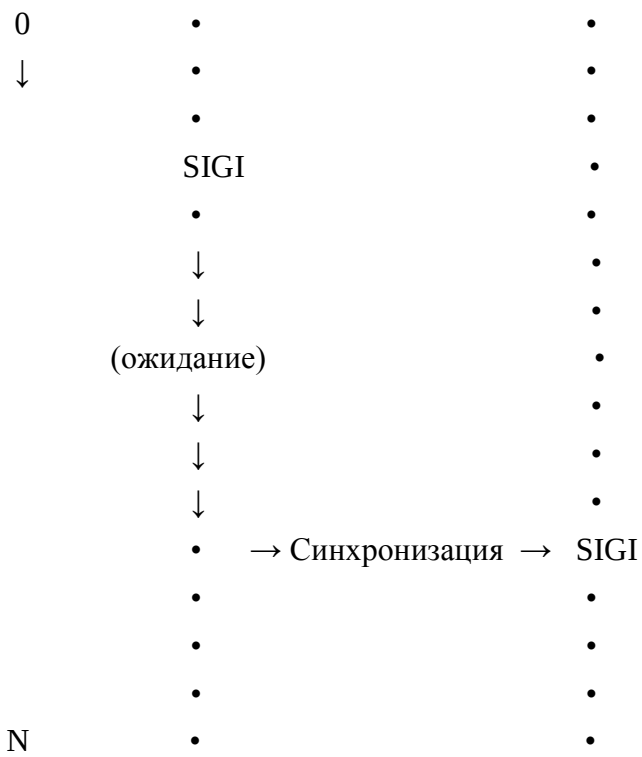
Пример 6.9 – Реализация P(S)

```
P:    OR     4, IOF     ; Конец блокировки (XF0=1)
      LDII   @S, R0     ; Начало блокированного чтения S
                        ; Содержимое S → R0
      BZ     P          ; Если S=0, переход на P и проверка заново
      SUBI   1, R0      ; Декремент R0 (=S)
      STII   R0, @S     ; Изменение S, конец блокировки (XF0=1)
```

Операция SIGI может быть использована для синхронизации на командном уровне нескольких ПЦОС.

Процессор 1 работает до тех пор, пока он выполняет SIGI. И далее он ожидает до тех пор, пока процессор 2 выполняет SIGI. В этой точке два процессора синхронизируются и продолжают выполнение.

Пример 6.10 – Программа для синхронизации двух ПЦОС на программном уровне



6.5 Операция сброса

ПЦОС имеет внешний сигнал сброса RESET#, который используется для обеспечения сброса системы. Данный подраздел описывает операцию сброса.

При включении питания состояние процессора ПЦОС не определено. Для установки процессора в известное состояние используется сигнал RESET#. Этот сигнал должен быть установлен в низкий уровень в течение 10 и более циклов H1 для гарантированного сброса системы. H1 – выходной тактовый сигнал, генерируемый ПЦОС (см. раздел 11 «Описание сигналов и электрические характеристики ПЦОС» для получения более подробной информации).

Сброс воздействует на другие выводы процессора синхронно или асинхронно. Синхронный сброс зависит от внутренней синхронизации ПЦОС. Асинхронный сброс прямо влияет на выводы и происходит быстрее синхронного.

При системном сбросе выполняются следующие дополнительные операции:

- сбрасывается периферия. Это синхронная операция. Сброс периферии описан в разделе 8 «Периферийные устройства»;
- сбрасываются регистры управления внешней шиной. Значения регистров управления при сбросе описаны в разделе 7 «Работа с внешней шиной»;
- в следующие регистры ЦПУ загружается 0:
    - ST (регистр состояния ЦПУ);
    - IE (флаги разрешения прерывания ЦПУ/ПДП);
    - IF (флаги прерывания ЦПУ);
    - IOF (флаги ввода-вывода);
- вектор сброса считывается по адресу памяти 0h и загружается в PC (счетчик команд). Этот вектор содержит стартовый адрес программы сброса;
- начинается выполнение программы инициализации;
- система из нескольких ПЦОС с общей системной синхронизацией может быть сброшена и засинхронизирована. При переходе сигнала RESET# из 1 в 0 процессор переводится в известное состояние внутренней фазировки, поэтому все ПЦОС переходят в одно состояние по внутренней синхронизации.

Кроме вышеописанных, состояние остальных регистров после сброса не определено.

6.6 Прерывания

ПЦОС поддерживают несколько внутренних и внешних прерываний, которые могут использоваться в различных приложениях. В текущем подразделе обсуждается действие данных прерываний.

Дополнительная информация, относящаяся к внутренним прерываниям, приведена в разделе 8 «Периферийные устройства».

Внешние прерывания имеют внутреннюю синхронизацию, что иллюстрируется тремя триггерами, синхронизируемыми сигналами H1 и H3. Затем вход прерывания устанавливает соответствующий разряд регистра флага прерываний (IF), если прерывание активно.

Внешние прерывания записываются внутри по заднему фронту H1 (см. временные диаграммы – рисунки 11.20 – 11.43). Внешнее прерывание должно удерживаться в низком уровне, по меньшей мере, в течение одного цикла H1/H3 для определения его процессором.

Прерывания должны удерживаться в низком уровне в течение такого времени, чтобы задний фронт сигнала H1 попадал в данный промежуток от одного до двух раз. Если прерывание удерживается в низком уровне более чем при трех задних фронтах сигнала H1, может быть определено более одного прерывания.

6.6.1 Разряды управления прерываниями

Когда соответствующее прерывание обработано ЦПУ или контроллером ПДП, соответствующий разряд флага прерываний очищается внутренним сигналом подтверждения прерывания. Необходимо отметить, что если сигнал INTn все еще в низком уровне после появления сигнала подтверждения прерывания, разряд флага прерывания будет очищен только на один цикл, после чего установлен снова, т. к. сигнал INTn все еще в низком уровне. Соответственно, теоретически возможно, что в зависимости от того, когда считан регистр IF, этот разряд может быть нулем, даже если сигнал INTn в низком уровне. При сбросе ПЦОС в регистр флага прерываний записываются нули, очищая, таким образом, все необработанные (ожидающие обработки) прерывания.

Разряды регистра флагов прерываний могут быть считаны и записаны из программы. Запись 1 в соответствующий разряд IF устанавливает соответствующий флаг прерывания в 1. Соответственно, запись 0 сбрасывает соответствующий флаг прерывания в 0. Таким образом, все прерывания могут быть записаны и/или очищены программным путем. Т. к. флаги прерываний могут быть считаны, выводы прерываний могут быть включены как программные, если не требуется интерфейс, управляемый по прерываниям.

Внутренние прерывания работают подобным же образом. Разряды регистра IF, относящиеся к внутренним прерываниям, могут быть считаны и записаны из программы. Запись 1 в соответствующий разряд IF устанавливает соответствующий флаг прерывания в 1, запись 0 – очищает его. Все внутренние прерывания имеют длину в один цикл H1/H3.

Разряд разрешения глобального прерывания ЦПУ (GIE), расположенный в регистре состояния ЦПУ (ST), управляет всеми прерываниями ЦПУ. Все прерывания ПДП (DMA) управляются разрядом разрешения глобального прерывания ПДП, который не зависит от ST (GIE) и относится только к ПДП. Разряд разрешения глобального прерывания ПДП зависит, в частности, от состояния разрядов DMA SYNCH. Они впрямую не доступны из программы (см. раздел 11 «Описание сигналов и электрические характеристики ПЦОС»). Результат операции AND над содержимым разряда флага прерывания и разрешениями прерывания поступает в блок обработки внутренних прерываний.

Для обеспечения максимальной производительности в обработке прерываний, предусмотрена команда подтверждения прерывания (IACK). IACK управляет выводом IACK# и обеспечивает холостое (фиктивное) чтение. Чтение выполняется по адресу, определяемому операндом команды IACK. При использовании IACK он обычно установлен на начало программы обработки прерывания. Для соответствующих приложений он может быть установлен на конец программы обработки прерываний или вовсе не является необходимым.

6.6.2 Рассмотрение прерываний в ПЦОС

Следует внимательно рассмотреть вопросы, касающиеся прерываний ПЦОС, в особенности при изменении содержимого регистра состояния, когда устанавливается разряд

глобального разрешения прерывания (GIE). Это может привести к неправильной установке и сбросу разряда GIE, как описано ниже.

Разряд GIE установлен в 0 при прерывании. Это может привести к сбою процессора, если в течение двух циклов по определению прерывания из какой-либо команды делается попытка чтения или изменения регистра состояния.

Например, если регистр состояния записывается в стек, он будет сохранен неправильно, если прерывание будет подтверждено двумя циклами ранее команды сохранения.

Когда сигнал прерывания определен, ПЦОС продолжает выполнение команд, находящихся в стадии чтения и декодирования на конвейере. Однако, поскольку прерывание подтверждено, разряд GIE сбрасывается в 0, и команда сохранения, находящаяся в конвейере, запишет неверное значение регистра состояния. Например, если программа выглядит следующим образом:

```
                        ...
                        NOP
Прерывание определено → LDI   @V_ADDR, AR1
                        MPYI  *AR1, R0
                        PUSH  ST
                         ...
                        POP   ST
                         ...
```

– то команда PUSH ST сохранит содержимое ST в памяти, при этом GIE=0.

Так как предполагается, что устройство имеет GIE=1, то команда POP ST возвратит неверное значение регистра состояния в ST.

Подобная же ситуация может возникнуть, если GIE =1 и выполняемая инструкция предназначена для изменения других разрядов состояния без установки разряда GIE.

В вышеприведенном примере подобная ошибка может возникнуть при определении прерывания за два цикла до команды POP ST. В этом случае прерывание очищает разряд GIE, но выполнение команды POP ST устанавливает разряд GIE. Т. к. прерывание было определено, то программа обработки прерываний будет выполнена с разрешенными прерываниями, в то время как ожидался запрет.

В подобной ситуации одним из решений может быть использование TRAP.

Например, можно использовать TRAP0 для сброса GIE и TRAP1 для его установки. TRAP0 и TRAP1 могут быть выполнены при использовании команд RETS и RETI, соответственно.

Другим выходом является включение следующего программного фрагмента, который защищает от изменения или сохранения статусный регистр путем запрещения прерываний через регистр разрешения прерывания:

```
PUSH   IE          ; Сохраняет регистр IE.
                    ; Команды включены для решения проблем с конвейером
LDI    0, IE       ; Очистка регистра IE
```

```
NOP                 ; Две NOP или другие команды
NOP                 ;
AND  0DFFFh, ST     ; Установить GIE=0.
                    ; Команда, которая считывает или записывает в регистр ST
POP  IE             ; Команды включены для решения
                     ; проблем с конвейером
```

ПЦОС имеет два дополнительных исключения при обработке прерываний:

- первое исключение – разряд глобального разрешения прерываний регистра состояния (GIE) может быть ошибочно сброшен в 0 (запрещающая установка), если выполнены все следующие условия:

- выбрана команда условного системного прерывания (TRAPcond),

- условие TRAP не выполнено, и возник конфликт конвейера, что ведет за собой задержку фазы декодирования или чтения команды.

В течение фазы декодирования условного программного прерывания, прерывания временно запрещены для гарантии того, что системное прерывание будет выполнено до последующего прерывания. Если возникает конфликт конвейера, вызывая задержку выполнения условного системного прерывания, состояние запрета прерывания может стать последним известным состоянием разряда GIE. В случае если условие системного прерывания не выполняется (false), то прерывания будут постоянно запрещены до тех пор, пока не будет непосредственно установлен разряд GIE. Условие не присутствует само по себе, когда условие системного прерывания выполняется (true), потому что при нормальном выполнении команды GIE сбрасывается, и стандартным подходом к программированию является установка GIE в единицу до окончания подпрограммы системного прерывания.

Вот пример нескольких последовательностей команд, которые могут привести к конфликту конвейера:

```
a)   LDI        mem, SP
     TRAPcond   n
b)   LDI        mem, SP
     NOP
     TRAPcond   n
c)   STI        SP, mem
     TRAPcond   n
d)   STI        Rx, *ARy
     LDI        *ARx, Ry
    || LDI      *ARz, Rw
     TRAPcond   n
```

Другие подобные условия также могут привести к задержке в выполнении.

Однако можно порекомендовать следующее решение для преодоления проблемы: ввести две команды NOP непосредственно перед командой TRAPcond. Одна команда NOP в некоторых случаях не помогает, как показано выше в примере (b).

Это предотвращает возможности возникновения многих конфликтов конвейера в непосредственно предшествующих командах и позволяет выполнять условный переход по системному программному прерыванию без задержек;

- второе исключение – асинхронные обращения к регистру флагов прерываний (IF) может привести к ошибке распознавания и обработки прерывания в ПЦОС. Это может произойти в случае, когда прерывания сгенерированы и записаны в регистр IF в том же цикле, когда ЦПУ изменяет содержимое IF.

Логика работы схемы прерываний такова, что ЦПУ имеет приоритет записи; таким образом, прерывание может быть потеряно. Это также верно в случае внутреннего прерывания, такого как прерывание ПДП. Эта ситуация может возникнуть как результат решения опросить определенные прерывания или желания очистить текущие (ожидающие обработки) прерывания, которые установлены из-за большой длины импульса на входе прерывания. Для случая большой длины импульса прерывание может быть выработано после того, как ЦПУ откликается на прерывание и пытается автоматически очистить его при обработке вектора прерывания.

В данном случае рекомендуется не использовать технику опроса прерывания, а строить внешние входы прерываний таким образом, чтобы ширина импульса на входе прерывания была между 1 – 2 длин командного цикла. Можно также при использовании техники опроса периодически запрещать и разрешать прерывания, которые будут опрашиваться, обеспечивая, таким образом, нормальную обработку прерываний; это автоматически очищает флаг прерываний без воздействия на другие прерывания. Если необходимо очистить прерывание, ожидающее обработку, рекомендуется использование ячейки памяти, служащей признаком того, что прерывание недействительно. Когда программа обработки прерывания сможет прочитать эту ячейку, необходимо очистить ее (если ожидающее прерывание недействительно) и сделать возврат.

6.6.3 Приоритеты и управление

ЦПУ управляет расстановкой приоритетов прерываний (см. таблицу 6.2, в которой даны адреса векторов сброса и прерываний и приоритеты). Если ПДП не использует прерывания для синхронизации пересылок, на него будет влиять обработка прерываний ЦПУ. Если ЦПУ вовлечен в конфликт конвейера (переход, регистр или память), реакции на прерывание не будет до тех пор, пока не разрешится конфликт.

Возможно, однако, прервать ЦПУ и ПДП одновременно с теми или другими прерываниями и, таким образом, синхронизировать их работу. Например, может быть, необходимо вызвать высокоприоритетную передачу ПДП, что предотвращает конфликты шин с ЦПУ, т. е. дает ПДП более высокий приоритет, чем ЦПУ.

Это можно сделать при использовании прерывания, которое указывает ЦПУ на необходимость перехода к программе прерывания, которая содержит команду IDLE. Тогда, если то же самое прерывание используется для синхронизации пересылок ПДП, счетчик пересылок ПДП может быть использован для генерации прерывания и, таким образом, возвращать управление ЦПУ вслед за пересылкой ПДП.

Так как ЦПУ и ПДП используют один и тот же набор флагов прерываний, ПДП может очистить флаг прерывания до того, как ЦПУ сможет отозваться на него.

Например, если прерывания ЦПУ запрещены, ПДП может отработать прерывания и таким образом очистить соответствующие флаги прерываний.

Если в конвейере возникает задержанный переход, выполнение прерываний задерживается до окончания выполнения перехода. Если прерывание возникает на первом цикле выборки команды, выбранная команда не выполняется, и адрес данной команды записывается в вершину системного стека. Если прерывание возникает после первого цикла выборки, в случае многоцикловой команды, вызванной состояниями ожидания, эта команда выполняется и адрес следующей команды, которая должна будет выбрана, записывается в вершину системного стека. Если программная выборка отсутствует, новая выборка не происходит. После записи в стек адреса соответствующей команды, вектор прерывания выбирается и загружается в счетчик команд (PC), и выполнение продолжается.

Таблица 6.2 – Адреса векторов сброса и прерываний

Сброс или прерывание вектора	Адрес	Приоритет	Функция
RESET#	0h	0	Вход внешнего сигнала сброса на вывод RESET#
INT0#	1h	1	Вход внешнего прерывания на вывод INT0#
INT1#	2h	2	Вход внешнего прерывания на вывод INT1#
INT2#	3h	3	Вход внешнего прерывания на вывод INT2#
INT3#	4h	4	Вход внешнего прерывания на вывод INT3#
XINT0	5h	5	Внутреннее прерывание; вырабатывается, когда буфер передачи последовательного порта 0 пуст
RINT0	6h	6	Внутреннее прерывание; вырабатывается, когда буфер приема последовательного порта 0 заполнен
XINT1	7h	7	Внутреннее прерывание; вырабатывается, когда буфер передачи последовательного порта 1 пуст
RINT1	8h	8	Внутреннее прерывание; вырабатывается, когда буфер приема последовательного порта 1 заполнен
TINT0	9h	9	Внутреннее прерывание; вырабатывается таймером 0
TINT1	0Ah	10	Внутреннее прерывание; вырабатывается таймером 1
DINT	0Bh	11	Внутреннее прерывание; вырабатывается контроллером ПДП

В ПЦОС ЦПУ и ПДП могут получать и обрабатывать прерывания параллельно. Прерывания опрошены, и ЦПУ и ПДП начинают их обработку. В течение прерываний, относящихся к ЦПУ, флаг прерывания, соответствующий разрешенному прерыванию с самым высоким приоритетом, очищается, и GIE устанавливается в ноль. ЦПУ завершает все выбранные команды. Вектор прерывания выбирается и загружается в PC, и ЦПУ продолжает выполнение. Цикл ПДП такой же, как для ЦПУ. После очистки соответствующего флага прерывания ПДП работает в соответствии с состоянием разряда SYNCH в регистре глобального управления ПДП.

7 Работа с внешней шиной

Память и внешние периферийные устройства могут быть доступны через два внешних интерфейса на ПЦОС: основной и расширенный. Генерация состояния ожидания, обеспечивающая доступ к медленной памяти и периферии, может управляться путем манипуляции с картированными в памяти регистрами управления, связанными с интерфейсами и внешними входными сигналами.

Основные положения, связанные с аппаратным интерфейсом и обсуждаемые в данном разделе, приведены ниже:

- регистры управления внешним интерфейсом (подраздел 7.1):
    - основная шина (7.1.1);
    - расширенная шина (7.1.2);
- синхронизация внешнего интерфейса (подраздел 7.2);
- программируемые состояния ожидания (подраздел 7.3);
- программируемые групповые переключения (подраздел 7.4).

7.1 Регистры управления внешним интерфейсом

ПЦОС поддерживает два внешних интерфейса: основной и расширенный. Основная шина состоит из 32-разрядной шины данных, 24-разрядной шины адреса и набора сигналов управления. Расширенная шина состоит из 32-разрядной шины данных, 13-разрядной шины адреса и набора сигналов управления. Обе шины поддерживают программно-управляемые состояния ожидания и внешний входной сигнал готовности и обе шины могут быть использованы для доступа данных, программ и ввода-вывода.

Доступ определяется по активному сигналу строба (STRB#, MSTRB#, IOSTRB#).

Когда обеспечивается доступ основной шины, сигнал STRB# находится в низком уровне. Расширенная шина в ПЦОС поддерживает два типа доступа:

- доступ к памяти по низкому уровню сигнала MSTRB#. Синхронизация доступа для MSTRB# та же, что и для доступа STRB# на основной шине;
- доступ к внешнему периферийному устройству по низкому уровню сигнала IOSTRB#.

Каждая из шин (основная и дополнительная) имеет связанный с ней регистр управления.

7.1.1 Регистр управления основной шиной

Регистр управления основной шиной – 32-разрядный регистр, который состоит из разрядов управления основной шины. Таблица 7.1 содержит список разрядов регистра с их именами и функциями.

Таблица 7.1 – Разряды регистра управления основной шиной

Разряд	Имя	Значение при сбросе	Функция
0	HOLDST	x (0 или 1)	Разряд состояния удержания. Этот сигнал указывает, будет ли порт удержан (HOLDST=1) или нет (HOLDST=0). Этот разряд состояния действителен при удержании порта аппаратно или программно.
1	NOHOLD	0	Сигнал удержания порта. NOHOLD разрешает или запрещает удержание порта внешним сигналом HOLD#. Когда NOHOLD=1, ПЦОС захватывает внешнюю шину и управляет ею, вне зависимости от запросов на продолжение или обслуживание от внешних устройств. Сигнал HOLDA# вырабатывается при получении сигнала HOLD#. Однако он выставляется при выработке внутреннего удержания (HIZ=1).
2	HIZ	0	Внутреннее удержание. Когда установлен (HIZ=1), порт устанавливается в состояние удержания. Это эквивалентно внешнему сигналу HOLD#. Устанавливая высокоимпедансное состояние, ПЦОС может «уступить» порт внешней памяти программным путем. HOLDA# устанавливается в низкий уровень при установке порта в высокоимпедансное состояние.
3-4	SWW	1 1	Программный режим ожидания. Совместно с WTCNT это двухразрядное поле определяет режим генерации состояния ожидания.
5-7	WTCNT	1 1 1	Программный режим ожидания. Это трехразрядное поле определяет число циклов для дальнейшего использования в программировании режима ожидания для генерации внутренних состояний ожидания. Диапазон от нуля (WTCNT=0 0 0) до семи (WTCNT=1 1 1) циклов H1/H3.
8-12	BNKCMP	1 0 0 0 0	Сравнение банка. Это 5-разрядное поле используется для определения числа старших разрядов адреса (MSB), которые будут использованы для определения размера банка. Устанавливается в 1 0 0 0 0 при сбросе.
13-31	Резервные	0	Считываются как 0.

7.1.2 Регистр управления расширенной шиной

Регистр управления расширенной шиной – 32-разрядный регистр, который состоит из разрядов управления для расширенной шины.

Таблица 7.2 – Разряды регистра управления расширенной шиной

Разряд	Имя	Значение при сбросе	Функция
2-0	Зарезервированы	0 0 0	Считываются как 0.
4-3	SWW	1 1	Программный режим ожидания. Совместно с WTCNT это двухразрядное поле определяет режим генерации состояния ожидания. Устанавливается в 1 1 при сбросе.
7-5	WTCNT	1 1 1	Программный режим ожидания. Это трехразрядное поле определяет число циклов для дальнейшего использования в программировании режима ожидания для генерации внутренних состояний ожидания. Диапазон от нуля (WTCNT=0 0 0) до семи (WTCNT=1 1 1) циклов H1/H3. Устанавливается в 1 1 1 при сбросе.
31-8	Зарезервированы	0-0	Считываются как 0.

7.2 Временные диаграммы внешнего интерфейса

В этом подразделе представлены функциональные временные диаграммы операций на основной и расширенной шинах, двух независимых параллельных шинах ПЦОС. Детальные временные спецификации для всех сигналов ПЦОС приведены в разделе 11 «Описание сигналов и электрические характеристики ПЦОС».

Параллельные шины обеспечивают три взаимоисключающих адресных пространства через использование трех отдельных сигналов управления: STRB#, MSTRB# и IOSTRB#. Сигнал STRB# управляет запросами на основной шине, а сигналы MSTRB# и IOSTRB# управляют запросами на расширенной шине, т. к. две шины независимы, два запроса могут идти параллельно.

За исключением переключения банка и внешней функции HOLD, временные диаграммы циклов основной шины и циклов расширенной шины MSTRB# идентичны и обсуждаются вместе. В случае идентичности диаграммы используется обозначение (M)STRB#. Аналогично обозначения (X)R/W#, (X)A, (X)D и (X)RDY# используются для обозначения эквивалентности сигналов основной и расширенной шины. Циклы IOSTRB# расширенной шины имеют другую диаграмму и обсуждаются отдельно.

7.2.1 Диаграммы основной шины

Все циклы шины включают в себя целое число тактовых циклов H1. Один цикл H1 определен от одного падающего фронта H1 до следующего падающего фронта. Для полноскоростных запросов (без состояния ожидания) запись занимает два цикла H1 и чтение занимает один цикл; однако, если чтение следует за записью, чтение занимает два цикла. Это применимо как для основной шины, так и для доступа по MSTRB# к расширенной шине С другой стороны (с точки зрения ЦПУ и ПДП), запись требует только одного цикла при отсутствии текущих обращений по этому интерфейсу. Нижеследующие рассуждения относятся к обращениям с нулевым состоянием ожидания, кроме специально оговоренных случаев.

Сигнал (M)STRB# находится в низком уровне, как при чтении, так и при записи, которые продолжаются в течение одного цикла H1. Кроме того, до и после активной части – (M)STRB# низкий – только для записи присутствует переходный цикл H1.

В течение переходного цикла происходит следующее:

- (M)STRB# – высокий;

- если требуется, изменяется состояние (X)R/W# по возрастающему фронту H1;

- если требуется, по возрастающему фронту H1 изменяется адрес, если предыдущий цикл H1 был активной частью записи. Если предыдущий цикл H1 был чтением, адрес изменяется по падающему фронту H1.

Данные в цикле считываются как можно позднее для обеспечения максимального времени доступа по отношению к установке адреса. Необходимо обратить внимание, что хотя внешняя запись требует двух циклов внутри (с точки зрения ЦПУ и ПДП), они требуют только одного цикла при отсутствии текущих обращений по данному интерфейсу.

На типовых временных диаграммах для всех внешних интерфейсов строб (X)R/W# не изменяется, пока активны (M)STRB# или IOSTRB#.

Адрес и записываемые данные удерживаются приблизительно в течение половины цикла после изменения (M)STRB#.

Длительность (M)STRB#, (X)R/W# и (X)A также увеличивается на один цикл.

В последующем при выборе (X)RDY# он равен 0.

Пока изначально (X)RDY#=1, цикл записи увеличивается. (M)STRB#, (X)R/W# и (X)A увеличивается на один цикл. В последующем при выборе (X)RDY# он равен 0.

7.2.2 Циклы ввода-вывода расширенной шины

В противоположность основной шине и циклам (M)STRB#, IOSTRB# осуществляет чтение и запись в течение двух циклов (без состояния ожидания) при схожей временной диаграмме. В течение этих циклов адрес всегда изменяется по заднему фронту H1, и IOSTRB# находится в низком уровне от переднего фронта первого цикла H1 до переднего фронта второго цикла H1. Сигнал IOSTRB# всегда неактивен (в высоком уровне) между циклами, и XR/W# в высоком уровне – для чтения и в низком – для записи.

Для обращений IOSTRB# циклы чтения и записи требуют как минимум два цикла. Некоторые внешние периферийные устройства могут менять свои разряды состояния при чтении или записи. Важно, однако, чтобы адрес удерживался при связи с данной периферией. Для чтения и записи при активном IOSTRB#, он должен полностью охватываться адресом.

Для каждого дополнительного состояния ожидания IOSTRB#, XR/W# и XA увеличиваются на один цикл. Выбор XRDY# повторяется каждый цикл.

Стробы (STRB#, MSTRB#, IOSTRB#) переходят в 1. Адреса не определены и сигналы готовности (XRDY# или RDY#) игнорируются.

HOLD# – внешний асинхронный вход. Задержка по времени между установкой HOLD#=0 и HOLDA#=0 минимум 1 цикл. Если HOLDA#=0, то адрес, шина данных и связанные с ними стробы устанавливаются в высокоимпедансное состояние. Все обращения через интерфейс завершаются до подтверждения удержания.

7.3 Программируемые состояния ожидания

Управление генерацией состояния ожидания через регистры управления, картированные в памяти, связано как с основным, так и с расширенным интерфейсом.

Используется поле WTCNT для загрузки внутреннего таймера и поле SWW для выбора одного из четырех следующих режимов генерации состояния ожидания:

- внешний RDY#;
- полученный из WTCNT RDY# wtcnt;
- логическое AND (И) над RDY# и RDY# wtcnt;
- логическое OR (ИЛИ) над RDY# и RDY# wtcnt.

Четыре поля используются для выработки внутреннего сигнала готовности, RDY# int, который управляет доступом (обращениями). Пока RDY# int=1, текущий внешний запрос задерживается. Когда RDY# int=0, выполняется текущий запрос.

Т. к. использование программируемых состояний ожидания для обоих внешних интерфейсов идентично, в следующих разделах обсуждается только интерфейс основной шины.

RDY# wtcnt – внутренний сигнал готовности. Когда начинается внешнее обращение, значение WTCNT загружается в счетчик. WTCNT может иметь значение от 0 до 7. Счетчик уменьшается каждый цикл H1/H3, пока не станет равным 0. После установки счетчика в 0 его значение не меняется до следующего обращения.

Пока счетчик не 0, RDY# wtcnt=1. Пока счетчик равен 0, RDY# wtcnt=0.

Когда SWW=0 0, RDY# int зависит только от RDY#. RDY# wtcnt игнорируется.

Таблица истинности для данного режима приведена в таблице 7.3.

Таблица 7.3 – Генерация состояния ожидания при SWW=0 0

RDY#	RDY# wtcnt	RDY# int
0	0	0
0	1	0
1	0	1
1	1	1

Когда SWW=0 1, RDY# int только от RDY# wtcnt, RDY# игнорируется.

Таблица истинности для данного режима приведена в таблице 7. 4.

Таблица 7.4 – Генерация состояния ожидания при SWW=0 1

RDY#	RDY# wtcnt	RDY# int
0	0	0
0	1	1
1	0	0
1	1	1

Когда SWW=1 0, RDY# int определяется как логическое ИЛИ (электрически И, т. к. эти сигналы имеют активный низкий уровень) от RDY# и RDY# wtcnt (см. таблицу 7.5).

Когда SWW=1 1, RDY# int определяется как логическое И (электрически ИЛИ, т. к. эти сигналы имеют активный низкий уровень) от RDY# и RDY# wtcnt (см. таблицу 7.6).

Таблица 7.5 – Генерация состояния ожидания при SWW=1 0

RDY#	RDY# wtcnt	RDY# int
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 7.6 – Генерация состояния ожидания при SWW=1 1

RDY#	RDY# wtcnt	RDY# int
0	0	0
0	1	1
1	0	1
1	1	1

7.4 Программируемое переключение банков

Программируемое переключение банков позволяет производить переключение между банками внешней памяти без вводимых извне состояний ожидания благодаря памяти, которая требует нескольких циклов для выключения. Переключение банков поддерживается только для основной шины.

Размер банка определяется числом разрядов, которые будут опрошены. Например, если BNKCMP=16, то для определения банка используется 16 старших разрядов адреса. Т. к. адреса 24-разрядные, размер банка определяется 8 младшими разрядами, определяя размер банка в 256 слов. Если BNKCMP≥16, компарируются только 16 старших разрядов. Поддерживаются размеры банков от 2^8=256 до 2^{24}=16M. В таблице 7.7 приведено соответствие между BNKCMP, разрядами адреса, использованными для определения банка и результирующим размером банка.

Таблица 7.7

BNKCMP	Разряды	Размер (32-битное слово)
00000	нет	2^{24}=16M
00001	23	2^{23}=8M
00010	23-22	2^{22}=4M
00011	23-21	2^{21}=2M
00100	23-20	2^{20}=1M
00101	23-19	2^{19}=512K
00110	23-18	2^{18}=256K
00111	23-17	2^{17}=128K
01000	23-16	2^{16}=64K
01001	23-15	2^{15}=32K
01010	23-14	2^{14}=16K
01011	23-13	2^{13}=8K
01100	23-12	2^{12}=4K
01101	23-11	2^{11}=2K
01110	23-10	2^{10}=1K
01111	23-9	2^9=512
10000	23-8	2^8=128
10000-11111	Зарезервированы	Не определены

ПЦОС имеет внутренний регистр, который содержит старшие разряды (как определено в поле BNKCMP) последнего адреса, используемого для чтения или записи через основной интерфейс. При сбросе регистр устанавливается в ноль.

Если старшие разряды адреса, который будет использован для текущего чтения через основной интерфейс, не совпадают с содержащимися во внутреннем регистре, чтение не выполняется в один цикл H1/H3. В течение дополнительного цикла синхронизации шина адреса переключается на новый адрес, но строб STRB# не активен (высокий). Содержимое внутреннего регистра заменяется значениями старших разрядов адреса, по данному адресу будет выполняться текущее чтение. Если старшие разряды адреса, который использован для текущего чтения, совпадают с содержащимися во внутреннем регистре, имеет нормальный цикл чтения.

Если осуществляются повторные чтения из одного банка памяти, дополнительные циклы чтения не требуются. Если осуществляются чтения из разных банков памяти, конфликт памяти разрешается введением дополнительных циклов.

Эта возможность может быть запрещена при установке BNKCMP в 0. Введение дополнительных циклов осуществляется только при чтении. Изменения старших разрядов во внутреннем регистре происходят для всех циклов чтения и записи через основной интерфейс.

8 Периферийные устройства

ПЦОС содержит два таймера, два последовательных порта и внутрикристальный контроллер прямого доступа к памяти (ПДП). Эти периферийные модули управляются через регистры, картированные в памяти, указывающие на соответствующую периферийную шину. Контроллер ПДП используется для обеспечения операций ввода-вывода без обращения к ЦПУ. Таким образом, это дает возможность обращения к медленной внешней памяти и периферии (АЦП, последовательным портам и т. д.) без соответствующего обращения к ЦПУ.

Основные характеристики периферийных устройств, обсуждаемые в данном разделе, приведены ниже:

- модули таймера (подраздел 8.1);
- регистры;
- генерация импульсов;
- режимы работы;
- последовательные порты (подраздел 8.2);
- регистры;
- рабочие конфигурации;
- временные диаграммы;
- примеры;
- контроллер ПДП (подраздел 8.3);
- регистры;
- операция пересылки памяти при ПДП;
- синхронизация каналов ПДП.

8.1 Модули таймера

Модули таймера ПЦОС – 32-разрядные счетчики общего назначения, работающие как таймер или счетчик событий, с двумя режимами сигнализации и внутренним или внешним тактированием. Модули таймера могут быть использованы для выдачи сигналов ПЦОС (или внешним устройствам) через определенные интервалы или считывания внешних событий. С внутренним тактированием таймер может быть использован в качестве указателя внешнему ЦАП начать преобразование или прервать контроллер ПДП ПЦОС для начала пересылки данных. Прерывание таймера является одним из внутренних прерываний. С внешней синхронизацией таймер может считать внешние события и прерывать ЦПУ после определенного числа событий. Доступный для каждого таймера вывод входа/выхода может быть использован как тактовый вход таймера, как выходной сигнал синхронизации или вывод входа/выхода общего назначения.

Каждый таймер имеет три регистра, картированных в памяти:

- регистр глобального управления (Global-control register);
- регистр периода (Period register);
- регистр счетчика (Counter register).

Регистр глобального управления определяет режим работы таймера, отслеживает состояние таймера и управляет выводом входа/выхода таймера.

Регистр периода определяет частоту выдачи сигналов таймером.

Регистр счетчика содержит текущее значение инкрементируемого счетчика. Таймер может инкрементироваться по переднему или заднему фронту входного сигнала синхронизации. Счетчик устанавливается в 0 и может вызвать внутреннее прерывание, когда его значение равно значению, содержащемуся в регистре периода. Генератор импульсов генерирует два типа сигналов: импульс или такт.

8.1.1 Регистр глобального управления таймером

Регистр глобального управления таймером – 32-разрядный регистр, содержащий разряды глобального управления и управления портом для модуля таймера. Таблица 8.1 определяет разряды регистра, их имена и функции. Разряды 3-0 – разряды управления портом; разряды 11-6 – разряды глобального управления таймером. Необходимо обратить внимание, что при сбросе все разряды устанавливаются в 0, за исключением DATIN (устанавливается по значению на TCLK).

Таблица 8.1 – Регистр глобального управления таймером

Разряды	Имя	Значение при сбросе	Функция
1	2	3	4
0	FUNC	0	FUNC управляет работой TCLK: если FUNC=0, TCLK конфигурируется как цифровой порт ввода-вывода общего назначения; если FUNC=1, TCLK конфигурируется как вывод таймера
1	I/O	0	Если FUNC=0 и CLKSRC=0, TCLK конфигурируется как вывод входа/выхода общего назначения. В этом случае, если I/O=0, TCLK конфигурируется как входной вывод общего назначения. Если I/O=1, TCLK конфигурируется как выходной вывод общего назначения.
2	DATOUT	0	Управляет TCLK, когда ПЦОС находится в режиме порта ввода-вывода. Может быть также использован как ввод в таймер.
3	DATIN	Уровень на TCLK	Вход данных на TCLK или DATOUT. При запись состояние не изменяется.
5-4	Зарезер-вированы	0-0	Считываются как 0.
6	GO	0	Разряд GO сбрасывает и запускает счетчик таймера. Когда GO=1 и таймер не удерживается, счетчик обнуляется и начинает инкрементироваться по следующему возрастающему фронту тактового входа таймера. GO очищается по тому же возрастающему фронту. GO=0 не влияет на таймер.

Окончание таблицы 8.1

1	2	3	4
7	HLD	0	Сигнал удержания счетчика. Когда этот разряд ноль, счетчик запрещен и удерживается в его текущем состоянии. Когда таймер управляется TCLK, состояние TCLK тоже удерживается. Внутренний счетчик (с делением на два) тоже удерживается таким образом, что счетчик может продолжиться с состояния останова при установке HLD в 1. Регистры таймера могут считываться и модифицироваться, пока счетчик удерживается. RESET# имеет более высокий приоритет, чем HLD. В таблице 8.2 приводится результат записи в GO и HLD.
8	C/P	0	Управление режимом такт/импульс. Когда C/P=1, выбирается режим такт, что указывает на то, что флаг состояния и внешний выход будут иметь цикл со скважностью 50 %. Когда C/P=0, флаг состояния и внешний выход будут активными для одного цикла H1 в течение каждого периода таймера.
9	CLKSRC	0	Определяет источник тактирования таймера. Когда CLKSRC=1, для инкрементирования счетчика будет использоваться внутренний такт с частотой в половину частоты H1. Разряд INV не влияет на внутренний источник тактирования. Когда CLKSRC=0, для инкрементирования счетчика может использоваться внешний сигнал с вывода TCLK. Внешний такт синхронизирован изнутри, обеспечивая, таким образом, возможность работы с внешними асинхронными источниками тактирования, которые не превышают максимально поддерживаемой частоты внешнего такта. Это будет меньше, чем f(H1)/2.
10	INV	0	Разряд управления инверсией. Если использован внешний источник тактирования и INV=1, то внешний такт инвертируется при входе в счетчик. Если выход генератора импульсов подключен к TCLK и INV=1, выход инвертируется до вывода TCLK. Если INV=0 инверсии не происходит ни на входе, ни на выходе таймера. При использовании TCLK как входа/выхода порта состояние INV, вне зависимости от его значения, не влияет на функционирование.
11	TSTAT	0	Указывает на состояние таймера. Он отслеживает выход неинвертированного вывода TCLK. Этот флаг выставляет прерывание ЦПУ при переходе из 0 в 1. Запись не влияет.
31-12	Зарезервированы	0-0	Считываются как 0.

В таблице 8.2 приводится результат записи при использовании определенных значений разрядов GO и HLD в регистр глобального управления.

Таблица 8.2 – Результат записи определенных значений GO и HLD

GO	HLD	Результат
0	0	Останов всех операций таймера. Сброс не происходит (значение сброса).
0	1	Таймер обрабатывается из состояния, предшествующего записи.
1	0	Все операции таймера остановлены, включая обнуление счетчика. Разряд GO не очищается до того, как таймер выйдет из состояния удержания.
1	1	Таймер сбрасывается и запускается.

8.1.2 Регистры периода таймера и счетчика

32-разрядный регистр периода таймера используется для определения частоты выдачи сигнала таймера. Регистр счетчика таймера – 32-разрядный регистр, который сбрасывается в ноль каждый раз по достижении значения, находящегося в регистре периода. Оба регистра устанавливаются в ноль при сбросе.

Определенные условия, такие как ноль в регистре периода и переполнение счетчика, влияют на работу таймера. Эти условия перечислены ниже:

- когда регистры периода и счетчика установлены в 0, работа таймера зависит от режима, выбранного через C/P. При выборе импульсного режима (C/P=0) TSTAT устанавливается и остается в установленном состоянии. При тактовом режиме (C/P=1) цикл составляет 2/f(H1) и внешний такт игнорируется;

- когда регистр счетчика не 0, а регистр периода=0, счетчик будет считать, перейдет через 0 и будет вести себя, как описано выше;

- когда регистр счетчика установлен на значение, большее значения регистра периода, счетчик может переполниться при инкрементировании. Когда счетчик достигнет максимального 32-разрядного значения (0FFFFFFFFh), он просто перейдет через 0 и продолжит счет.

Записи с периферийной шины «передавливают» изменения регистра счетчика, и новое состояние изменяет значение регистра управления.

8.1.3 Генерация импульсов таймера

Генератор импульсов таймера может генерировать несколько различных внешних сигналов. Эти сигналы могут быть инвертированы разрядом INV. Источник внутреннего такта f(такт таймера) в обоих случаях имеет частоты f(H1)/2, а генерируемый извне источник такта f(такт таймера) может иметь максимальную частоту f(H1)/2,6. В импульсном режиме (C/P=0) ширина импульса составляет 1/f(H1).

Период выдачи сигнала таймера определяется частотой входа такта таймера и регистром периода. Следующие выражения определены либо для внутреннего, либо для внешнего тактирования таймера:

f(импульсный режим) = f(такта таймера) / период регистра

f(тактовый режим) = f(такта таймера) / (2 × период регистра)

Если регистр периода равен 0, см. 8.1.2.

8.1.4 Режимы работы таймера

Таймер может передавать с выхода и принимать на вход в нескольких различных режимах, зависящих от установок CLKSRC, FUNC и I/O.

Режимы работы таймера описаны ниже.

Если CLKSRC=1 и FUNC=0, вход таймера поступает с внутреннего такта. Внутреннее тактирование не зависит от разряда INV. В этом режиме TCLK подключен к управлению ввода-вывода порта и может быть использован как вывод ввода-вывода общего назначения. Если I/O=0, TCLK конфигурируется как вход общего назначения, состояние которого может быть считано в DATIN. При этом DATOUT не влияет ни на TCLK, ни на DATIN. Если I/O=1, TCLK конфигурируется как выход общего назначения. DATOUT помещается на TCLK и может быть считано в DATIN.

Если CLKSRC=1 и FUNC=1, вход таймера поступает с внутреннего такта, выход таймера при этом идет на TCLK. Это значение может быть проинвертировано, используя INV, и значение выхода на TCLK может быть считано в DATIN.

Если CLKSRC=0 и FUNC=0, таймер управляется в зависимости от состояния разряда I/O. Если I/O =0, вход таймера идет от TCLK. Это значение может быть проинвертировано, используя INV, и значение выхода на TCLK может быть считано в DATIN. Если I/O =1, то TCLK – выходной вывод. При этом и TCLK и таймер управляются DATOUT. Все переходы из 0 в 1 на DATOUT инкрементируют счетчик. Значение DATOUT может быть считано в DATIN.

Если CLKSRC=0 и FUNC=1, таймер управляется TCLK. Если INV=0, счетчик инкрементируется при всех переходах из 0 в 1 на TCLK. Если INV=1, счетчик инкрементируется при всех переходах из 1 в 0 на TCLK. Значение TCLK может быть считано в DATIN.

8.1.5 Прерывания таймера

Прерывание таймера вырабатывается при автоматическом сбросе регистра счетчика таймера в 0. Регистр счетчика таймера автоматически сбрасывается в 0 каждый раз, когда его значение больше или равно значению в регистре периода таймера. Прерывание таймера может быть использовано для прерывания либо ЦПУ, либо ПДП.

Управление разрешением прерывания существует для каждого таймера либо для ЦПУ, либо для ПДП, как указано в регистре разрешения прерываний ЦПУ/ПДП (см. 3.1.8).

При выработке прерывания таймера проверяется состояние соответствующего вывода TCLK (при FUNC=1 и CLKSRC=1) в регистре глобального управления таймера. Точное изменение состояния зависит от состояния разряда C/P.

8.1.6 Инициализация/реконфигурация таймера

Таймеры управляются через картированные в памяти регистры, адресуемые по соответствующей периферийной шине. Общая процедура для инициализации и/или реконфигурации таймеров:

- остановить таймер, очистив разряды GO/HLD в регистре глобального управления таймера. Чтобы сделать это, необходимо записать 0 в регистр глобального управления таймера. Необходимо обратить внимание, что таймер останавливается по RESET#;
- сконфигурировать таймер через регистр глобального управления таймера (с GO=HLD=0), регистр счетчика таймера и регистр периода таймера, если необходимо;
- запустить таймер установкой разрядов GO/HLD в регистре глобального управления таймера.

8.2 Последовательные порты

ПЦОС имеет два полностью независимых двунаправленных последовательных порта. Оба порта одинаковы и имеют дополнительно набор регистров управления.

Каждый последовательный порт может быть настроен на передачу 8-, 16-, 24- или 32-разрядных слов данных одновременно в обоих направлениях. Тактирование для каждого последовательного порта может быть как внутренним, посредством таймера регистра последовательного порта и регистра периода, так и внешним, посредством подключения внешнего источника синхронизации. Внутренняя синхронизация представляет собой разделенную частоту выходной синхронизации, f(H1). Поддерживается также непрерывный режим пересылки, при котором последовательный порт может передавать и принимать любое количество слов без нового импульса синхронизации.

Каждый последовательный порт имеет восемь регистров, картированных в памяти:

- регистр глобального управления;
- два регистра управления для 6 последовательных выводов входа/выхода;
- три регистра таймера приема/передачи;
- регистр передачи данных;
- регистр приема данных.

Регистр глобального управления контролирует общее функционирование последовательного порта и определяет режим работы последовательного порта. Два регистра управления порта контролируют функционирование 6 выводов последовательного порта. Буфер передачи содержит следующее полное передаваемое слово. Буфер приема содержит последнее полное принятое слово. Три дополнительных регистра связаны с секцией приема/передачи таймера последовательного порта.

Регистр глобального управления последовательного порта – 32-разрядный регистр, содержащий разряды общего управления последовательного порта.

В таблице 8.3 определены разряды регистра, названия разрядов и их функции.

Таблица 8.3 – Описание разрядов регистра глобального управления последовательного порта

Разряд	Имя	Значение при сбросе	Функция
1	2	3	4
0	RRDY	0	Если RRDY=1, в буфере приема имеются новые данные, готовые к считыванию. От считывания DRR до RRDY=1 проходит 3 цикла H1/H3. Возрастающий фронт этого сигнала выставляет RINT. Если RRDY=0 при сбросе, то приемный буфер не имеет новых данных с последнего чтения. RRDY=0 при сбросе и после считывания буфера приема.
1	XRDY#	0	Если XRDY#=1, последние данные записаны из буфера передачи в сдвигатель, и буфер готов к чтению нового слова. От загрузки сдвигового регистра передачи до установки XRDY#=1 проходит 3 цикла H1/H3.
2	FSXOUT	0	Устанавливает вывод FSX как вход (FSXOUT=0) или как выход (FSXOUT=1).

Продолжение таблицы 8.3

1	2	3	4
3	XSREMPTY	0	Если XSREMPTY=0, то передающий сдвиговый регистр пуст. Если XSREMPTY=1, то передающий сдвиговый регистр не пуст. Сброс или XRESET устанавливает данный разряд в 0.
4	RSRFULL	0	Если RSRFULL=1, наблюдается переполнение приемника. В непрерывном режиме RSRFULL устанавливается в 1, когда и RSR и DRR полны. В пакетном (непрерывном) режиме RSRFULL устанавливается в 1, когда и RSR и DRR полны, и приходит новый FSR. При чтении данный разряд устанавливается в 0. Этот разряд может быть установлен в 0 только при системном сбросе, сбросе приема последовательного порта (RRESET=1) или при чтении. Когда приемник пытается установить RSRFULL в 1 в то же самое время, когда происходит чтение глобального регистра, приемник обладает большим приоритетом, и RSRFULL устанавливается в 1. Если RSRFULL=0, переполнения приемника не происходит.
5	HS	0	Если HS=1, то режим квитирования передачи разрешен. Если HS=0, то режим квитирования передачи запрещен.
6	XCLKSRCE	0	Если XCLKSRCE=1, используется внутренняя синхронизация передачи. Если XCLKSRCE=0, используется внешняя синхронизация передачи.
7	RCLKSRCE	0	Если RCLKSRCE=1, используется внутренняя синхронизация приема. Если RCLKSRCE=0, используется внешняя синхронизация приема.
8	XVAREN	0	Этот разряд определяет фиксированную (XVAREN=0) или переменную (XVAREN=1) выдачу указателя периода данных при передаче. При фиксированном периоде данных FSX активен для последнего цикла XCLK и становится неактивным до начала передачи. При переменном периоде данных FSX активен в течение передачи всех разрядов. При использовании внешнего FSX и переменного указателя периода данных вывод DX управляется передатчиком, когда FSX удерживается активным, или пока слово не будет сдвинуто наружу.
9	RVAREN	0	Этот разряд определяет фиксированную (RVAREN=0) или переменную (RVAREN=1) выдачу указателя периода данных при приеме. При фиксированном периоде данных FSR активен для последнего цикла RCLK и становится неактивным до начала приема. При переменном периоде данных FSR активен в течение всего приема всех разрядов.
10	XFSM	0	Режим кадровой синхронизации передачи. Устанавливает порт в режим непрерывной работы (XFSM=1) или стандартный режим (XFSM=0). В непрерывном режиме только первое слово блока вырабатывает импульс синхронизации, а остаток просто непрерывно передается до окончания блока. В стандартном режиме с каждым словом связан импульс синхронизации.

Продолжение таблицы 8.3

1	2	3	4
11	RFSM	0	Режим кадровой синхронизации приема. Устанавливает порт в режим непрерывной работы (RFSM=1) или стандартный режим (RFSM=0). В непрерывном режиме только первое слово блока вырабатывает импульс синхронизации, а остаток просто непрерывно принимается без ожидания другого синхроимпульса. В стандартном режиме с каждым словом связан импульс синхронизации.
12	CLKXP	0	Полярность CLKX. Если CLKXP=0, то CLKX активен высоким уровнем. Если CLKXP=1, то CLKX активен низким уровнем.
13	CLKRP	0	Полярность CLKR. Если CLKRP=0, то CLKR активен высоким уровнем. Если CLKRP=1, то CLKR активен низким уровнем.
14	DXP	0	Полярность DX. Если DXP=0, то DX активен высоким уровнем. Если DXP=1, то DX активен низким уровнем.
15	DRP	0	Полярность DR. Если DRP=0, то DR активен высоким уровнем. Если DRP=1, то DR активен низким уровнем.
10	FSXP	0	Полярность FSX. Если FSXP=0, то FSX активен высоким уровнем. Если FSXP=1, то FSX активен низким уровнем.
11	FSRP	0	Полярность FSR. Если FSRP=0, то FSR активен высоким уровнем. Если FSRP=1, то FSR активен низким уровнем.
19-18	XLEN	0-0	Эти два разряда определяют длину слова передачи последовательного порта. Предполагается, что все данные (rignt-justified) установлены по правой границе буфера передачи, когда определены менее 32 разрядов.
21-20	RLEN	0-0	Эти два разряда определяют длину слова приема последовательного порта. Все данные (rignt-justified) установлены по правой границе буфера приема: 0-0 … 8 разрядов; 1-0 … 24 разряда; 0-1 …16 разрядов; 1-1 … 32 разряда.
22	XTINT	0	Разрешение прерывания таймера передачи: если XTINT=0, прерывание таймера передачи запрещено; если XTINT=1, прерывание таймера передачи разрешено.
23	XINT	0	Разрешение прерывания передачи: если XINT=0, прерывание передачи запрещено; если XINT=1, прерывание передачи разрешено.
24	RTINT	0	Разрешение прерывания таймера приема: если RTINT=0, прерывание таймера приема запрещено; если RTINT=1, прерывание таймера приема разрешено.
25	RINT	0	Разрешение прерывания приема: если RINT=0, прерывание приема запрещено; если RINT=1, прерывание приема разрешено.
26	XRESET	0	Сброс передатчика. Если XRESET=0, передающая часть последовательного порта сбрасывается. Для выхода из сброса передающей части последовательного порта необходимо установить XRESET в 1. Однако нельзя устанавливать XRESET в 1 до истечения минимум 3 циклов после установки XRESET в неактивное состояние. Это применимо только к системному сбросу. Установка XRESET в 0 не изменяет состояния регистров управления порта. Он устанавливает передатчик в состояние, соответствующее началу кадра данных. Сброс передатчика приводит к установке прерывания передачи. XFSM может быть записан без сброса регистра глобального управления.

Окончание таблицы 8.3

1	2	3	4
27	RRESET	0	Сброс приемника. Если RRESET=0, приемная часть последовательного порта сбрасывается. Для выхода из сброса приемной части последовательного порта необходимо установить RRESET в 1. Установка RRESET в 0 не изменяет состояния регистров управления порта. Он устанавливает приемник в состояние, соответствующее началу кадра данных. RFSM может быть записан без сброса регистра глобального управления.
31-28	Зарезервированы	0-0	Считываются как 0.

8.2.1 Регистр управления выводами FSX, DX, CLKX последовательного порта

Этот 32-разрядный регистр управляет работой выводов FSX, DX, CLKX последовательного порта. При сбросе все разряды данного регистра устанавливаются в 0. В таблице 8.4 определены разряды регистра, названия разрядов и их функции.

Таблица 8.4 – Разряды регистра управления выводами FSX, DX, CLKX последовательного порта

Разряд	Наименование	Значение при сбросе	Функция
0	CLKXFUNC	0	CLKXFUNC управляет работой CLKX. Если CLKXFUNC=0, CLKX установлен как цифровой порт ввода-вывода общего назначения. Если CLKXFUNC=1, то CLKX – вывод последовательного порта.
1	CLKXI/O	0	Если CLKXI/O=0, CLKX установлен как вход общего назначения. Если CLKXI/O=1, то CLKX – выход общего назначения.
2	CLKXDATOUT	0	Выход данных для CLKX.
3	CLKXDATIN	x *	Вход данных для CLKX. Запись не влияет.
4	DXFUNC	0	DXFUNC управляет работой DX. Если DXFUNC=0, DX установлен как цифровой порт ввода-вывода общего назначения. Если DXFUNC=1, то DX – вывод последовательного порта.
5	DXI/O	0	Если DXI/O=0, DX установлен как вход общего назначения. Если DXI/O=1, то DX – выход общего назначения.
6	DXDATOUT	0	Выход данных для DX.
7	DXDATIN	x *	Вход данных для DX. Запись не влияет.
8	FSXFUNC	0	FSXFUNC управляет работой FSX. Если FSXFUNC=0, FSX установлен как цифровой порт ввода-вывода общего назначения. Если FSXFUNC=1, то FSX – вывод последовательного порта.
9	FSXI/O	0	Если FSXI/O=0, FSX установлен как вход общего назначения. Если FSXI/O=1, то FSX – выход общего назначения.
10	FSXDATOUT	0	Выход данных для FSX.
11	FSXDATIN	x *	Вход данных на FSX. Запись не влияет.
31-12	Зарезервированы	0-0	Считываются как 0.

* x = 0 или 1

8.2.2 Регистр управления выводами FSR, DR, CLKR последовательного порта

Этот 32-разрядный регистр управляет работой выводов FSR, DR, CLKR последовательного порта. При сбросе все разряды данного регистра устанавливаются в 0. В таблице 8.5 определены разряды регистра, названия разрядов и их функции.

Таблица 8.5 – Разряды регистра управления выводами FSR, DR, CLKR последовательного порта

Разряд	Наименование	Значение при сбросе	Функция
0	CLKRFUNC	0	CLKRFUNC управляет работой CLKR. Если CLKRFUNC=0, CLKR установлен как цифровой порт ввода-вывода общего назначения. Если CLKRFUNC=1, то CLKR – вывод последовательного порта.
1	CLKRI/O	0	Если CLKRI/O=0, CLKR установлен как вход общего назначения. Если CLKRI/O=1, то CLKR – выход общего назначения.
2	CLKRDATOUT	0	Выход данных для CLKR.
3	CLKRDATIN	x *	Вход данных для CLKR. Запись не влияет.
4	DRFUNC	0	DRFUNC управляет работой DR. Если DRFUNC=0, DR установлен как цифровой порт ввода-вывода общего назначения. Если DRFUNC=1, то DR – вывод последовательного порта.
5	DRI/O	0	Если DRI/O=0, DR установлен как вход общего назначения. Если DRI/O=1, то DR – выход общего назначения.
6	DRDATOUT	0	Выход данных для DR.
7	DRDATIN	x*	Вход данных для DR. Запись не влияет.
8	FSRFUNC	0	FSRFUNC управляет работой FSR. Если FSRFUNC=0, FSR установлен как цифровой порт ввода-вывода общего назначения. Если FSRFUNC=1, то FSR – вывод последовательного порта.
9	FSRI/O	0	Если FSRI/O=0, FSR установлен как вход общего назначения. Если FSRI/O=1, то FSR – выход общего назначения.
10	FSRDATOUT	0	Выход данных для FSR.
11	FSRDATIN	x *	Вход данных на FSR. Запись не влияет.
31-12	Зарезервированы	00	Считываются как 0.

* x = 0 или 1.

8.2.3 Регистр управления таймера приема/передачи

32-разрядный регистр управления таймера приема/передачи содержит разряды управления для модуля таймера. При сбросе все разряды устанавливаются в 0. В таблице 8.6 перечислены разряды регистра, их наименования и функции. Разряды 5-0 управляют таймером передатчика. Разряды 11-6 управляют таймером приемника. См. подраздел 8.1 «Таймеры» для более полной информации о таймерах.

Таблица 8.6 – Регистр управления таймером приема/передачи

Разряд	Имя	Значение при сбросе	Функция
1	2	3	4
0	XGO	0	Разряд XGO сбрасывает и запускает счетчик таймера. Когда XGO=1 и таймер не удерживается, счетчик обнуляется и начинает инкрементироваться по следующему возрастающему фронту тактового входа таймера. XGO очищается по тому же возрастающему фронту. XGO=0 не влияет на таймер.
1	XHLD	0	Сигнал удержания счетчика передачи. Когда этот разряд ноль, счетчик запрещен и удерживается в его текущем состоянии. Внутренний счетчик с делением на два тоже удерживается таким образом, что счетчик может продолжиться с состояния останова при установке XHLD в 1. Регистры таймера могут считываться и модифицироваться, пока счетчик удерживается. RESET# имеет более высокий приоритет, чем XHLD.
2	XC/P	0	Управление режимом такт/импульс. Когда XC/P=1, выбирается режим такт, что указывает на то, что флаг состояния и внешний выход будут иметь цикл со скважностью 50 %. Когда XC/P=0, флаг состояния и внешний выход будут активными для одного цикла CLKOUT в течение каждого периода таймера.
3	XCLKSRC	0	Определяет источник синхронизации таймера передачи. Когда XCLKSRC=1, для инкрементирования счетчика будет использоваться внутренний такт с частотой в половину частоты CLKOUT. Когда XCLKSRC=0, для инкрементирования счетчика может использоваться внешний сигнал с вывода CLKX. Внешний такт синхронизирован изнутри, обеспечивая, таким образом, возможность работы с внешними асинхронными источниками тактирования, которые не превышают максимально поддерживаемой частоты внешнего такта. Это будет меньше, чем f(H1)/2,6.
4	Зарезервирован	0	Считывается как 0.
5	XTSTAT	0	Указывает состояние таймера передачи. Он следит за тем, что будет на выходе неинвертированного вывода CLKX. Данный флаг выставляет прерывание ЦПУ при переходе из 0 в 1. Запись не влияет.
6	RGO	0	Разряд RGO сбрасывает и запускает счетчик таймера. Когда RGO=1 и таймер не удерживается, счетчик обнуляется и начинает инкрементироваться по следующему возрастающему фронту тактового входа таймера. RGO очищается по тому же возрастающему фронту. RGO=0 не влияет на таймер.
7	RHLD	0	Сигнал удержания счетчика приема. Когда этот разряд ноль, счетчик запрещен и удерживается в его текущем состоянии. Внутренний счетчик с делением на два тоже удерживается таким образом, что счетчик может продолжиться с состояния останова при установке RHLD в 1. Регистры таймера могут считываться и модифицироваться, пока счетчик удерживается. RESET# имеет более высокий приоритет, чем RHLD.

Окончание таблицы 8.6

1	2	3	4
8	RC/P	0	Управление режимом R такт/импульс. Когда RC/P=1, выбирается режим такт, что указывает на то, что флаг состояния и внешний выход будут иметь цикл со скважностью 50 %. Когда RC/P=0, флаг состояния и внешний выход будут активными для одного цикла CLKOUT в течение каждого периода таймера.
9	RCLKSRC	0	Определяет источник синхронизации таймера приема. Когда RCLKSRC=1, для инкрементирования счетчика будет использоваться внутренний такт с частотой в половину частоты CLKOUT. Когда RCLKSRC=0, для инкрементирования счетчика может использоваться внешний сигнал с вывода CLKR. Внешний такт синхронизирован изнутри, обеспечивая, таким образом, возможность работы с внешними асинхронными источниками тактирования, которые не превышают максимально поддерживаемой частоты внешнего такта. Это будет меньше, чем f(H1)/2,6.
10	Зарезервирован	0	Считывается как 0.
11	RTSTAT	0	Указывает состояние таймера приема. Он следит за тем, что будет на выходе неинвертированного вывода CLKR. Данный флаг выставляет прерывание ЦПУ при переходе из 0 в 1. Запись не влияет.
31-12	Зарезервированы	0-0	Считываются как 0.

8.2.4 Регистр счетчика таймера приема/передачи

Регистр счетчика таймера приема/передачи представляет собой 32-разрядный регистр. Разряды 15-0 относятся к счетчику таймера передачи, разряды 31-16 относятся к счетчику таймера приема. Каждый счетчик устанавливается в 0 при достижении значения, хранящегося в регистре периода (см. 8.2.6). Он также устанавливается в 0 при сбросе.

Рисунок 8.1 – Регистр счетчика таймера приема/передачи

8.2.5 Регистр периода таймера приема/передачи

Регистр периода таймера приема/передачи – 32-разрядный регистр.

Разряды 15-0 относятся к периоду таймера передачи, разряды 31-16 относятся к периоду таймера приема. Каждый счетчик используется для определения периода таймера.

Он также устанавливается в 0 при сбросе.

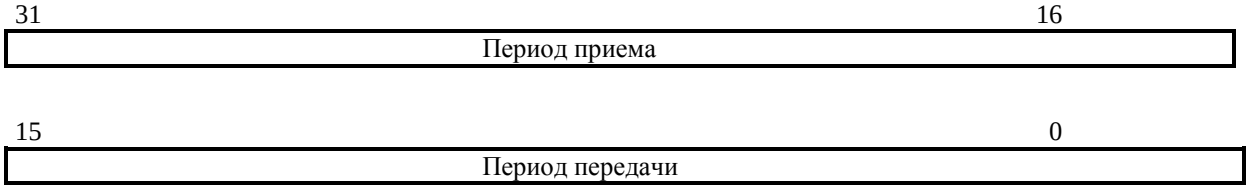


Рисунок 8.2 – Регистр периода таймера приема/передачи

8.2.6 Регистр передачи данных

Когда регистр передачи данных (DXR) загружен, передатчик загружает слово в передающий регистр сдвига (XSR) и поразрядно выдает его наружу. Задержка от момента записи в DXR до выставления FSX (который может быть также установлен извне) составляет два цикла CLKX. Слово не загружается в регистр сдвига, пока не очистится сдвигатель. Когда DXR загружается в XSR, выставляется разряд XRDY, указывающий на то, что буфер готов к приему следующего слова. Для передачи слова в сдвиговом передающем регистре предусмотрены 4 точки отвода, соответствующие 4 размерам передаваемого слова. Сдвиг происходит влево (от младших разрядов (LSB) к старшим (MSB)) с выдачей данных от старшего разряда в соответствующей точке отвода.

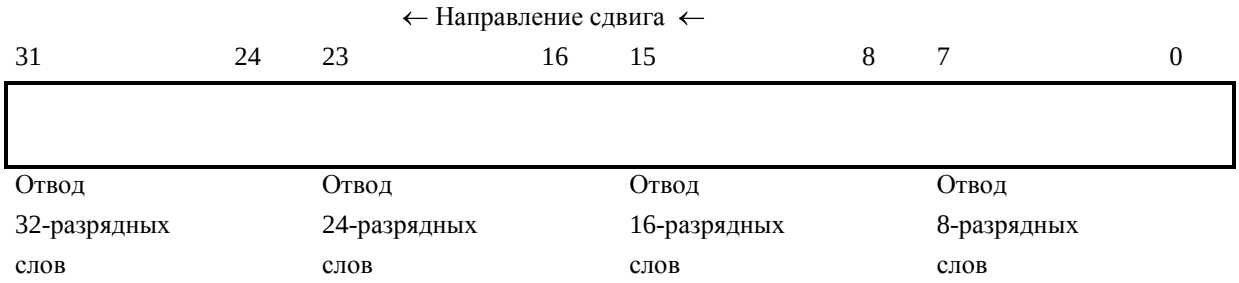


Рисунок 8.3 – Работа сдвигового регистра передачи

Когда приходят последовательные данные, приемник сдвигает их в приемный регистр сдвига (RSR). Когда определенное число разрядов сдвинуто внутрь, содержимое RSR записывается в регистр приема данных (DRR) и выставляется разряд состояния RRDY. Приемник обладает двойной буферизацией. Если DRR не считан, и RSR полон, прием останавливается. Новые данные, приходящие на вход DX, игнорируются. Приемный сдвигатель не будет записываться через DRR. DRR должен быть считан для того, чтобы новые данные из RSR записались в DRR. Если запись в DRR имеет место в то же самое время, что и передача из RSR в DRR, приоритет имеет именно передача.

Данные сдвигаются влево (от LSB к MSB). Предполагается, что принято 8-разрядное слово и старшие три байта буфера приема изначально не определены. На первой части рисунка 8.4 байт "a" был принят. Когда принимается байт "b", байт "a" сдвигается влево. Когда считывается регистр приема данных, считываются оба байта "a" и "b".

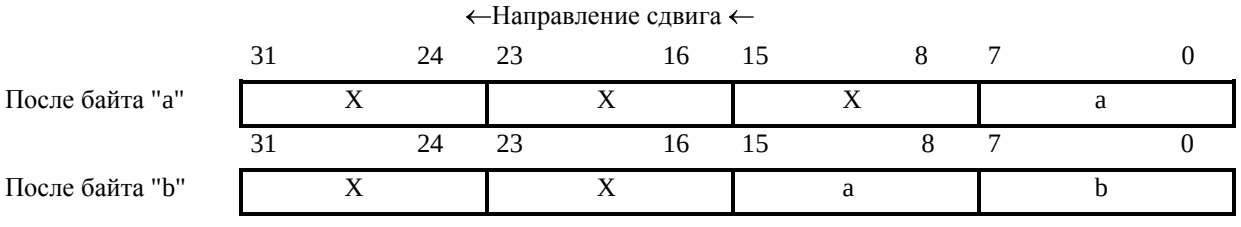


Рисунок 8.4 – Работа сдвига буфера приема

8.2.7 Установка параметров последовательного порта

Обеспечиваются несколько установок для работы таймера последовательного порта и его синхронизации. Синхроимпульсы для каждого последовательного порта могут вырабатываться как внутри, так и подаваться извне.

8.2.8 Временные диаграммы последовательного порта

Формула для вычисления частоты синхронизации последовательного порта с внутренней синхронизацией зависит от режима работы таймеров последовательного порта и может быть определена как:

f(импульсный режим) = f(такт таймера)/регистр периода

f(тактовый режим) = f(такт таймера)/(2 × регистр периода)

Источник внутреннего синхроимпульса f(такт таймера) имеет максимальную частоту f(H1)/2. Внешний источник синхронизации f(такт таймера) (CLKX или CLKR) имеет максимальную частоту меньше, чем f(H1)/2,6. См. временные диаграммы последовательного порта в разделе 11 и 8.1.3 относительно генерации импульс/такт таймера.

Передаваемые данные выдаются по возрастающему фронту выбранного синхросигнала порта. Принимаемые данные записываются в приемный регистр сдвига по заднему фронту синхросигнала порта. Все данные передаются и загружаются, начиная со старшего разряда и выравненными по правой границе. При передаче менее 32 разрядов, данные выравниваются в 32-разрядных буферах приема и передачи. Таким образом, младшие разряды буфера передачи – разряды, которые передаются.

Сигнал готовности передачи (XRDY) определяет готовность регистра передачи данных (DXR) к загрузке новых данных. XRDY становится активным, как только данные загружены в сдвиговый регистр передачи (XSR). Последнее слово может еще выдаваться, когда XRDY становится активным. Если DXR загружается до того, как закончена передача последнего слова, передаваемые разряды данных будут последовательными (непрерывными), т. е. младший разряд первого слова непосредственно предваряет старший разряд второго, с выдачей всех сигналов как при двух независимых передачах. XRDY становится неактивным после загрузки DXR и остается неактивным, пока данные загружены в сдвигатель.

Сигнал готовности приема (RRDY) активен, пока новое слово данных загружено в регистр приема данных и не считано. Как только данные считаны, разряд RRDY очищается.

Когда FSX определен как выход, активность сигналов определена однозначно внутренним состоянием последовательного порта (п/порт – далее используемое в таблицах сокращение). Если определен фиксированный период данных, FSX становится активным, когда DXR загружен в XSR для передачи.

Через один цикл синхросигнала порта FSX становится неактивным, и начинается передача данных. Если определен переменный период данных, вывод FSX становится активным при начале передачи данных и остается активным в течение всей передачи слова. Аналогично данные передаются через один цикл синхросигнала после их загрузки в регистр передачи данных.

Входной FSX в режиме фиксированной скорости передачи данных становится активным в течение, по крайней мере, одного цикла синхросигнала порта и снимается для начала передачи. После этого передатчик пересылает разряды, число которых определено разрядами LEN. В режиме переменного периода данных передатчик начинает передачу с момента установления FSX в активный уровень до того момента, пока определенное число разрядов не будет выдано наружу. В режиме переменной скорости передачи данных при изменении ста-

туса FSX до того, как все разряды будут выданы наружу, передача завершается, и вывод DX устанавливается в высокоимпедансное состояние. Вход FSR полностью комплементарен FSX.

При использовании внешнего FSX, если DXR и XSR пусты, запись в DXR приводит к пересылке DXR в XSR. Данные хранятся в XSR до возникновения FSX. При приеме внешнего FSX XSR начинает сдвиг данных. Если XSR ждет внешнего FSX, запись в DXR изменяет DXR, но пересылка DXR в XSR не происходит. XSR начинает сдвиг при приеме внешнего FSX или когда он сброшен, используя XRESET.

Непрерывные режимы передачи и приема

Когда выбран непрерывный режим, последующие записи не генерируются или ожидают сигнала нового синхроимпульса. Только первое слово блока начинается с активной синхронизации. После этого передача данных продолжается, пока загружаются данные в DXR до того, как будет передано последнее слово. Как только TXRDY становится активным, и все данные будут переданы из сдвигового регистра, вывод DX устанавливается в высокоимпедансное состояние и последующая запись в DXR инициирует новый блок и новый FSX.

Подобно FSR, приемник продолжает сдвиг внутрь новых данных и загрузку DRR.

Если чтение из буфера приема данных не происходит до того, как следующее слово сдвигается внутрь, последующие входящие данные будут потеряны. Разряд RFSM может быть использован для прерывания непрерывного режима приема.

Режим квитирования передачи

Режим квитирования передачи (HS=1) обеспечивает прямое соединение между процессорами. В этом режиме все слова данных передаются с ведущей единицей. Например, при передаче 8-разрядного слова, первый переданный разряд будет 1, предваряя 8-разрядное слово данных.

В этом режиме, передав слово, последовательный порт не будет передавать следующее слово, пока не примет отдельно переданный нулевой разряд. Таким образом, 1 разряд (leading one), предшествующий каждому слову данных, является, фактически, разрядом запроса.

После того, как последовательный порт принимает слово (с предшествующей 1), и это слово считано из DRR, принимающий последовательный порт передает единичный 0 (single zero) в передающий последовательный порт. Таким образом, единичный ноль рассматривается, как разряд подтверждения. Этот единичный разряд подтверждения передается каждый раз при чтении DRR, даже если DRR не содержит новые данные.

Когда последовательный порт установлен в режим квитирования передачи, внесение и удаление ведущей 1 для передаваемых данных и передача 0 для подтверждения приема данных и ожидание этого разряда подтверждения, выполняется автоматически.

При использовании этой схемы просто соединить процессоры без дополнительных аппаратных затрат и гарантировать надежную связь. В режиме квитирования передачи FSX автоматически конфигурируется как выход.

Непрерывный режим автоматически отменяется. После системного сброса или XRESET передатчик всегда установлен для передачи. Приемник и передатчик должны быть сброшены после введения режима квитирования передачи.

8.2.9 Источники прерываний последовательного порта

Последовательный порт имеет четыре источника прерываний:

- первый – прерывание передающего таймера: передний фронт XTSTAT вызывает одноцикловый импульс прерывания. Когда XTINT = 0, импульс прерывания запрещен;

- второй – прерывание приемного таймера: передний фронт RTSTAT вызывает одноцикловый импульс прерывания. Когда RTINT = 0, импульс прерывания запрещен;

- третий – прерывание передатчика: возникает непосредственно после пересылки DXR в XSR. Прерывание передатчика – одноцикловый импульс. Когда разряд регистра глобального управления последовательного порта XINT = 0, этот импульс прерывания запрещен;

- четвертый – прерывание приемника возникает непосредственно после пересылки RSR в DRR. Прерывание приемника – одноцикловый импульс. Когда разряд глобального управления последовательного порта RINT = 0, этот импульс прерывания запрещен.

Импульс прерывания передающего таймера и импульс прерывания передатчика объединяются по логическому ИЛИ для генерации флага прерывания передачи ЦПУ XINT. Импульс прерывания приемного таймера и импульс прерывания приемника объединяются по логическому ИЛИ для генерации флага прерывания приема ЦПУ RINT.

8.2.10 Функциональные операции последовательного порта

Далее иллюстрируются временные диаграммы функционирования различных режимов работы последовательного порта. Описание временных диаграмм приведено в предположении, что все полярности сигналов приняты положительными, т. е. CLKXP = CLKRP = DXR = DRP = FSXP = FSRP = 0. Логические временные диаграммы,когда одна или более полярности инвертированы, подобны, за исключением того, что касается противоположной полярности контрольных точек, т. е. передний вместо заднего фронта и т. д.

Эти обсуждения относятся к различным режимам работы и конфигурациям логики последовательного порта. Если необходимо переключить режимы работы или изменить конфигурации последовательного порта, то это надо делать только тогда, когда XRESET или RRESET установлен (низким), как предназначено. Следовательно, при изменении конфигурации передачи, XRESET должен быть низким, при изменении конфигурации приема RRESET должен быть низким. При использовании режима квитирования передачи, однако, т. к. передатчик и приемник взаимосвязаны, необходимо делать любые изменения конфигурации с обоими низкими XRESET и RRESET.

Все операции конфигурации последовательного порта могут быть в общих чертах классифицированы по двум категориям: временная диаграмма фиксированного периода данных и временная диаграмма переменного периода данных. Ниже обсуждается работа при фиксированной и переменной скорости передачи данных и всех ее вариантах.

Временная диаграмма работы при фиксированной скорости передачи данных

Пересылки последовательного порта при фиксированной скорости передачи данных могут быть в двух вариантах: пакетный режим и непрерывный режим. В пакетном режиме работы пересылки отдельных слов отделены периодами отсутствия активности последовательного порта. В непрерывном режиме нет промежутков между успешными пересылками слов; первый разряд нового слова пересылается по последующему импульсу CLKX/CLKXR, следующим за последним разрядом предыдущего слова. Это происходит непрерывно до прерывания процесса.

В пакетном режиме с фиксированной скоростью передачи данных импульсы FSX/FSR начинают пересылки, и каждая пересылка включает одно слово. При внутренней генерации FSX, передача начинается загрузкой DXR. В этом режиме имеется задержка около 2,5 циклов CLKX (в зависимости от частоты CLKX и H1) с момента загрузки DXR до возникновения FSX. Если FSX – внешний, то импульс FSX инициирует передачу и 2,5-цикловая задержка действительно становится необходимой для обеспечения требования загрузки DXR по FSX. Следовательно, в этом случае DXR должен быть загружен не позднее трех циклов CLKX до возникновения FSX. Как только XSR загружен из DXR, генерируется XINT.

При операциях приема, когда пересылка уже началась, FSR игнорируется до последнего разряда. Для пересылок пакетного режима FSR должен быть низким в течение последнего разряда, или начнется следующая пересылка. После приема полного слова и пересылки в DRR, генерируется RINT.

В режиме фиксированной скорости передачи данных непрерывные пересылки могут выполняться тогда, когда R/XFSM = 0, пока обеспечен соответствующий кадр синхронизации, или пока DXR перезагружается каждый цикл при внутренней генерации FSX.

При операциях приема с внешней генерацией FSX, после того, как передача началась, кадровые синхроимпульсы требуются только в течение последнего передаваемого разряда для инициализации следующей непрерывной передачи. В противном случае входы кадровых синхроимпульсов игнорируются. Следовательно, непрерывная передача происходит, если кадровый синхроимпульс удерживается в высоком уровне. При внутренней генерации FSX возникает задержка около 2,5 циклов CLKX от загрузки DXR до возникновения FSX. Эта задержка возникает каждый раз при загрузке DXR; следовательно, в течение непрерывной передачи команда, которая загружает DXR, должна быть выполнена на N-3 разрядов для N-разрядной передачи. Т. к. задержки из-за конвейера могут изменяться, необходимо определить жесткие пределы безопасности, принимая во внимание эту задержку.

После начала процесса в начале каждой пересылки генерируются RINT и XINT.

XINT указывает, что XSR был загружен из DXR и может быть использован для указания, что DXR должен быть перезагружен. Для обеспечения непрерывной передачи в режиме фиксированной скорости передачи с кадровыми синхроимпульсами, особенно при внутренней генерации FSX, DXR должен быть перезагружен в поступающей пересылке.

RINT показывает, что полное слово принято и переслано в DRR. Следовательно, RINT обычно используется для указания необходимости чтения из DRR.

Непрерывные пересылки прерываются путем прерывания кадровых синхроимпульсов или при внутренней генерации FSX, не перезагружая DXR.

Непрерывные пересылки последовательного порта могут быть завершены без использования кадровых синхроимпульсов, если R/XFSM установлен в 1. В этом режиме работа последовательного порта подобна работе с кадрами, за исключением того, что импульс кадровой синхронизации установлен только при передаче первого слова, но больше кадровые синхроимпульсы не используются. В течение передачи первого слова не генерируются внутренние синхроимпульсы и входные синхроимпульсы игнорируются. Кроме того, R/XFSM должен быть установлен до или в течение пересылки первого слова и должен быть установлен не позже, чем пересылается N-1 разряд первого слова, за исключением операций передачи. Для операций передачи в режиме фиксированной скорости передачи данных XFSM должен быть установлен не позднее передачи N-2 разряда. Очистка R/XFSM должна быть произведена не позднее, чем разряд N-1 должен быть распознан в текущем цикле.

Временные диаграммы RINT и XINT и пересылки данных в/из DXR и DRR, соответственно, такие же, как и в непрерывном режиме с фиксированной скоростью передачи данных с кадровыми синхроимпульсами.

В этом режиме работы также имеет место задержка в 2,5 цикла CLKX после загрузки DXR до генерации внутреннего FSX. Как и в случае работы в непрерывном режиме с фиксированной скоростью передачи данных, R/XFSM может быть очищен или установлен, как требуется, в том числе в течение действительных пересылок, для разрешения или запрещения использования кадровых синхроимпульсов, как диктуется системными требованиями. В зависимости от многих условий, эффект изменения состояния R/XFSM обнаруживается в течение пересылки, в которой производилось изменение R/XFSM, при условии, что изменение было сделано заранее в течение пересылки. Для операций передачи с внутренним FSX в режиме фиксированной скорости передачи данных, однако, имеет место задержка на одно слово до генерации кадрового синхроимпульса, возникающего при очистке XFSM в ноль. Следовательно, в этом случае передается одно дополнительное слово до генерации следующего FSX. Необходимо обратить также внимание на то, что как обсуждалось выше, очистка XFSM будет распознаваться в течение текущего слова, которое будет передаваться, если только XFSM очищен не позднее, чем на N-1 разряде. Установка XFSM распознается тогда, когда XFSM установлен не позднее N-2 разряда.

Временная диаграмма операции с переменной скоростью передачи данных

Операции с переменной скоростью передачи данных также поддерживают два типа работы: пакетный и непрерывный. Пакетный режим с переменной скоростью передачи данных подобен пакетному с фиксированной скоростью передачи данных. В режиме с переменной скоростью передачи данных, однако, FSX/FSR и синхронизация данных немного отличаются в начале и в конце пересылок. Три основных отличия между временной диаграммой с фиксированной и с переменной скоростями передачи данных:

- импульсы FSX/FSR продолжаются в течение определенного интервала пересылки, хотя FSR и внешний FSX игнорируются после пересылки первого разряда. Импульсы

FSX/FSR в режиме фиксированной скорости передачи данных обычно продолжаются в течение только одного цикла CLKX/CLKR, но могут продолжаться и более долго;

- пересылка данных начинается в течение цикла CLKX/CLKR, в котором возникает FSX/FSR, в отличие от цикла CLKX/CLKR, следующим за FSX/FSR, как в случае фиксированного периода данных;

- при переменной скорости передачи данных кадровые импульсы игнорируются до конца передачи последнего разряда, в отличие от начала передачи последнего разряда, как в случае фиксированного периода данных.

При непрерывной передаче в режиме переменной скорости передачи данных с кадровым синхроимпульсом, временная диаграмма такая же, как для режима фиксированной скорости передачи данных, за исключением различий между этими двумя режимами, которые описывались для пакетного режима. Единственное исключение состоит в том, что DXR должен быть загружен не позднее, чем на N-4 бите данных для обеспечения непрерывной операции в режиме переменной скорости передачи данных, не позднее N-3 для режима фиксированной скорости передачи данных.

Непрерывная операция в переменном режиме без кадровых синхроимпульсов также подобна непрерывной без кадровых синхроимпульсов в режиме фиксированной скорости передачи. Как и в переменном непрерывном режиме с кадровыми синхроимпульсами, DXR должен быть перезагружен не позднее, чем на N-4 бите данных для обеспечения выполнения непрерывной операции. Кроме того, когда R/XFSM установлен или очищен в переменном режиме, изменение должно быть произведено не позднее, чем на N-1 бите, чтобы повлиять на текущую пересылку.

8.2.11 Примеры использования интерфейса последовательного порта ПЦОС

При использовании режима квитирования передачи сигналами передачи и приема данных являются (FSX/DX/CLKX) и (FSR/DR/CLKR), соответственно. Другими словами, когда ПЦОС принимает данные только в режиме квитирования передачи, сигналы передачи также нужны для передачи сигнала подтверждения.

Так как при выборе режима квитирования передачи FSX установлен как выход, и непрерывный режим запрещен, разряды XFSM и RFSM должны быть установлены в 0, разряд FSXOUT в регистре глобального управления должен быть установлен в 1. Для начала связи в режиме квитирования передачи разряды XRESET, RRESET и HS должны быть установлены в 1. Рекомендуется, чтобы полярность выводов последовательного порта была с активным высоким уровнем для простоты. Хотя CLKX, CLKR могут быть установлены и как входы и как выходы, рекомендуется установить CLKX как выход, а CLKR – как вход.

Таймер последовательного порта необходим тогда, когда CLKX или CLKR установлены как выход. В вышеописанном случае, т. к. только CLKX установлен как выход, регистр управления таймера должен быть установлен в 0Fh. Когда используется таймер порта, регистр периода таймера также должен быть установлен в соответствующее значение для синхронизации скорости. Скорость тактирования таймера последовательного порта устанавливается аналогично таймеру ПЦОС (см. подраздел 8.1).

Максимальная частота синхронизации для последовательных пересылок равна F(CLKIN)/4, если используется внутренняя синхронизация и равна F(CLKIN)/5,2, если используется внешняя синхронизация. Следовательно, если два ПЦОС имеют общую синхронизацию, как в вышеописанном случае, регистр периода таймера должен быть установлен в значение, большее или равное 1, что устанавливает частоту синхронизации в F(CLKIN)/8.

Примеры установок регистров последовательного порта для вышеописанного случая приведены ниже (предполагается, что два ПЦОС имеют общую синхронизацию).

Установка 1:

Глобальное управление = 0EBC0064h ; 32-разрядный, фиксированный период
Управление передачи порта = 0111h ; данных, пакетный режим
Управление приема порта = 0111h ; FSX, CLKX (выходы) = F(CLKIN)/8
Управление таймером п/порта = 0Fh ; CLKR (вход), режим квитирования
Управление счетчиком таймером последовательного порта = 0h ; передачи, прерывания приема
Управление периодом таймером п/порта ≥ 01h ; и передачи разрешены.

Установка 2:

Глобальное управление = 0C000364h ; 8-разрядный, переменный период
Управление передачи порта = 0111h ; данных, пакетный режим
Управление приема порта = 0111h ; FSX, CLKX (выходы) = F(CLKIN)/24
Управление таймером п/порта = 0Fh ; CLKR (вход), режим квитирования
Управление счетчиком таймером п/порта = 0h ; передачи, прерывания приема
Управление периодом таймером п/порта ≥01h ; и передачи запрещены.

Так как данные имеют предшествующую 1 и сигнал подтверждения в виде 0 в режиме квитирования передачи, последовательный порт ПЦОС может отличать сигналы данных и подтверждения. Следовательно, если при приеме данных последовательный порт принимает данные до сигнала подтверждения, данные не могут быть неверно интерпретированы, как сигнал подтверждения, и потеряны. Кроме того, сигнал подтверждения не генерируется, пока данные читаются из регистра приема данных DRR. Следовательно, ПЦОС не будет передавать данные и сигнал подтверждения одновременно.

8.2.12 Инициализация/реконфигурация последовательного порта

Последовательный порт управляется через картированные в памяти регистры, адресуемые через соответствующую периферийную шину. Общая процедура инициализации и/или реконфигурации последовательного порта следующая:

- остановить последовательный порт, очистив разряды XRESET и/или RRESET регистра глобального управления порта. Чтобы сделать это, необходимо записать 0 в регистр глобального управления порта. Необходимо помнить, что порт останавливается по RESET#;

- сконфигурировать порт через регистр глобального управления порта (с XRESET = RRESET = 0) и регистры управления FSX/DX/CLKX и FSR/DR/CLKR. Если необходимо, определить регистры приема/передачи: управления таймера (с XHLD=RHLD = 0), счетчика таймера и периода таймера;

- запустить операцию последовательного порта с помощью установки разрядов XRESET и RRESET регистра глобального управления порта и разрядов XHLD и RHLD таймера приема/передачи последовательного порта, если требуется.

8.3 Контроллер ПДП

ПЦОС имеет внутрикристальный контроллер прямого доступа к памяти (ПДП), который обеспечивает потребности ЦПУ при выполнении функций ввода-вывода.

Контроллер ПДП может выполнять операции ввода-вывода без взаимодействия операциями ЦПУ. Следовательно, существует возможность взаимодействия с медленными устройствами (внешней памятью и периферией, такой как ЦАП/АЦП, последовательные порты и т. д.) без использования ЦПУ.

Пересылки ПДП состоят из двух операций: чтения из адреса памяти и запись по адресу памяти. Контроллер ПДП может читать из и записывать по любому адресу карты памяти ПЦОС, включая и всю картированную в памяти периферию. Работа ПДП управляется следующим набором картированных в памяти регистров:

- регистром глобального управления ПДП;
- регистром адреса источника ПДП;
- регистром адреса назначения ПДП;
- регистром счетчика пересылок ПДП.

8.3.1 Регистр глобального управления ПДП

Регистр глобального управления ПДП управляет состоянием работы контроллера ПДП. Этот регистр также показывает состояние ПДП, которое изменяется каждый цикл. Адреса источника и назначения могут инкрементироваться, декрементироваться и синхронизироваться, используя определенные разряды регистра глобального управления. При сбросе системы все разряды регистра управления ПДП устанавливаются в 0. В таблице 8.7 приведен список разрядов регистра, их имена и функции.

Таблица 8.7 – Разряды регистра глобального управления ПДП

Разряд	Имя	Значение при сбросе	Функция
1	2	3	4
0-1	START	0 0	Управляет состоянием, в котором ПДП стартует и останавливается. ПДП может быть остановлен без потери данных (см. таблицу 8.8).
2-3	STAT	0 0	Эти разряды показывают состояние ПДП и изменяются каждый цикл (см. таблицу 8.9).
4	INCSRC	0	Если INCSRC=1, адрес источника инкрементируется после каждого чтения.
5	DECSRC	0	Если DECSRC=1, адрес источника декрементируется после каждого чтения. Если DECSRC =INCSRC, адрес источника не изменяется после чтения.
6	INCDST	0	Если INCDST=1, адрес назначения инкрементируется после каждого чтения.
7	DECDST	0	Если DECDST=1, адрес назначения декрементируется после каждого чтения. Если DECDST=INCDST, адрес назначения не изменяется после чтения.
9-8	SYNC	0 0	Разряды SYNC определяют синхронизацию между событиями, начинающими пересылки источника и назначения. Интерпретация разрядов SYNC приведена в таблице 8.10.
10	TC	0	Разряд TC влияет на работу счетчика пересылок. Если TC=0, пересылки не прерываются, когда счетчик пересылок становится равным 0. Если TC=1, пересылки не прерываются, когда счетчик пересылок становится равным 0.

Окончание таблицы 8.7

1	2	3	4
11	TCINT	0	Если TCINT=1, устанавливается прерывание ПДП, когда счетчик пересылок переходит через 0. Если TCINT=1, не устанавливается прерывание ПДП, когда счетчик пересылок переходит через 0.
31-12	Зарезервированы	0 0	Считываются как 0.

Таблица 8.8 – START разряды для режимов ПДП

START	Функция
0 0	Циклы чтения или записи ПДП, находящиеся в работе, будут завершены; любое чтение данных будет игнорироваться. Любые незавершенные чтения/записи будут прерваны. ПДП сбрасывается в состояние момента старта, новая пересылка начинается; т. е. выполняется чтение (это значение устанавливается при сбросе).
0 1	Если чтение или запись начаты, они завершатся до остановки: например, в середине или в конце пересылок ПДП. Если не начиналось чтение или запись, они не начинаются.
1 0	Если пересылка ПДП началась, полная пересылка завершится (включая операции чтения и записи) до останова. Если пересылка не началась, она не начинается.
1 1	ПДП стартует со сброса или перестартует с предыдущего значения.

Таблица 8.9 – STAT разряды для режимов ПДП

STAT	Функция
0 0	ПДП остановлен между пересылками ПДП (между записью и чтением). Это значение устанавливается при сбросе.
0 1	ПДП остановлен в середине пересылки ПДП, т. е. между чтением и записью.
1 0	Зарезервировано.
1 1	ПДП занят, т. е. ПДП выполняет чтение или запись или ждет прерывания синхронизации источника или назначения.

Таблица 8.10 – SYNC разряды для режимов ПДП

SYNC	Функция
0 0	Нет синхронизации. Разрешенные прерывания игнорируются (значение при сбросе).
0 1	Синхронизация источника. Чтение выполняется при возникновении разрешенного прерывания.
1 0	Синхронизация назначения. Запись выполняется при возникновении разрешенного прерывания.
1 1	Синхронизация источника и назначения. Чтение выполняется при возникновении разрешенного прерывания. Запись выполняется при возникновении следующего разрешенного прерывания.

8.3.2 Регистры адреса источника и назначения

Регистры адреса источника и назначения – 24-разрядные регистры, содержание которых определяет адреса источника и назначения. Как определено управляющими разрядами DECSRC, INCSRC, DECDST и INCDST регистра глобального управления, эти регистры инкрементируются или декрементируются в конце соответствующего обращения к памяти регистра источника для чтения и регистра назначения для записи. При сбросе системы в эти регистры записывается 0.

8.3.3 Регистр счетчика пересылок

Регистр счетчика пересылок – 24-разрядный регистр, управляемый 24-разрядным счетчиком, который считает в обратную сторону. Счетчик уменьшается в начале записи памяти ПДП. Таким образом, он может быть использован для управления размером блока передаваемых данных. Регистр счетчика пересылок устанавливается в 0 при системном сбросе. Когда разряд TCINT регистра глобального управления установлен, регистр счетчика пересылок будет определять установку флага прерывания ПДП по установке счетчика в 0.

8.3.4 Регистр разрешения прерываний ЦПУ/ПДП

Регистр разрешения прерываний ЦПУ/ПДП (IE) – 32-разрядный регистр, размещенный в регистровом файле ЦПУ. Разряды разрешения прерываний ЦПУ/ПДП размещаются в адресах 10-1. Разряды разрешения прерываний ПДП размещаются в адресах 26-16. 1 в разряде регистра разрешения прерываний ЦПУ/ПДП разрешает соответствующее прерывание, 0 запрещает соответствующее прерывание. При сбросе в этот регистр записывается 0.

В таблице 8.11 приведен список разрядов, имен и функций регистра разрешения прерываний ЦПУ/ПДП. Приоритеты и декодирование схем прерываний ЦПУ и ПДП идентичны. Следует помнить, что когда ПДП получает прерывание, это прерывание запускается в зависимости от поля SYNC регистра управления ПДП. Также, необходимо обратить внимание, что прерывания могут влиять на ПДП, но не на ЦПУ и наоборот. См. 3.1.8 и раздел 6.

Таблица 8.11 – Разряды регистра разрешения прерываний ЦПУ/ПДП

Разряд	Имя	Функция
0	EINT0	Разрешает внешнее прерывание 0 (ЦПУ)
1	EINT1	Разрешает внешнее прерывание 1 (ЦПУ)
2	EINT2	Разрешает внешнее прерывание 2 (ЦПУ)
3	EINT3	Разрешает внешнее прерывание 3 (ЦПУ)
4	EXINT0	Разрешает прерывание передачи п/порта 0 (ЦПУ)
5	ERINT0	Разрешает прерывание приема п/порта 0 (ЦПУ)
6	EXINT1	Разрешает прерывание передачи п/порта 1 (ЦПУ)
7	ERINT1	Разрешает прерывание приема п/порта 1 (ЦПУ)
8	ETINT0	Разрешает прерывание таймера 0 (ЦПУ)
9	ETINT1	Разрешает прерывание таймера 1 (ЦПУ)
10	EDINT	Разрешает прерывание контроллера ПДП (ЦПУ)
11-15	Зарезервированы	Считываются как 0
16	EINT0	Разрешает внешнее прерывание 0 (ПДП)
17	EINT1	Разрешает внешнее прерывание 1 (ПДП)
18	EINT2	Разрешает внешнее прерывание 2 (ПДП)
19	EINT3	Разрешает внешнее прерывание 3 (ПДП)
20	EXINT0	Разрешает прерывание передачи п/порта 0 (ПДП)
21	ERINT0	Разрешает прерывание приема п/порта 0 (ПДП)
22	EXINT1	Разрешает прерывание передачи п/порта 1 (ПДП)
23	ERINT1	Разрешает прерывание приема п/порта 1 (ПДП)
24	ETINT0	Разрешает прерывание таймера 0 (ПДП)
25	ETINT1	Разрешает прерывание таймера 1 (ПДП)
26	EDINT	Разрешает прерывание контроллера ПДП (ПДП)
27-31	Зарезервированы	Считываются как 0

8.3.5 Операция пересылки памяти ПДП

Каждая операция пересылки памяти ПДП состоит из двух частей:

- чтение данных по адресу, определенному в регистре источника ПДП;
- запись считанных данных по адресу, определенному регистром назначения ПДП.

Циклы	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Внутренний начальный адрес	R		R		R													
Внутренний конечный адрес		W		W		W												
Начальный адрес – основная шина	R.	R.	R.	I		R.	R.	R.	I		R.	R.	R .	I				
Внутренний конечный адрес		C_r					C_r					C_r						
				W						W					W			
Начальный адрес – расширенная шина	R.	R.	R.	I		R.	R.	R.	I		R.	R.	R.	I				
Внутренний конечный адрес		C_r					C_r					C_r						
				W						W					W			

Начальный адрес	Конечный адрес: внутренний
Внутренний	(1+1)T
Основная шина	[(2+Cr)+1]T
Расширенная шина	[(2+Cr)+1]T

Принятые условные обозначения:

- T – число передач;
- C_r – состояние ожидания чтения источника;
- R – одноцикловые чтения;
- W – одноцикловые записи;
- R.R – многоцикловые чтения;
- W.W – многоцикловые записи;
- I – цикл внутреннего регистра.

Рисунок 8.5 – Временная диаграмма и число циклов передачи данных ПДП во внутренний адрес

Пересылка завершена только после выполнения чтения и записи. Пересылка может быть остановлена установкой разрядов START в требуемое значение. При перезапуске ПДП (STAT = 11), он завершает любую ожидаемую пересылку.

Циклы	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Внутренний начальный адрес	R		R				R											
Конечный адрес – основная шина		W.	W.	W.	W	W.	W.	W.	W	W.	W.	W.	W					
				C_r				C_r				C_r						
Начальный адрес – основная шина	R.	R.	R	I					R.	R.	R	I						
		C_r								C_r								
Конечный адрес – основная шина					W.	W.	W.	W					W.	W.	W.	W		
							C_w								C_w			
Начальный адрес – расширенная шина	R.	R.	R	I		R.	R.	R	I		R.	R.	R	I				
		C_r					C_r					C_r						
Конечный адрес – основная шина					W.	W.	W.	W		W.	W.	W.	W		W.	W.	W.	W
							C_w					C_w					C_w	

Начальный адрес	Конечный адрес: основная шина
Внутренний	$1+(2+C_w)T$
Основная шина	$(2+C_r+2+C_w)\ T$
Расширенная шина	$(2+C_r+2+C_w)+[(2+C_w+\max(0,C_r-C_w+1)](T-1)$

Принятые условные обозначения:

- T – число передач;
- C_r – состояние ожидания чтения источника;
- R – одноцикловые чтения;
- W – одноцикловые записи;
- R.R – многоцикловые чтения;
- W.W – многоцикловые записи;
- I – цикл внутреннего регистра.

Рисунок 8.6 – Временная диаграмма и число циклов передачи данных ПДП на основную шину

В конце чтения ПДП адрес источника изменяется согласно значению в разрядах SRCINC и SRCDEC регистра глобального управления ПДП. В конце записи ПДП адрес назначения изменяется согласно значению в разрядах DSTINC и DSTDEC регистра глобального управления ПДП. По окончании каждой записи ПДП счетчик пересылок ПДП уменьшается.

Запись и чтение внутри кристалла (записи и чтения внутрикристальной памяти и периферии) совершаются за один цикл. Чтение ПДП внешней памяти совершается за два цикла. Первый цикл – внешнее чтение, второй цикл загружает регистр ПДП. Внешний цикл чтения ПДП аналогичен внешнему циклу чтения ЦПУ. Запись ПДП во внешнюю память аналогична записи ЦПУ во внешнюю память. Если ПДП запущен и передает данные через одну из внешних шин, регистр управления шиной, связанный с данной шиной, не должен

изменяться. Если регистр управления шиной (см. раздел 7) должен быть изменен, необходимо остановить ПДП, произвести изменение и перезапустить ПДП. Если это сделать неверно, в результате может возникнуть непредвиденный доступ шины с нулевым состоянием ожидания.

Посредством 24-разрядных регистров источника и назначения ПДП может адресовать любой картированный в памяти адрес карты памяти ПЦОС. Рисунки 8.5 и 8.6 иллюстрируют временные диаграммы пересылок ПДП. |R| и |W| – одноцикловые чтение и запись, соответственно |R.R| и |W.W| – многоцикловые чтение и запись. $|C_r|$ и $|C_w|$ показывают число циклов ожидания для чтения и записи.

В таблице 8.12 приведены максимальные значения скорости пересылок при отсутствии состояний ожидания ($C_r = C_w = 0$). В таблице 8.13 приведены максимальные значения скорости пересылок при наличии одного состояния ожидания для чтения ($C_r = 1$) и отсутствии состояний ожидания для записи ($C_w = 0$). В таблице 8.14 приведены максимальные значения скорости пересылок при наличии одного состояния ожидания для чтения ($C_r = 1$) и одного состояния ожидания для записи ($C_w = 1$).

В каждой таблице рассматривается время, необходимое для полной пересылки ПДП (чтение и запись). Т. к. требуется одно обращение к шине для чтения и одно для записи, скорость внутренних передач по шине вдвое больше скорости передач ПДП.

Предполагается также, что отсутствуют конфликты с ЦПУ.

Таблица 8.12 – Максимальные скорости пересылок, когда $C_r = C_w = 0$

Источник	Назначение			Единица измерения
	Внутренняя	Основная	Расширенная	
Внутренняя	40,0	40,0	40,0	Мбайт/с
Основная	26,7	20,0	26,7	Мбайт/с
Расширенная	26,7	26,7	20,0	Мбайт/с

Таблица 8.13 – Максимальные скорости пересылок, когда $C_r = 1$, $C_w = 0$

Источник	Назначение			Единица измерения
	Внутренняя	Основная	Расширенная	
Внутренняя	40	40	40	Мбайт/с
Основная	20	16	20	Мбайт/с
Расширенная	20	20	16	Мбайт/с

Таблица 8.14 – Максимальные скорости пересылок, когда $C_r = 1$, $C_w = 1$

Источник	Назначение			Единица измерения
	Внутренняя	Основная	Расширенная	
Внутренняя	40	26,7	26,7	Мбайт/с
Основная	20	13,3	20,0	Мбайт/с
Расширенная	20	20,0	13,3	Мбайт/с

8.3.6 Синхронизация каналов ПДП

Канал ПДП может быть синхронизирован через использование прерываний. Смотри таблицу 8.10 для выяснения соотношения между разрядами SYNC регистра глобального управления ПДП и выполняемой синхронизацией. В данном разделе обсуждаются следующие четыре механизма синхронизации:

- отсутствие синхронизации (SYNC = 0 0);
- синхронизация источника (SYNC = 0 1);
- синхронизация назначения (SYNC = 1 0);
- синхронизация источника и назначения (SYNC = 1 1).

Отсутствие синхронизации

Когда SYNC = 0 0, не выполняется синхронизация. ПДП выполняет чтение и запись всякий раз при отсутствии конфликтов. Все прерывания игнорируются и, следовательно, должны быть глобально запрещены. Однако разряды регистра разрешения прерываний ПДП не изменяются.

Синхронизация источника

Когда SYNC = 0 1, ПДП синхронизирован по источнику. Чтение не будет производиться до того, как ПДП получит прерывание. Тогда все прерывания ПДП глобально запрещаются. Однако разряды регистра разрешения прерываний ПДП не изменяются.

Синхронизация по назначению

Когда SYNC = 1 0, ПДП синхронизирован по назначению. Сначала все прерывания игнорируются до окончания чтения. Предполагается, что все прерывания ПДП глобально запрещены, однако, разряды регистра разрешения прерываний ПДП не изменяются. Запись не будет производиться до того, как ПДП получит прерывание.

Синхронизация источника и назначения

Когда SYNC = 1 1, ПДП синхронизирован как по источнику, так и по назначению. Чтение выполняется по получении прерывания. Запись выполняется по получении следующего прерывания.

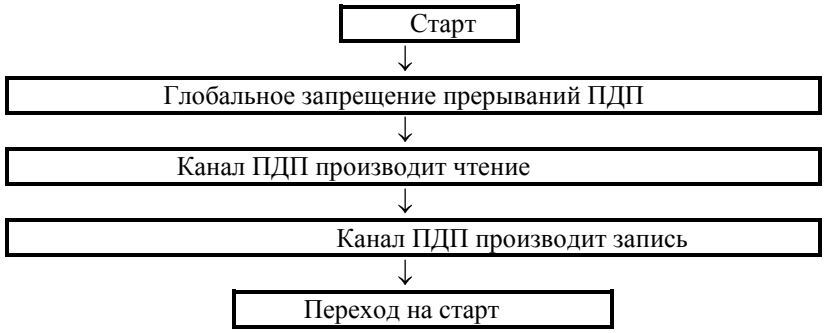


Рисунок 8.7 – Отсутствие синхронизации ПДП

Рисунок 8.8 – Синхронизация источника

Рисунок 8.9 – Синхронизация по назначению

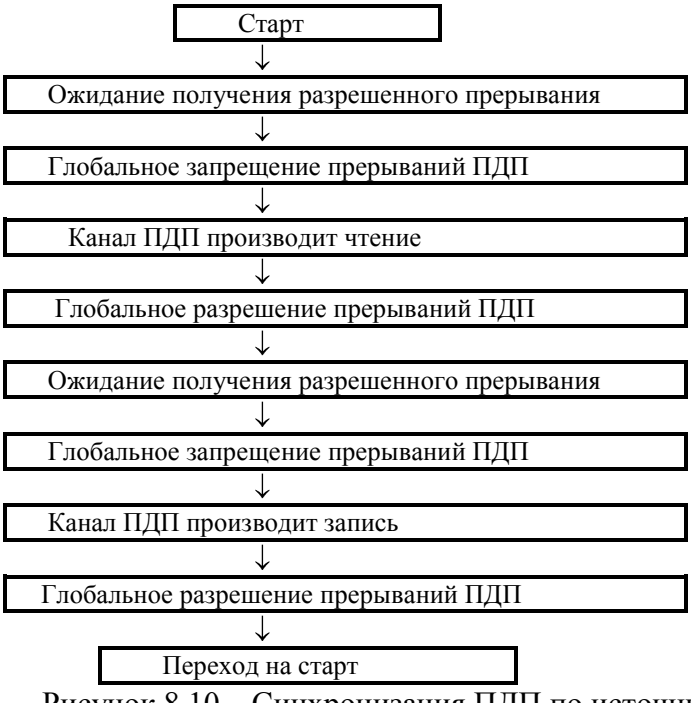


Рисунок 8.10 – Синхронизация ПДП по источнику и назначению

8.3.7 Прерывания ПДП

Прерывания ПДП в ЦПУ могут генерироваться всякий раз, когда счетчик пересылок достигает нуля, показывая, что имеет место последняя пересылка. Разряд TCINT регистра глобального управления ПДП определяет, будет ли генерироваться прерывание. Если TCINT = 1, вырабатывается прерывание ПДП. Если TCINT = 0, прерывание не вырабатывается. Если вырабатываются прерывания ПДП, то 10 разряд в регистре разрешения прерываний EDINT должен быть установлен для разрешения прерывания ЦПУ из ПДП.

Второй разряд регистра глобального управления TC также связан с состоянием разряда TCINT и операцией прерывания. Разряд TC определяет, будут ли пересылки прерываться при равенстве нулю счетчика пересылок, или они будут продолжаться. Если TC = 1, пересылки будут прерываться при равенстве нулю счетчика пересылок. Если TC = 0, пересылки не будут прерываться при равенстве нулю счетчика пересылок.

В основном, если TCINT = 0, TC также должен быть установлен в 0. В противном случае, пересылка ПДП будет прервана, а ЦПУ не будет оповещен. Если TCINT=1, TC в большинстве случаев также должен быть 1. В этом случае ЦПУ будет оповещен при завершении пересылки, и ПДП будет остановлен и готов начать новую пересылку.

8.3.8 Примеры установки и использования ПДП

- Переслать блок из 256 слов из внешней во внутреннюю память и генерировать прерывание по окончании. Порядок использования памяти при выполнении.

Адрес источника ПДП: 800000h

Адрес назначения ПДП: 809800h

Счетчик пересылок ПДП: 00000100h

Глобальное управление ПДП: 00000C53h

Разрешение прерываний ЦПУ/ПДП (IE): 00000400h

- Переслать блок из 128 слов из внутренней во внешнюю память и генерировать прерывание по окончании. Порядок использования памяти должен быть инвертирован; таким образом, старший адресуемый элемент блока становится младшим.

Адрес источника ПДП: 809800h

Адрес назначения ПДП: 800000h

Счетчик пересылок ПДП: 00000080h

Глобальное управление ПДП: 00000C93h

Разрешение прерываний ЦПУ/ПДП (IE): 00000400h

- Переслать блок из 200 слов из регистра приема последовательного порта 0 во внутреннюю память и генерировать прерывание по окончании. Пересылка должна быть синхронизирована с прерыванием приема последовательного порта 0.

Адрес источника ПДП: 80804Ch

Адрес назначения ПДП: 809C00h

Счетчик пересылок ПДП: 000000C8h

Глобальное управление ПДП: 00000D43h

Разрешение прерываний ЦПУ/ПДП (IE): 00200400h

- Переслать блок из 200 слов из внешней памяти в регистр передачи последовательного порта 0 и генерировать прерывание по окончании. Пересылка должна быть синхронизирована с прерыванием передачи последовательного порта 0.

Адрес источника ПДП: 809C00h

Адрес назначения ПДП: 808048h

Счетчик пересылок ПДП: 000000C8h

Глобальное управление ПДП: 00000E13h

Разрешение прерываний ЦПУ/ПДП (IE): 00400400h

- Пересылать данные непрерывно между приемным регистром последовательного порта 0 и регистром передачи последовательного порта 0 для создания цифровой обратной связи. Пересылка должна быть синхронизирована с прерыванием приема и передачи последовательного порта 0.

Адрес источника ПДП: 80804Ch

Адрес назначения ПДП: 808048h

Счетчик пересылок ПДП: 00000000h

Глобальное управление ПДП: 00000303h

Разрешение прерываний ЦПУ/ПДП (IE): 00600000h

8.3.9 Инициализация/реконфигурация ПДП

ПДП управляется через картированные в памяти регистры, расположенные на соответствующей периферийной шине. Ниже дана общая процедура инициализации и/или реконфигурации ПДП:

- остановить ПДП, очистив разряд START регистра глобального управления ПДП. Это может быть сделано при записи 0 в регистр глобального управления ПДП. ПДП останавливается при сбросе RESET#;

- определить ПДП через регистр глобального управления ПДП (со START=00), а также регистры источника, назначения счетчика пересылок, если необходимо, см. 8.3.8;

- запустить ПДП, установив разряды START регистра глобального управления ПДП как требуется.

9 Работа конвейера

Две характеристики, которые вносят вклад в высокую производительность ПЦОС: конвейеризация и параллельное выполнение операций ЦПУ и ввода-вывода.

Пять функциональных элементов управляют работой ПЦОС: выборка, декодирование, чтение, выполнение и ПДП. Конвейеризация – это перекрытие или параллельное выполнение уровней основной команды: выборки, декодирования, чтения и выполнения.

Выполняя операции ввода-вывода, контроллер ПДП освобождает ЦПУ от выполнения этих операций, снижая нагрузку конвейера и увеличивая вычислительную мощность.

Основные понятия, обсуждаемые в данном разделе:

- структура конвейера (подраздел 9.1);
- конфликты конвейера (подраздел 9.2);
- конфликты переходов (9.2.1);
- конфликты памяти (9.2.3);
- разрешение конфликтов регистров (подраздел 9.3);
- разрешение конфликтов памяти (подраздел 9.4);
- синхронизация обращений к памяти (подраздел 9.5);
- выборки программ;
- загрузка и сохранение данных;
- обращения ПДП.

9.1 Структура конвейера

Пять функциональных элементов структуры конвейера ПЦОС и их функции следующие:

- выборка (F): выбирается слово команды из памяти и изменяется счетчик команд (PC);
- декодирование (D): декодируется слово команды и выполняется генерация адреса. Также управляет любыми изменениями вспомогательных регистров и указателем стека;
- чтение (R): если требуется, считывается операнд из памяти;
- выполнение (E): если требуется, считывается операнд из регистрового файла, выполняется необходимая операция и записывается результат в регистровый файл. Если требуется, результат предыдущей операции записывается в память;
- канал прямого доступа к памяти (ПДП): чтение и запись памяти.

Базовая команда имеет четыре уровня выполнения: выборка, декодирование, чтение и выполнение. Уровни индексированы в соответствии с циклом команды и выполнения. Полное перекрытие в конвейере, где все четыре элемента работают параллельно, возникает на цикле (m). Те уровни, которые должны быть почти выполнены на m+1, уже выполнены на уровне m-1. Управление конвейером обеспечивает высокоскоростное выполнение (E) за один цикл, при этом конфликты конвейера прозрачны для пользователя. Нет необходимости предпринимать какие-либо меры для гарантии корректного выполнения операции.

Цикл	F	D	R	E	
m-3	W	-	-	-	
m-2	X	W	-	-	
m-1	Y	X	W	-	
m	Z	Y	X	W	← полное перекрытие
m+1	-	Z	Y	X	
m+2	-	-	Z	Y	
m+3	-	-	-	Z	

Примечания

1 W, X, Y, Z – представляют команды.

2 F, D, R, E – выборка, декодирование, чтение и выполнение, соответственно.

Рисунок 9.1 – Структура конвейера ПЦОС

Приоритеты от старшего к младшему, присвоенные для каждого из функциональных элементов, следующие:

- E – выполнение (высший),
- R – чтение,
- D – декодирование,
- F – выборка,
- ПДП – (низший).

Когда процесс обработки команды готов к переходу на следующий высший уровень конвейера, но этот уровень не готов к новому входу, возникает конфликт конвейера. В этом случае элемент нижнего уровня ждет, пока устройство с высшим приоритетом завершит свою текущую функцию.

Несмотря на низкий приоритет контроллера ПДП, конфликты с ЦПУ могут быть минимизированы путем соответствующего структурирования данных, потому что контроллер ПДП имеет собственную шину данных и адреса.

9.2 Конфликты конвейера

Конфликты конвейера ПЦОС могут быть сгруппированы в следующие три категории:

- конфликты переходов: включают большинство команд или операций, которые читают и/или изменяют PC;
- конфликты регистров: включают задержки, которые могут возникнуть при чтении и записи в регистры, которые используются для генерации адреса;
- конфликты памяти: возникают, когда внутренние устройства ПЦОС находятся в состязании за ресурсы памяти.

Каждый из этих трех типов обсуждается далее. Примеры прилагаются. Необходимо обратить внимание в этих примерах, что когда данные перевыбираются или операция повторяется, символ, представляющий стадию конвейера, дополняется номером. Например, если выборка выполняется снова, мнемоника команды повторяется. Когда обращение задерживает несколько циклов из-за отсутствия готовности, используются символы RDY# и RDY для указания отсутствия или наличия готовности, соответственно.

9.2.1 Конфликты переходов

Первый тип конфликтов конвейера возникает при стандартных (незадержанных) переходах, т. е. BR, Bcond, DBcond, CALL, IDLE, RPTB, RPTS, RETIcond, RETScond, прерываниях и сбросе. Конфликты возникают при этих командах и операциях, т. к. в течение их исполнения конвейер используется только для завершения операции; другая информация, выбранная в конвейер, отбрасывается или перевыбирается или конвейер неактивен. Это называется «промывкой» конвейера. «Промывка» конвейера необходима в этих случаях для гарантии, что команды не будут выполнены по частям.

TRAPcond и CALLcond классифицируются отдельно от других типов переходов и рассматриваются позже.

В примере 9.1 приведена программа и работа конвейера для стандартного перехода. Обратите внимание, что выполняется одна пустая выборка (MPYF) и тогда после того, как доступен адрес перехода, выполняется новая выборка (команда OR). Эта пустая выборка влияет на кэш.

Пример 9.1 – Стандартное ветвление

```
   BR THREE      ; Безусловный переход
   MPYF          ; Не выполняется
   ADD           ; Не выполняется
   SUBF          ; Не выполняется
   AND           ; Не выполняется
   .
   .
THREE OR         ; Выбрана после выборки BR
   STI
   .
   .
```

Работа конвейера:

PC	F	D	R	E
n	BR	-	-	-
n+1	MPYF	BR	-	-
n+1	(nop)	(nop)	BR	-
n+1	(nop)	(nop)	(nop)	BR
THREE	OR	(nop)	(nop)	(nop)
	STI	OR	(nop)	(nop)

THREE → PC Выборка задержана для нового значения PC.

Обе команды RPTS и RPTB очищают конвейер, обеспечивая возможность загрузки регистров RS, RE и RC в соответствующее время относительно хода конвейера. Если эти регистры загружены без использования RPTS и RPTB, не возникает "промывка" конвейера. Если не используется ни один из режимов повторения, RS, RE и RC могут быть использова-

ны как 32-разрядные регистры общего назначения без возникновения конфликтов конвейера. В таких случаях, как вложение RPTB, обусловленное вложением прерываний, может быть, необходимо загрузить и сохранить эти регистры прямо во время использования режима повторения. Т. к. до четырех команд может быть выбрано до введения режима повторения, загрузки должны предшествовать для очистки конвейера. Если RC изменяется при загрузке его командой, прямая загрузка имеет приоритет над изменением, производимым логикой режима повторений.

Задержанные переходы вводятся для гарантии выборки следующих трех команд.

Задержанные переходы включают BRD, BcondD и DBcondD. В примере 9.2 приведены программа и работа конвейера для задержанного перехода.

Пример 9.2 – Задержанный переход

```
    BRD THREE    ; Безусловный задержанный переход
    MPYF         ; Выполняется
    ADD          ; Выполняется
    SUBF         ; Выполняется
    AND          ; Не выполняется
    .
    .
    .
THREE MPYF       ; Выбирается после выборки SUBF.
    .
    .
    .
```

Работа конвейера:

PC	F	D	R	E	
n	BRD	-	-	-	
n+1	MPYF	BRD	-	-	Не выполняется задержка
n+2	ADDF	MPYF	BRD	-	
n+3	SUBF	ADDF	MPYF	BRD	
THREE	MPYF	SUBF	ADDF	MPYF	

THREE → PC

9.2.2 Конфликты регистров

Конфликты регистров представляют собой чтение или запись регистров, использованных для адресации. Эти конфликты возникают, когда соответствующий регистр не готов для использования. Некоторые условия, при которых можно избежать конфликтов регистров, обсуждаются в подразделе 9.3

Регистры образуют три функциональные группы:

- группа 1: вспомогательные регистры (AR0 - AR7), индексные регистры (IR0, IR1) и регистр размера блока (BK);

- группа 2: указатель страницы данных (DP);

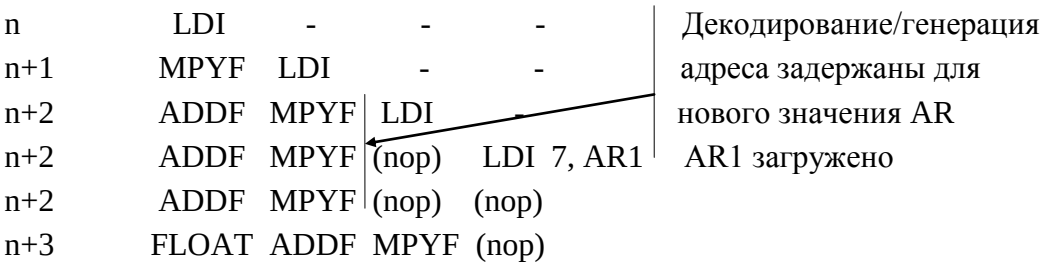
- группа 3: указатель системного стека (SP).

Если команда производит запись в одну из этих групп, элемент декодирования не может использовать любой регистр внутри группы до завершения записи, т. е. до завершения выполнения команды. В примере 9.3 вспомогательный регистр загружается, а другой вспомогательный регистр используется для следующей команды. Т. к. стадия декодирования нуждается в результате записи во вспомогательный регистр, декодирование этой второй команды задерживается на два цикла. Каждый раз при задержке декодирования выполняется перевыборка слова программы; т. е. ADDF выбирается три раза. Т. к. это действительные перевыборки, они могут привести не только к конфликтам с контроллером ПДП, но и к кэш-попаданию и кэш-отсутствию.

Пример 9.3 – Запись в AR с последующей генерацией адреса AR

```
        LDI 7, AR1          ; 7 → AR1
NEXT  MPYF *AR2, R0    ; Декодирование задержано на 2 цикла
        ADDF
        FLOAT
```

Работа конвейера:



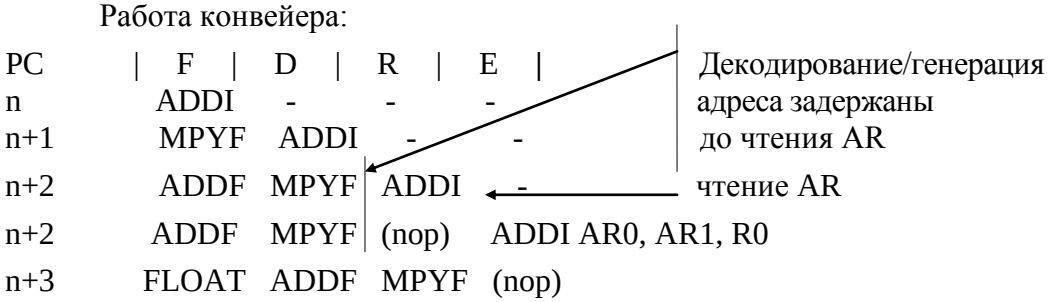
Случай чтения для этих групп аналогичен записи. Если команда должна считать значение в одном из элементов группы, использование той же группы при декодировании следующей команды задерживается до завершения чтения. Регистры считываются в начале цикла исполнения, поэтому требуется только один цикл задержки последующего декодирования. Для регистров IR0, IR1, BK или DP задержка не возникает. Для всех остальных, включая SP, задержка возникает.

В примере 9.4 производится сложение содержимого двух вспомогательных регистров с записью в регистр расширенной точности. Следующая команда использует другой вспомогательный регистр как адресный.

Пример 9.4. Чтение AR с последующим использованием AR для генерации адреса

```
        ADDI   AR0, AR1, R1          ; AR0 + AR1 → R1
NEXT   MPYF   *++AR2, R0           ; Декодирование, задержанное на 1 цикл
        ADDF
        FLOAT
```

Работа конвейера:



Команды DBR (декремент и ветвление) используют вспомогательные регистры в качестве счетчиков цикла, что аналогично использованию их для адресации. Следовательно, операции, приведенные в вышеописанных примерах, также присутствуют и при выполнении этих команд.

9.2.3 Конфликты памяти

Конфликты памяти могут возникать при превышении границы физической памяти. Например, блоки 0 и 1 ОЗУ могут поддерживать только 2 обращения за один цикл. Внешний интерфейс может поддерживать только одно обращение каждый цикл. Некоторые условия, при которых можно избежать конфликтов памяти, обсуждаются в подразделе 9.4.

Конфликты конвейера при работе с памятью состоят из следующих четырех типов:

- «ожидание программы» – выборка программы предотвращена с начала;
- «выборка программы не завершена» – выборка программы начата, но еще не завершена;
- «только выполнение» – последовательность команды требует трех обращений ЦПУ к данным за один цикл;
- «задержание всего» – операция основной или расширенной шины должна быть завершена до начала любой другой.

Эти типы конфликтов памяти обсуждены и проиллюстрированы примерами ниже.

«Ожидание программы»

Два условия могут предотвратить выборку программы.

1 Начало доступа ЦПУ к данным когда:

- происходит два обращения ЦПУ к внутренним блокам ОЗУ или ПЗУ и необходима выборка программы из того же блока;
- один из внешних портов начинает доступ к данным ЦПУ и необходима выборка программы из того же порта.

2 Требуется многоцикловое обращение ЦПУ или ПДП к данным через внешнюю шину.

В примере 9.5 иллюстрируется ожидание завершения обращения ЦПУ к данным.

В этом случае *AR0 и *AR1 оба указывают на блок 0 ОЗУ, и команда MPYF также выбирает из блока 0 ОЗУ. Это приводит к конфликту. Т. к. может быть произведено не более двух обращений к блоку 0 ОЗУ за один цикл, выборка программы не может быть начата и должна ждать завершения доступа ЦПУ к данным.

Пример 9.5 – Программа ожидает завершения обращения ЦПУ к данным

```
ADDF3 *AR0, *AR1, R0
FIX
MPYF
ADDF3
NEGB
```

Работа конвейера:

```
PC       |  F   |   D   |   R   |   E   |
n          ADDF3  -       -       -
n+1         FIX   ADDF3   -       -
n+2       (WAIT)  FIX     ADDF3   -
n+2        MPYF   (nop)   FIX     ADDF3   *AR0, AR1, R0
n+3        ADDF3  MPYF    (nop)   FIX
n+4        NEGB   ADDF3   MPYF    (nop)
```

Задержка выборки пока читаются AR

Чтение AR

В примере 9.6 приведено программное ожидание, связанное с многоцикловым обращением данные-данные или обращением ПДП. ADDF, MPYF и SUBF выбраны из некой области памяти иначе, чем требуется для внешнего порта ПДП. ПДП начинает многоцикловое обращение. Выборка программы, относящаяся к CALL, делается по тому же внешнему порту, который использует ПДП.

Эту ситуацию может вызвать одна из 2 причин:

1 Пересечение одной из 2 границ памяти:

- от 7FFFFFh до 800000h;
- от 809FFFh до 80A000h.

2 Выполняемый код был кэширован и команда до ADDF одна из следующих (условная или безусловная): команда задержанного ветвления или команда задержанного декремента и ветвления.

Хотя ПДП имеет самый низкий приоритет, многоцикловое обращение не может быть прервано. Выборка программы должна ожидать окончания обращения ПДП.

Пример 9.6 – Ожидание программы в связи с многоцикловым обращением

Работа конвейера:

```
PC       |  F   |   D   |   R   |   E   |
n          ADDF   -       -       -
n+1        MPYF   ADDF    -       -
n+2        SUBF   MPYF    ADDF    -
n+3       (WAIT)  SUBF    MPYF    ADDF
n+3        CALL   (nop)   SUBF    MPYF
n+4         -     CALL    (nop)   SUBF
```

двухцикловое обращение ПДП

«Выборка программы не завершена»

Незавершение выборки программы возникает, когда выборка программы требует более одного цикла для завершения из-за состояний ожидания. В примере 9.7 MPYF и конфликт – выборка через границу банка в основном порте, см. подраздел 7.4.

Пример 9.7 – Многоцикловые выборки программной памяти

Работа конвейера:

```
PC            |   F   |   D   |   R   |   E   |
n              MPYF     -       -       -
n+1            ADDF    MPYF     -       -
n+2   RDY      SUBF    ADDF    MPYF     -          ┐ требуется 1 состояние
n+2   RDY      SUBF    (nop)   ADDF    MPYF       ↓ ожидания
n+3            ADDI    SUBF    (nop)   ADDF
```

Тип конфликта конвейера «Только выполнение»

Тип конфликта конвейера «только выполнение» возникает, когда последовательность команд требует три обращения ЦПУ к данным в один цикл или при выполнении блокированной загрузки. Три случая, при которых возникает данный конфликт:

- команда выполняет сохранение и затем следует команда, выполняющая два чтения памяти;

- команда выполняет два сохранения и затем следует команда, выполняющая, по меньшей мере, одно чтение памяти;

- выполняется команда загрузки с блокировкой (LDII или LDFI) и XF= 1.

Т. к. эта последовательность требует трех обращений к памяти данных, а поддерживаются только два, выполняется только фаза исполнения конвейера. Двойные чтения, требуемые LDF||LDF, задерживаются на один цикл. Обратите внимание, что может возникнуть перевыборка следующей команды.

Пример 9.8 – Два чтения следуют за одиночным сохранением

```
STF  R0, *AR1    ; R0 → *AR1
LDF  *AR2, R1    ; *AR2 → R1  параллельно с
|| LDF  *AR3, R2  ; *AR3 → R2
```

Работа конвейера:

```
PC            |   F   |   D   |   R   |   E   |
n              STF      -       -       -
n+1         LDF||LDF    STF     -       -
n+2            W    LDF||LDF   STF     -          Запись должна завершиться
n+3            X        W    LDF||LDF  STF R0, *AR1  до того, как смогут
n+4            X        W    LDF||LDF  (nop)         завершиться два чтения
n+4            Y        X        W     LDF||LDF *AR2, R1 and *AR3,R2
```

В примере 9.9 приводится параллельное сохранение, за которым следует одна загрузка или чтение. Т. к. требуются два параллельных сохранения, следующее чтение данных ЦПУ должно ожидать один цикл до начала. Может иметь место одна перевыборка памяти.

Пример 9.9 – Одно чтение следует за параллельным сохранением

```
STF  R0, *AR0  ; R0 → *AR0  параллельно с
|| STF  R2, *AR1  ; R2 → *AR1
ADDF @SUM,R1  ; R1 + @SUM → R1
IACK
ASH
```

Работа конвейера:

PC	F	D	R	E	
n	STF||STF	-	-	-	Чтение должно ожидать пока
n+1	ADDF	STF||STF	-	-	завершатся записи
n+2	IACK	ADDF	STF||STF	-	
n+3	ASH	IACK	ADDF	STF||STF	R0, *AR0 and R2, *AR1
n+4	ASH	IACK	ADDF	(nop)	
n+4	-	ASH	IACK	ADDF	

Последний случай представляет команды загрузки с блокировкой (LDII или LDFI) и XF1 = 1. Т. к. загрузки с блокировкой используют вывод XF1 для подтверждения того, что чтение завершено, они могут потребовать расширения цикла чтения, как показано в примере 9.10. Следует помнить, что может возникнуть перевыборка программы.

Пример 9.10 – Загрузка с блокировкой

```
NOT   R1, R0
LDII   300h, AR2
ADII   *AR2, R2
CMPI   R0, R2
```

Работа конвейера:

PC	F	D	R	E	
n	NOT	-	-	-	
n+1	LDII	NOT	-	-	
n+2	ADDI	LDII	NOT	-	
n+3	CMPI	ADDI	LDII	NOT	XF1 = 1
n+3	-	CMPI	ADDI	LDII	XF1 = 0
n+4	-	CMPI	ADDI	LDII	

Типы конфликтов конвейера «задержание всего»

Существует три типа конфликтов конвейера при работе с памятью типа «задержание всего»:

- операция загрузки или сохранения ЦПУ не может быть выполнена из-за того, что занята внешняя шина;

- внешняя загрузка занимает более одного цикла;

- условные вызовы и системные прерывания.

Первый тип конфликта «задержание всего» возникает, когда один из внешних портов занят из-за обращения, которое началось, но не завершилось. В примере 9.11 первое сохра-

нение – двухцикловое. ЦПУ записывает данные во внешний порт. Управление портом требует два цикла для завершения записи данные-данные. LDF – чтение через тот же внешний порт. Т. к. сохранение не завершено, ЦПУ продолжает попытки выполнить LDF, пока порт не станет доступен.

Пример 9.11 – Занят внешний порт

```
STF  R0, @DMA1
LDF  @DMA2, R0
```

Работа конвейера:

PC	F	D	R	E	
n	STF	-	-	-	
n+1	LDF	STF	-	-	
n+2	W	LDF	STF	-	
n+2	W	LDF	(nop)	STF	двухцикловое обращение записи
n+2	W	LDF	(nop)	(nop)	к внешней шине
n+3	X	W	LDF	(nop)	
n+4	Y	X	W	LDF	

Второй тип конфликта «задержание всего» представляет собой многоцикловое чтение данных. Чтение началось и продолжается до завершения. В примере 9.12 выполняется LDF из внешней памяти, что требует нескольких циклов для завершения.

Пример 9.12 – Многоцикловое чтение данных

```
LDF  @DMA,R0
```

Работа конвейера:

PC	F	D	R	E	
n	LDF	-	-	-	
n+1	I	LDF	-	-	
n+2	J	I	LDF	-	двухцикловое обращение записи
n+3	K(пустой)	I	LDF	-	к внешней памяти
n+3	K2	J	I	LDF	

Последний тип конфликтов «задержание всего» связан с условными вызовами и системными прерываниями, которые отличаются от других команд ветвления. Поскольку другие команды ветвления выполняют условное сохранение, условные вызовы и системные прерывания выполняют условное сохранение, которое занимает на один цикл больше условного перехода (см. пример 9.13). Дополнительный цикл используется для проталкивания адреса возврата после оценки условия вызова.

Пример 9.13 – Условные вызовы и системные прерывания

Работа конвейера:

PC	F	D	R	E	
n	CALLcond	-	-	-	
n+1	I	CALLcond	-	-	
n+1	(nop)	(nop)	CALLcond	-	
n+1	(nop)	(nop)	(nop)	CALLcond	
n+1	(nop)	(nop)	(nop)	CALLcond	
n+2/CALLaddr	I	(nop)	(nop)	(nop)	Цикл сохранения PC

9.3 Разрешение конфликтов регистров

Если осуществляется обращение к вспомогательным (AR7-AR0), индексным (IR0, IR1) регистрам, указателю страницы данных (DP), указателю стека (SP) с какой либо иной целью, кроме как для генерации адреса, может возникнуть конфликт конвейера, связанный со следующим обращением к памяти. Конфликты конвейера и задержки представлены в 9.2.2. Примеры с 9.14 по 9.16 демонстрируют общее использование этих регистров, не ведущее к возникновению конфликта или пути, с помощью которых можно избежать этих конфликтов.

Пример 9.14 – Изменение генерации адреса AR с последующей генерацией AR адреса

```
LDF   7.0, R0                ; 7.0 → R0
MPYF  *++AR0(IR1), R0
ADDF  *AR2,R0
FIX
MPYF
ADDF
```

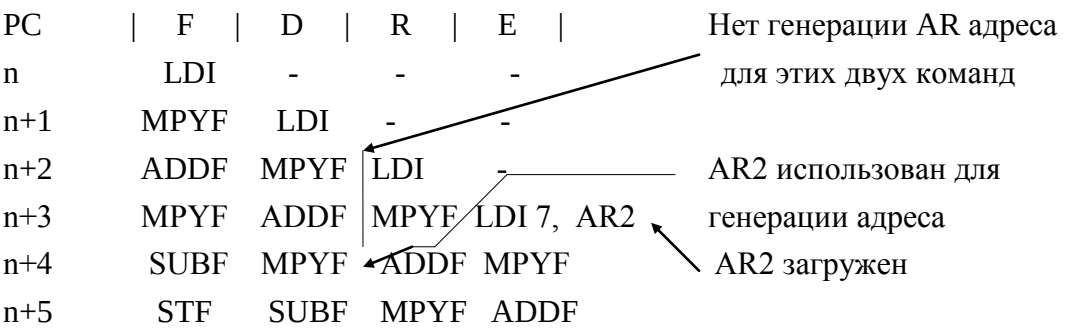
Работа конвейера:

PC	F	D	R	E	
n	LDF	-	-	-	
n+1	MPYF	LDF	-	-	Генерация адреса и изменение AR
n+2	ADDF	MPYF	LDF	-	Генерация адреса
n+3	FIX	ADDF	MPYF	LDF	
n+4	MPYF	FIX	ADDF	MPYF	
n+5	ADDF	MPYF	FIX	ADDF	

Пример 9.15 – Запись в AR с последующим использованием AR для генерации адреса без конфликта конвейера

```
LDI   @TABLE, AR2
MPYF  @VALUE, R1
ADDF  R2, R1
MPYF  *AR2++, R1
SUBF
STF
```

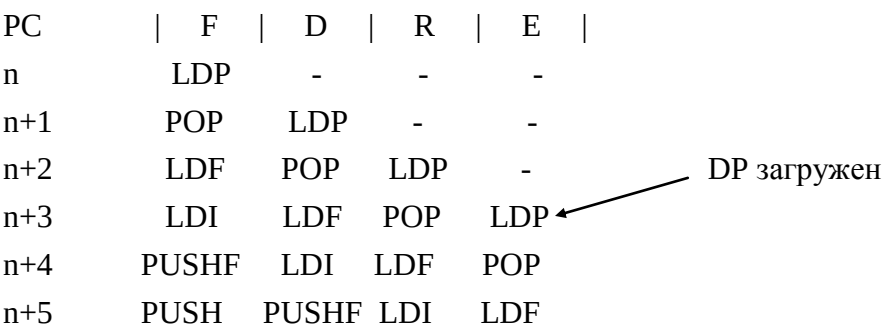
Работа конвейера:



Пример 9.16 – Запись в DP с последующим прямым чтением памяти без конфликтов конвейера

```
LDP TABLE_ADDR
POP R0
LDF *-AR3(2), R1
LDI @TABLE_ADDR, AR0
PUSHF R6
PUSH R4
```

Работа конвейера:



9.4 Разрешение конфликтов памяти

Если производится выборка программы и обращение к данным таким образом, что ресурсы, которые будут использованы, не обеспечивают необходимый диапазон, выборка программы задерживается до завершения выборки данных. Определенные конфигурации выборки программ и обращений к данным обеспечивают условия, при которых ПЦОС имеет максимальную производительность.

В таблице 9.1 приведены сведения о том, сколько выборок данных может быть произведено для разных областей памяти, когда необходимо произвести выборку программы и одиночное обращение к данным и при этом получить максимальную производительность (один цикл). В четырех случаях получается одноцикловая максимизация.

Таблица 9.1 – Одна выборка программы и одно обращение к данным для максимальной производительности

Случай	Обращения по основной шине	Выборки из внутренней памяти	Обращения по расширенной шине
1	1	1	–
2	1	–	1
3	–	2 из любой комбинации внутренней памяти	–
4	–	1	1

В таблице 9.2 показано как много выборок может производиться из различных областей памяти, когда необходимо производить выборку программы и две выборки данных, с целью обеспечения максимальной производительности (один цикл).

Таблица 9.2 – Одна выборка программы и два обращения к данным для максимальной производительности

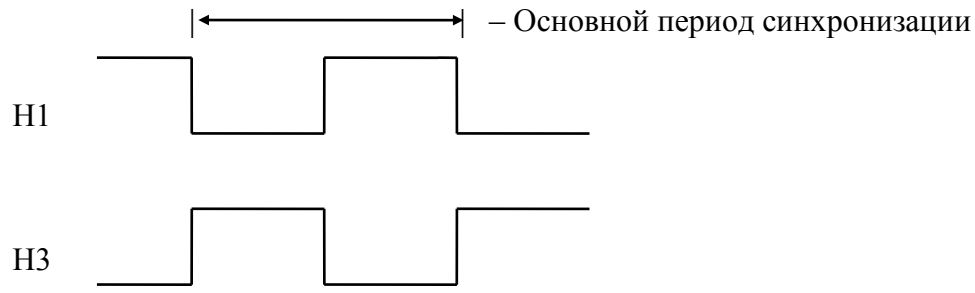
Случай	Обращения по основной шине	Выборки из внутренней памяти	Обращения по расширенной шине
1	1	2 из любой конфигурации памяти	–
2	1 программа	1 данные	1 данные
3	1 данные	1 данные	1 программа
4	–	2 из одного блока внутренней памяти и одна из другого блока внутренней памяти	–
5	–	3 из разных блоков внутренней памяти	–
6	–	2 из любой конфигурации памяти	1

9.5 Синхронизация доступа к памяти

Внутренние фазы синхронизации (H1 и H3) и их соотношения для организации доступа к памяти обсуждаются в данном разделе, чтобы показать, каким образом ПЦОС управляет несколькими обращениями к памяти. В то время как в предыдущем разделе обсуждается взаимодействие между последовательностями команд, здесь обсуждается поток данных на основе отдельной команды.

Каждый основной период синхронизации в 60 нс состоит из двух меньших периодов синхросигнала по 30 нс, называемых H3 и H1.

Активный период синхросигнала для H3 и H1 – время, когда этот сигнал в высоком уровне.



Точные операции чтения и записи памяти могут быть определены в соответствии с этими меньшими периодами синхронизации. Типы операций, которые могут возникать: выборка программы, загрузка и сохранение данных и обращения ПДП.

9.5.1 Выборки программы

Внутренние выборки программы всегда выполняются в течение H3, если другая команда в конвейере не должна произвести одно сохранение данных в то же самое время.

В этом случае выборка программы производится в течение H1 и сохранение данных – в течение H3.

Внешние выборки программы всегда начинаются в начале H3 с адресом, представляемым на внешней шине. В конце H1 они завершаются с фиксированием слова команды.

9.5.2 Загрузка и сохранение данных

Загрузку, чтение и сохранение данных выполняют 4 типа команд: двухоперандные команды, трехоперандные команды, операции умножителя/АЛУ с командами сохранения и команды параллельного умножения и сложения. См. в разделе 5 «Адресация» более детально информацию о способах адресации.

Как описано в разделе 7 «Работа с внешней шиной», число циклов шины для обращений к внешней памяти отличается в некоторых случаях от числа циклов исполнения ЦПУ. Для внешнего чтения число циклов шины и исполнения ЦПУ идентично. При внешней записи присутствует как минимум два цикла шины, но, кроме случаев конфликта доступа порта, один цикл исполнения ЦПУ. В последующих примерах приведены различия в количестве циклов шины и ЦПУ.

Обращения к памяти двухоперандных команд

Двухоперандные команды включают все те команды, которые в разрядах 31-29 имеют значения 000 или 010. В случае чтения данных разряды 15-0 являются операндом src. Внутреннее чтение всегда производится в течение H1. Внешнее чтение данных всегда начинается в начале H3, с адресом, выставляемым на внешней шине, и завершается с защелкиванием слова данных в конце H1.

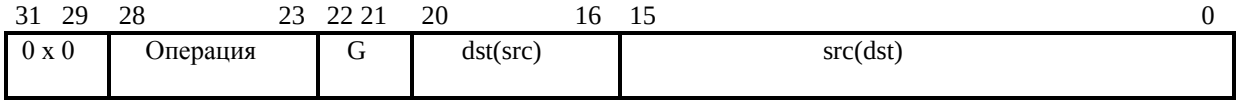


Рисунок 9.2 – Слово двухоперандной команды

В случае сохранения данных разряды 15-0 являются операндом dst. Внутренние данные сохраняются в течение H3. Внешняя запись данных всегда начинается в начале H3, с адресом и данными, выставляемыми на внешней шине.

Трехоперандные команды включают все команды, у которых разряды 31-29 будут 001. Операнды источника src1 и src2 приходят либо из регистра, либо из памяти. Когда один или более операндов источника из памяти, тогда эти команды всегда производят чтение памяти.

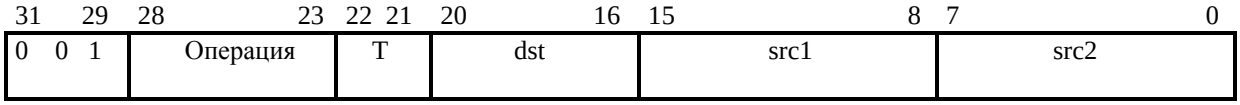


Рисунок 9.3 – Слово трехоперандной команды

Если только один из операндов источника из памяти (либо src1, либо src2) и расположен во внутренней памяти, то данные читаются в течение H1. Если один операнд памяти источника находится во внешней памяти, чтение данных начинается по H3, с адресом, выставляемым на внешней шине, и завершается с защелкиванием слова данных в конце H1.

Если оба операнда должны выбираться из памяти, могут возникать различные варианты. Если оба операнда расположены во внутренней памяти, чтение src1 производится в течение H3, а src2 – в течение H1, таким образом завершая два чтения памяти в один цикл.

Если src1 во внутренней памяти, а src2 – во внешней, выборка src2 начинается по началу H3 и защелкивается по концу H1. В то же самое время, выборка src1 из внешней памяти осуществляется в течение H3. Таким образом, опять получается два чтения памяти в один цикл.

Если src1 во внешней памяти, а src2 – во внутренней, для завершения чтения требуются два цикла. В первом цикле производится внутренняя выборка src2. Выборка src1 также производится, но не фиксируется до следующего H3.

Если src1 и src2 оба во внешней памяти, для завершения чтения требуются два цикла. В первом цикле производится выборка src1 и загрузка по следующему H3. Во втором цикле производится выборка src2 и загрузка по H1 того же цикла.

Следующий тип команд включает все команды, которые производят параллельное сохранение и другую команду. Разряды 31 и 30 для этих команд равны 1 1.

Для тех операций, которые производят умножение или операции АЛУ параллельно с сохранением, формат слова команды приведен на рисунке 9.4.

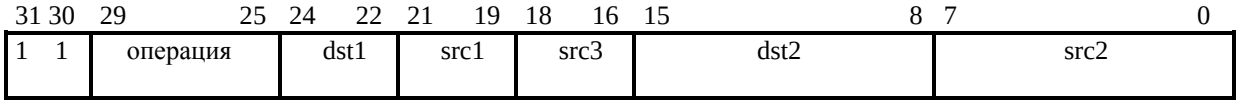


Рисунок 9.4 – Умножение или операции с параллельным сохранением

Если операция сохранения для внешнего или внутреннего операнда назначения dst2, она выполняется в течение H3. Два цикла шины требуются для внешних сохранений, но только один цикл ЦПУ необходим для завершения записи.

Если операция чтения памяти – внешняя, она начинается в начале H3 и защелкивается в конце H1. Если операция чтения памяти является внутренней, она выполняется в течение H1. Следует помнить, что чтение памяти выполняется ЦПУ в течение фазы чтения (R) конвейера и сохранение выполняется в течения фазы исполнения (E).

Формат слова команды для тех команд, в которых присутствуют параллельные сохранения в памяти, приведены на рисунке 9.5.

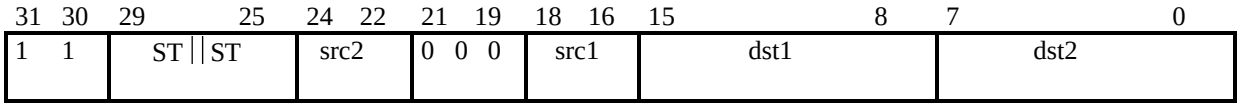


Рисунок 9.5 – Два параллельных сохранения

Если оба операнда назначения, dst1 и dst2, расположены во внутренней памяти, dst1 сохраняется в течение H3, а dst2 в течение H1, завершая таким образом два сохранения в памяти в один цикл.

Если dst1 – во внешней памяти, а dst2 – во внутренней, сохранение dst1 начинается в начале H3. Сохранение dst2 во внутренней памяти выполняется в течение H1. Для внешнего сохранения требуется два цикла шины, но только один цикл ЦПУ требуется для завершения записи. Снова два сохранения в памяти завершается в один цикл.

Если dst1 – во внутренней памяти, а dst2 – во внешней, требуется дополнительный цикл шины для завершения сохранения dst2. Для завершения записи требуется только один

цикл ЦПУ, но доступ порта требуется в течение трех циклов шины. В первом цикле выполняется внутреннее сохранение dst1 в течение H3, и dst2 записывается в порт в течение H1. В течение следующего цикла выполняется сохранение dst2 на внешней шине, начиная с H3, и выполняется как нормальное в течение следующего цикла.

Если оба операнда (dst1 и dst2) пишутся во внешнюю память, то по прежнему требуется только один цикл ЦПУ для завершения сохранения. В этом случае требуются четыре цикла шины.

В первом цикле оба операнда (dst1 и dst2) пишутся в порт, и начинается доступ к внешней шине dst1.

Сохранение dst1 завершается во втором цикле.

Сохранение dst2 начинается в третьем цикле.

Сохранение dst2 завершается на четвертом цикле внешней шины.

Параллельные умножения и сложения

Адресация памяти для параллельных умножений и сложений подобна адресации для трехоперандных команд. Параллельное умножение и сложение включает все команды, разряды 31-30 которых равны 10.

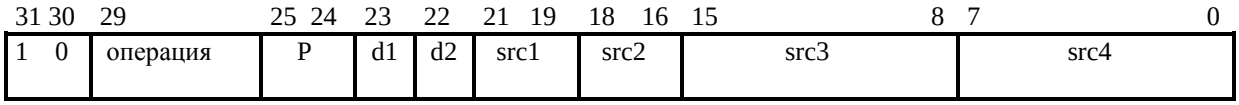


Рисунок 9.6 – Параллельные умножения и сложения

Для этих операций операнды src3 и src4 расположены в памяти. Если оба операнда расположены во внутренней памяти, src3 выполняется в течение H3, а src4 – в течение H1, таким образом, два чтения памяти завершаются в один цикл.

Если src3 – во внутренней памяти, а src4 – во внешней, выборка src4 начинается в начале H3 и защелкивается в конце H1. В то же самое время, обращение src4 во внутреннюю память выполняется в течение H3.

Если src3 – во внешней памяти, а src4 – во внутренней, для завершения двух операций чтения требуется один цикл. В первом цикле выполняется внутренняя выборка src4. В течение H3 следующего цикла выполняется выборка src3.

Если оба src3 и src4 – во внешней памяти, для завершения двух чтений требуется два цикла. В первом цикле производится выборка src3; во втором цикле производится выборка src3.

10 Команды языка ассемблера

Система команд языка ассемблера ПЦОС поддерживает высокопроизводительные вычислительные функции и может применяться как для цифровой обработки сигналов, так и для общего назначения. Команды объединены в основные группы, состоящие из команд загрузки и сохранения, двух- или трехоперандных арифметико-логических, параллельных команд, а также команд программного управления и блокировки (управления межпроцессорным взаимодействием). Режимы адресации, используемые в данных командах, описаны в разделе 5 «Адресация».

Система команд языка ассемблера ПЦОС может также использовать один из 20 кодов условий с одной из 10 условных команд, таких как LDF cond. Данный раздел описывает коды условий и флаги.

Ассемблер имеет дополнительные синтаксические формы для упрощения языка ассемблера для специальных команд. Список дополнительных форм с описанием приводится.

Каждая отдельная команда описана и приведена в алфавитном порядке. Пример команды демонстрирует специальный использованный формат и объясняет его контекст.

В данном разделе рассматриваются следующие основные положения:

- система команд (подраздел 10.1);
- команды загрузки и сохранения;
- двухоперандные арифметико-логические команды;
- трехоперандные арифметико-логические команды;
- команды программного управления;
- команды управления блокировкой;
- команды параллельных операций;
- коды условий и флаги (подраздел 10.2);
- отдельные команды (подраздел 10.3);
- символы и аббревиатура, используемые в командах;
- дополнительный синтаксис ассемблера;
- описание отдельных команд в алфавитном порядке, включая:
- синтаксис;
- описание работы;
- операнды;
- код;
- описание;
- количество циклов выполнения;
- разряды состояния;
- режимный разряд;
- примеры.

10.1 Система команд языка ассемблер

Система команд ПЦОС исключительно хорошо подходит для решения задач ЦОС и других приложений, требующих высокопроизводительные вычисления. Все команды имеют длину одного машинного слова, и большинство команд выполняется за один цикл. В дополнение к командам умножения и накопления, ПЦОС обладает полной системой команд общего назначения.

Система команд содержит 113 команд, организованных в следующие функциональные группы:

- команды загрузки и сохранения;
- двухоперандные арифметико-логические команды;
- трехоперандные арифметико-логические команды;
- команды программного управления;
- команды управления блокировкой;
- команды параллельных операций.

Каждая из этих групп обсуждается ниже.

10.1.1 Команды загрузки и сохранения

ПЦОС поддерживает 12 команд загрузки и сохранения (см. таблицу 10.1).

Эти команды могут:

- загрузить слово из памяти в регистр;
- сохранить слово из регистра в памяти;
- манипулировать данными в системном стеке.

Две из этих команд могут загружать данные по условию. Это используется для нахождения минимального или максимального значения из набора данных. См. подраздел 10.2 с детальным описанием кодов условий.

Таблица 10.1 – Команды загрузки и сохранения

Команда	Описание	Команда	Описание
LDE	Загружает экспоненту с ПЗ	POP	Выталкивает целое из стека
LDF	Загружает значение с ПЗ	POPF	Выталкивает значение с ПЗ
LDFcond	Загружает значение с ПЗ из стека по условию	PUSH	Загружает целое в стек
LDI	Загружает целое	PUSHF	Загружает значение с ПЗ в стек
LDIcond	Загружает целое по условию	STF	Сохраняет значение с ПЗ
LDM	Загружает мантиссу с ПЗ	STI	Сохраняет целое

10.1.2 Двухоперандные команды

ПЦОС поддерживает полную систему из 35 двухоперандных арифметических и логических команд. Два операнда являются исходным операндом и операндом результата. Исходным операндом может быть слово памяти, регистр или часть слова команды. Операнд результата – всегда регистр.

Эти команды обеспечивают целочисленную арифметику, арифметику с ПЗ или логические операции и арифметику повышенной точности. В таблице 10.2 приведены эти команды.

Таблица 10.2 – Двухоперандные команды

Команда	Описание	Команда	Описание
ABSF	Абсолютное значение числа с ПЗ	NORM	Нормализовать значение с ПЗ
ABSI	Абсолютное значение целого	NOT	Поразрядное логическое дополнение
ADDC*	Сложить целое с переносом	OR*	Поразрядное логическое ИЛИ
ADDF*	Сложить значение с ПЗ	RND	Округлить значение с ПЗ
ADDI*	Сложить целые	ROL	Циклический сдвиг влево
AND*	Поразрядное логическое И	ROLC	Левый циклический сдвиг с переносом
ANDN*	Поразрядное логическое И с дополнением	ROR	Циклический сдвиг вправо
ASH*	Арифметический сдвиг	RORC	Циклический сдвиг вправо через перенос
CMPF*	Сравнить значения с ПЗ	SUBB*	Вычитание целых с заемом
CMPI*	Сравнить целые	SUBC	Вычитание целых с условием
FIX	Преобразовать ПЗ в целое	SUBF	Вычитание значений с ПЗ
FLOAT	Преобразовать целое в ПЗ	SUBI	Вычесть целое
LSH	Логический сдвиг	SUBRB	Вычесть обратное целое с заемом
MPYF*	Умножить значения с ПЗ	SUBRF	Вычесть обратное ПЗ с заемом
MPYI*	Умножить целые	SUBRI	Вычесть обратное целое
NEGB	Отрицание целого с заемом	TSTB*	Тестировать разрядные поля
NEGF	Отрицание числа в формате с ПЗ	XOR*	Поразрядное исключающее ИЛИ
NEGI	Отрицание целого		

_____

* Двух- и трехоперандные версии команд.

10.1.3 Трехоперандные команды

Большинство команд имеет только два операнда, однако, некоторые арифметические и логические команды имеют трехоперандные версии. 17 трехоперандных команд позволяют ПЦОС считывать два операнда из памяти или регистрового файла ЦПУ в один цикл и сохранять результаты в регистре. Ниже описываются различия между двух- и трехоперандными командами:

- двухоперандные команды имеют один исходный операнд (или сдвиг счетчика) и один операнд результата;

- трехоперандные команды могут иметь два исходных операнда (или один исходный операнд и операнд счетчика) и один операнд результата. Исходным операндом является слово памяти или регистр. Операнд результата в трехоперандных командах всегда регистр.

В таблице 10.3 приведен список трехоперандных команд. Необходимо обратить внимание, что число 3 может быть опущено в мнемонике трехоперандной команды (см. 10.3.2).

Таблица 10.3 – Трехоперандные команды

Команда	Описание	Команда	Описание
ADDC3	Сложение с переносом	MPYF3	Умножить значения с ПЗ
ADDF3	Сложить значения с ПЗ	MPYI3	Умножить целые
ADDI3	Сложить целые	OR3	Поразрядное логическое ИЛИ
AND3	Поразрядное логическое И	SUBB3	Вычитание целых с заемом
AND3N	Поразрядное логическое И с дополнением	SUBF3	Вычитание значений с ПЗ
ASH3	Арифметический сдвиг	SUBI3	Вычитание целых
CMPF3	Сравнить значения с ПЗ	TSTB3	Тестирование разрядных полей
CMPI3	Сравнить целые	XOR3	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ
LSH3	Логический сдвиг		

10.1.4 Команды программного управления

Группа команд программного управления состоит из 16 команд, влияющих на выполнение программы. Режим повторения обеспечивает повторение блока команд (RPTB) или отдельной команды (RPTS). Поддерживаются как стандартные, так и задержанные (одноцикловые) переходы. Некоторые из команд программного управления могут зависеть от кодов условий (см. подраздел 10.2 для получения более детальной информации о кодах условий).

В таблице 10.4 приведены команды программного управления.

Таблица 10.4 – Команды программного управления

Команда	Описание	Команда	Описание
Bcond	Переход по условию (стандартный)	IDLE	Холостая работа до прерывания
BcondD	Переход по условию (задержанный)	NOP	Нет операции
BR	Безусловный переход (стандартный)	RETIcond	Возврат из прерывания по условию
BRD	Безусловный переход (задержанный)	RETScond	Возврат из подпрограммы по условию
CALL	Вызов подпрограммы	RPTB	Повтор блока команд
CALLcond	Вызов подпрограммы по условию	RPTS	Повтор отдельной команды
DBcond	Декремент и переход по условию (стандартный)	SWI	Программное прерывание
DBcondD	Декремент и переход по условию (задержанный)	TRAPcond	Условное системное прерывание
IACK	Подтверждение прерывания		

10.1.5 Команды операций блокировки

Команды операций блокировки поддерживают мультипроцессорные связи и используют внешние сигналы для обеспечения мощного механизма синхронизации. Они также гарантируют целостность связи и результатов в высокоскоростных операциях. См. в разделе 6 «Управление выполнением программы» примеры использования команд блокировки.

Таблица 10.5 – Команды операций блокировки

Команда	Описание	Команда	Описание
LDFI	Загрузить значение с ПЗ, с блокировкой	STFI	Сохранить значение с ПЗ, с блокировкой
LDII	Загрузить целое, с блокировкой	STII	Сохранить целое, с блокировкой
SIGI	Сигнализация с блокировкой		

10.1.6 Команды параллельных операций

Группа команд параллельных операций делает возможным высокую степень параллелизма. Некоторые команды ПЦОС могут объединяться парами, которые выполняются параллельно. Данные команды обеспечивают следующие возможности:

- параллельная загрузка регистров,
- параллельные арифметические операции,
- арифметико-логические команды, выполняемые параллельно с командой сохранения.

Каждая команда в паре вводится как отдельный исходный оператор. Вторая команда в паре должна быть отделена двумя вертикальными черточками (||). В таблице 10.6 приведен список пар команд.

Таблица 10.6 – Параллельные команды

Мнемоника	Описание
Команды параллельного выполнения арифметических операций и сохранения	
ABSF||STF	Абсолютное значение числа в формате с ПЗ и сохранить значение с ПЗ
ABSI||STI	Абсолютное значение целого числа и сохранить целое
ADDF3||STF	Сложить значения в формате с ПЗ и сохранить значение с ПЗ
ADDI3||STI	Сложить целые и сохранить целое
AND3||STI	Поразрядное логическое И и сохранить целое
ASH3||STI	Арифметический сдвиг и целое
FIX||STI	Преобразовать значение числа в формате с ПЗ в целое и сохранить целое
FLOAT||STF	Преобразовать целое в значение числа в формате с ПЗ и сохранить в формате с ПЗ
FRIEEE||STF	Преобразовать из IEEE формата в формат с ПЗ в дополнительном коде и сохранить в формате с ПЗ
LDF||STF	Загрузить значение в формате с ПЗ и сохранить в формате с ПЗ
LDI||STI	Загрузить целое и сохранить целое
LSH3||STI	Логический сдвиг и сохранить целое
MPYF3||STF	Умножить значения в формате с ПЗ и сохранить значение с ПЗ
MPYI3||STI	Умножить целое и сохранить целое
NEGF||STF	Обратное значение в формате с ПЗ и сохранить значение с ПЗ
NEG||STI	Обратное целое и сохранить целое
NOT||STI	Логическое дополнение (поразрядная инверсия) значения и сохранить целое
OR3||STI	Поразрядное логическое ИЛИ и сохранить целое
STF||STF	Сохранить значения в формате с ПЗ
STI||STI	Сохранить целые
SUBF3||STF	Вычесть значение в формате с ПЗ и сохранить значение в формате с ПЗ
TOIEEE||STF	Преобразование в IEEE формат и сохранение
SUBI3||STI	Вычесть целое и сохранить целое
XOR3||STI	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ значений и сохранить целое
Команды параллельной загрузки	
LDF||LDF	Загрузить значение в формате с ПЗ
LDI||LDI	Загрузить целое
Команды параллельного умножения и сложения/вычитания	
MPYF3||ADDF3	Умножить и прибавить значение в формате с ПЗ
MPYF3||SUBF3	Умножить и вычесть значение в формате с ПЗ
MPYI3||ADDI3	Умножить и прибавить целое
MPYI3||SUBI3	Умножить и вычесть целое

10.2 Коды условий и флаги

В ПЦОС используются 20 кодов условий (с 00000 по 10100, за исключением 01011) которые могут быть использованы с условными командами, такими как RETScond или LDFcond. Условиями являются знаковые и беззнаковые сравнения, сравнение с нулем и сравнения, основанные на состоянии отдельных флагов условий. Необходимо обратить внимание, что все условные команды могут включать суффикс U для обозначения безусловной операции.

Семь флагов условий содержат информацию о свойствах результата арифметических или логических команд. Флаги условий хранятся в регистре состояния (ST) и изменяются командой в одном из следующих двух случаев:

- регистр назначения является одним из регистров с повышенной точностью (R7-R0). Это делается для модификации регистров, используемых для адресации, но не влияет на флаги условий в ходе вычисления;

- команда является одной из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3). Это дает возможность установить флаги условий в соответствии с содержимым регистров ЦПУ.

Флаги условий могут изменяться многими командами либо когда предшествующие условия установлены, либо при выполнении следующих условий:

- результат получен, когда определенная операция выполнена с бесконечной точностью. Это подходит для случая команд сравнения и тестирования, которые не сохраняют результат в регистре. Это также может подходить для арифметических команд, которые приводят к переполнению или отрицательному переполнению (потере значимости).

Более детальное описание влияния отдельных команд на флаги условий см. 10.3.3.

LUF – фиксируемый флаг условия отрицательного переполнения при работе с ПЗ. Устанавливается всякий раз при установке UF (флаг отрицательного переполнения при работе в формате с ПЗ). LUF может быть очищен только при сбросе или при изменении регистра состояния (ST).

LV – фиксируемый флаг условия переполнения. Устанавливается всякий раз при установке V (флаг условия переполнения). LV может быть очищен только при сбросе или при изменении регистра состояния (ST).

UF – флаг условия отрицательного переполнения при работе в формате с ПЗ. Отрицательное переполнение при работе с ПЗ возникает всякий раз, когда экспонента результата меньше или равна минус 128. При возникновении отрицательного переполнения UF устанавливается, и выходное значение устанавливается в 0. Если не происходит отрицательного переполнения, UF очищается.

N – флаг отрицательного условия. Логические операции присваивают N значение старшего значащего разряда выходного значения. Для целочисленных операций и операций с ПЗ N устанавливается, если результат отрицательный, и очищается в противном случае. Ноль означает положительное значение.

Z – флаг нулевого условия. Для логических, целочисленных и ПЗ-операций Z устанавливается при равенстве выходного результата 0 и очищается в противном случае.

V – флаг условия переполнения. Для целочисленных операций V устанавливается, если результат не вписывается в формат, определенный для назначения (т. е., $-2^{32} \leq$ результат $\leq 2^{32}-1$). В противном случае V очищается. Для операций в формате с ПЗ V устанавливается, если экспонента результата больше 127; в противном случае V очищается. Логические операции всегда очищают V.

C – при выполнении целочисленного сложения C очищается в случае возникновения переноса относительно старшего значащего разряда выходного значения. При выполнении целочисленного вычитания C устанавливается при возникновении заема в разряде, относящемся к старшему значащему разряду выходного значения. В противном случае при целочисленных операциях C очищается. Для команд сдвига этот флаг устанавливается по значению последнего сдвигаемого вовне разряда; при нулевом значении сдвига флаг устанавливается в ноль.

В таблице 10.7 приведена мнемоника условия, код, описание и флаг для каждого из 19 кодов условий.

Таблица 10.7 – Коды условий и флаги

Условие	Код	Описание	Флаг*
Безусловные сравнения			
U	00000	Безусловный	Безусловное
Беззнаковые сравнения			
LO	00001	Меньше чем	C
LS	00010	Меньше или равно	C или Z
HI	00011	Больше чем	~C и~Z
HS	00100	Больше или равно	~C
EQ	00101	Равно	Z
NE	00110	Не равно	~Z
Знаковые сравнения			
LT	00111	Меньше чем	N
LE	01000	Меньше или равно	N или Z
GT	01001	Больше чем	~N и ~Z
GE	01010	Больше или равно	~N
EQ	00101	Равно	Z
NE	00110	Не равно	~Z
Сравнение с нулем			
Z	00101	Ноль	Z
NZ	00110	Не ноль	~Z
P	01001	Положительное	~N и ~Z
N	00111	Отрицательное	N
NN	01010	Не отрицательное	~N
Сравнение с флагами условий			
NN	01010	Не отрицательное	~N
N	00111	Отрицательное	N
NZ	00110	Не ноль	~Z
Z	00101	Ноль	Z
NV	01100	Нет переполнения	~V
V	01101	Переполнение	V
NUF	01110	Нет отрицательного переполнения	~UF
UF	01111	Отрицательное переполнение	UF

Окончание таблицы 10.7

Условие	Код	Описание	Флаг*
NC	00100	Нет переноса	~C
C	00001	Перенос	C
NLV	10000	Нет фиксируемого переполнения	~LV
LV	10001	Фиксируемое переполнение	LV
NLUF	10010	Нет фиксируемого отрицательного переполнения с ПЗ	~LUF
LUF	10011	Фиксируемое отрицательное переполнение с ПЗ	LUF
ZUF	10100	Ноль или отрицательное переполнение с ПЗ	Z ИЛИ UF

_____

* ~ означает логическое дополнение (состояние «ложь»).

10.3 Отдельные команды

Этот подраздел содержит отдельные команды языка ассемблер для ПЦОС. Команды приведены в алфавитном порядке. Информация о каждой команде включает синтаксис ассемблера, производимую операцию, операнды, код, описание, количество циклов, разряды состояния, режимные разряды и примеры.

Определение символов и аббревиатур, также как и дополнительных синтаксических форм, поддерживаемых ассемблером, находится в начале раздела описания отдельных команд. Также приводится пример команды с описанием используемого специального формата и объясняется ее контекст.

Группировка по функциям команд и полный список команд приведены выше в подразделе 10.1. Информация по режимам адресации содержится в разделе 6 «Управление выполнением программы».

10.3.1 Символы и аббревиатура

В таблице 10.8 приведены символы и аббревиатура, использованные в описании отдельных команд.

Таблица 10.8 – Символы и аббревиатура, использованные в описании команд

Обозначение	Назначение
src	Операнд источника (исходный)
src1	Операнд источника 1 (исходный)
src2	Операнд источника 2 (исходный)
src3	Операнд источника 3 (исходный)
src4	Операнд источника 4 (исходный)
dst	Операнд назначения (результата)
dst1	Операнд назначения 1 (результата)
dst2	Операнд назначения 2 (результата)
disp	Смещение
cond	Условие
count	Счетчик сдвига
G	Основные режимы адресации
T	Трехоперандные режимы адресации
P	Параллельные режимы адресации

Окончание таблицы 10.8

Обозначение	Назначение
B	Режимы адресации с условным переходом
ARn	Вспомогательный регистр n
IRn	Индексный регистр n
Rn	Регистр повышенной точности n
RC	Регистр счетчика повторений
RE	Регистр конечного адреса повторений
RS	Регистр начального адреса повторений
ST	Регистр состояния
C	Разряд переноса
GIE	Разряд разрешения глобальных прерываний
N	Вектор программного прерывания (trap)
PC	Счетчик команд
RM	Флаг режима повторения
SP	Указатель системного стека
|x|	Абсолютное значение x
x →y	Присваивает значение x значению y
x(man)	Поле мантиссы (знак + дробная часть) от x
x(exp)	Поле экспоненты
op1 ||op2	Операция 1 выполняется параллельно операции 2
x AND y	Поразрядное логическое И от x и y
x OR y	Поразрядное логическое ИЛИ от x и y
x XOR y	Поразрядное логическое ИСКЛЮЧАЮЩЕЕ ИЛИ от x и y
~x	Поразрядное логическое дополнение (инверсия) от x
x << y	Сдвиг x влево на y разряд
x >> y	Сдвиг x справа на y разряд
*++SP	Инкрементировать SP и использовать инкрементированный SP как адрес
*SP--	Использовать SP как адрес и декрементировать SP

10.3.2 Дополнительный синтаксис ассемблера

Ассемблер позволяет использовать упрощенную форму записи некоторых команд. Эти дополнительные формы упрощают синтаксис ассемблера, т. к. специфические формы синтаксиса могут игнорироваться.

Регистр назначения может быть опущен в одинарных арифметических и логических операциях, когда тот же регистр используется в качестве источника.

Например, ABSI R0, R0 может быть записана как ABSI R0 .

Используемые команды: ABSI, ABSF, FIX, FLOAT, NEGB, NEGF, NEGI, NORM, NOT, RND.

Все трехоперандные команды могут быть записаны без 3.

Например, ADDI3 R0, R1, R2 может быть записана как ADDI R0, R1, R2.

Используемые команды: ADDC3, ADDF3, ADDI3, AND3, ANDN3, ASH3, LSH3, MPYF3, MPYI3, OR3, SUBB3, SUBF3, SUBI3, XOR3.

Все трехоперандные команды сравнения могут быть записаны без 3.

Например, CMPI3 R0, *AR0 может быть записана как CMPI R0, *AR0.

Используемые команды: CMPI3, CMPF3, TSTB3.

Разрешены непрямые операнды с явно заданным нулевым смещением. В трехоперандных или параллельных командах операнды с нулевым смещением автоматически преобразуются в режим с отсутствием смещения. Например, разрешено: LDI *+AR0 (0), R1.

Также:

ADDI3 *+AR0 (0), R1, R2 эквивалентно ADDI3 *+AR0, R1, R2.

Непрямые операнды могут быть записаны без смещения, в таком случае предполагается единичное смещение. Например,

LDI *AR0++ (1), R0 может быть записано как LDI *AR0++, R0

Все условные команды включают в себя суффикс U для обозначения безусловной операции. Также, суффикс U может быть исключен из команды короткого безусловного перехода. Например, BU label (метка) может быть записана как B label (метка).

Метки могут быть записаны как с последующим символом (:), так и без него. Например:

label0: NOP

label1: NOP

label2: (метка относится к следующей строке).

Пустые выражения запрещены для указания смещения в косвенном режиме:

LDI *+AR0 (), R0 – запрещено.

Операнды длинного непосредственного режима (назначение операторов BR и CALL) могут быть написаны со знаком @:

BR метка может быть записан как BR @метка.

Псевдооперация LDP может быть использована для загрузки регистра (обычно DP) 8 младшими разрядами перемещаемого адреса следующим образом:

LDP addr, REG или LDP @addr, REG.

Знак @ является дополнительным. Если регистр назначения это DP, он может быть опущен в операнде. LDP генерирует команду LDI с непосредственным операндом и специальным типом смещения.

Параллельные команды могут быть написаны в другом порядке. Например,

ADDI может быть записано как STI

|| STI || ADDI

Символы (||), определяющие 2 часть параллельной команды, могут быть написаны в любом месте строки, начиная с 0 столбца. Например,

ADDI может быть записано как ADDI

|| STI || STI

Если второй операнд параллельной команды такой же как и третий (регистр назначения), третий операнд может быть опущен. Т. е. это дает возможность написания трехоперандной команды, которая выглядит как нормальная двухоперандная команда. Например,

ADDI *AR0, R2, R2 может быть записана как ADD *AR0, R2, R2

|| MPYI *AR1, R0, R0 || MPYI *AR1, R0

Команды, для которых применимо вышесказанное: ADDI, ADDF, AND, MPYI, MPYF, OR, SUBI, SUBF, XOR (для параллельных команд, имеющих в качестве второго операнда регистр).

Все коммутируемые операнды в параллельных командах могут быть записаны в другом порядке. Например, часть параллельной команды ADDI может быть записана одним из двух способов:

ADDI *AR0, R1, R2 или ADDI R1, *AR0, R2

Вышеописанное относится к параллельным командам, содержащим одну из перечисленных команд:

ADDI, ADDF, MPYI, MPYF, AND, OR, XOR.

10.3.3 Описание отдельных команд

Далее описана каждая команда ассемблера ПЦОС в алфавитном порядке.

Информация о каждой команде включает синтаксис ассемблера, производимую операцию, операнды, код, описание, количество циклов, разряды состояния, режимные разряды и примеры.

Таблица 10.9 – Синтаксис регистров ЦПУ

Синтаксис ассемблера	Альтернативный синтаксис регистров	Назначение регистров
R0	R0	Регистр повышенной точности 0
R1	R1	Регистр повышенной точности 1
R2	R2	Регистр повышенной точности 2
R3	R3	Регистр повышенной точности 3
R4	R4	Регистр повышенной точности 4
R5	R5	Регистр повышенной точности 5
R6	R6	Регистр повышенной точности 6
R7	R7	Регистр повышенной точности 7
AR0	R8	Вспомогательный регистр 0
AR1	R9	Вспомогательный регистр 1
AR2	R10	Вспомогательный регистр 2
AR3	R11	Вспомогательный регистр 3
AR4	R12	Вспомогательный регистр 4
AR5	R13	Вспомогательный регистр 5
AR6	R14	Вспомогательный регистр 6
AR7	R15	Вспомогательный регистр 7
DP	R16	Указатель страницы данных
IR0	R17	Индексный регистр 0
IR1	R18	Индексный регистр 1
BK	R19	Регистр размера блока
SP	R20	Указатель системного стека
ST	R21	Регистр состояния
IE	R22	Регистр разрешения прерывания ЦПУ/ПДП
IF	R23	Флаги прерываний ЦПУ
IOF	R24	Флаги ввода-вывода
RS	R25	Регистр адреса начала повторения
RE	R26	Регистр адреса конца повторения
RC	R27	Счетчик повторений

Синтаксис: INST src, dst

Или

INST1 src2, dst1

||INST2 src3, dst2

Каждая команда начинается с синтаксического выражения ассемблера. Метки могут размещаться либо до команды (мнемонического выражения) на той же строке, либо на предыдущей строке в первом столбце. Дополнительное поле комментария, которое завершает синтаксис, не включается в синтаксическое выражение. Пробелы требуются между всеми полями (метка, команда, операнд и поля комментария).

Примеры синтаксиса иллюстрируют общий однострочковый и двухстрочковый синтаксис, используемый в параллельной адресации. Необходимо обратить внимание, что символ (||), обозначающий параллельную адресную пару, может быть размещен где угодно перед мнемоникой во второй строке. Первая команда в паре может иметь метку, но вторая иметь метки не может.

Операция: |src| → dst

или

|src2| → dst1

|| src3 → dst2

Последовательность работы команды описывает процесс, который имеет место при выполнении команды. Для параллельных команд рабочая последовательность выполняется параллельно. Условные эффекты, определенные режимами регистра состояния приводятся для условных команд, таких как Bcond.

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

или

src2 - косвенная (смещение = 0, 1, IR0, IR1)

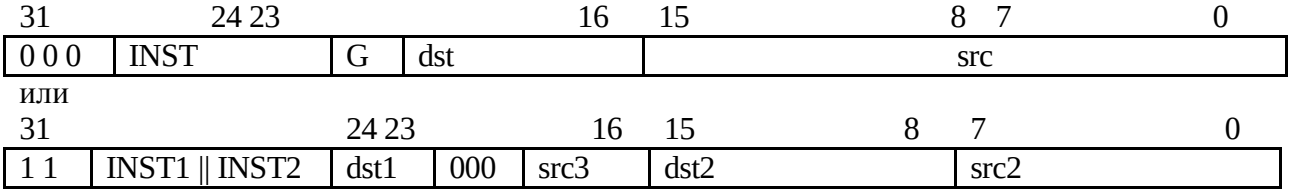
dst1 - регистр (Rn1, $0 \le n1 \le 7$)

src3 - регистр (Rn2, $0 \le n2 \le 7$)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Операнды определены в соответствии с режимом адресации и/или типом использованной адресации. Необходимо обратить внимание, что косвенная адресация использует смещения и индексные регистры (см. 5.1.3).

Код:



Примеры кодов приведены с использованием основной и параллельной адресации. Пара команд для примера параллельной адресации состоит из INST1 и INST2.

Описание:

Описано выполнение команды и ее влияние на состояние процессора и содержимое памяти. Описаны также ограничения, накладываемые на операнды процессором или ассемблером. Описание идет параллельно и дополняет информацию, приведенную в описании работы.

Циклы: 1

Число описывает количество циклов, требуемых для выполнения команды.

Разряды состояния:

LUF – фиксируемый флаг условия отрицательного переполнения при работе с ПЗ. Устанавливается всякий раз при установке UF (флаг отрицательного переполнения при работе в формате с ПЗ), в противном случае не изменяется.

LV – фиксируемый флаг условия переполнения. Устанавливается в 1 всякий раз при переполнении, в противном случае не изменяется.

UF – флаг условия отрицательного переполнения при работе в формате с ПЗ. При возникновении отрицательного переполнения UF устанавливается в 1, в противном случае 0.

N – флаг отрицательного условия. Для некоторых операций N имеет значение старшего значащего разряда выходного значения; 1 – если результат отрицательный и 0 – в противном случае.

Z – флаг нулевого условия. 1 – при равенстве выходного результата нулю и 0 – в противном случае. Для логических команд и команд сдвига: 1 – при генерации 0 выхода, 0 – в противном случае.

V – флаг условия переполнения. Устанавливается в 1 при переполнении, в 0 – в противном случае.

C – устанавливается в 1 при возникновении заема или переноса, 0 – в противном случае. Для команд сдвига этот флаг устанавливается по значению последнего сдвигаемого вовне разряда; при нулевом сдвиге устанавливается в ноль.

Семь флагов условий, сохраняемые в регистре состояния (ST), изменяются большинством команд тогда, когда регистр назначения представляет собой один из R7-R0. Они выдают информацию о свойствах результата или выхода арифметических или логических операций.

Разряд режима: Флаг режима переполнения OVM. В общем случае на выполнение целочисленных операций влияет значение разряда OVM (описано в таблице 3.2).

Пример: INST @98AEh, R5

До команды:

DP = 80h

R5 = 0766900000h = 2.30562500e+02

Значение в ячейке памяти по адресу 8098AEh = 5CDFh = 1.00001107e+00

LUF LV UF N Z V C = 0 0 0 0 0 0 0

После команды:

DP = 80h

R5 = 0066900000h = 1.80126953e+00

Значение в ячейке памяти по адресу 8098AEh = 5CDFh = 1.00001107e+00

LUF LV UF N Z V C = 0 0 0 0 0 0 0

Примерный код, представленный в вышеописанном формате, показывает влияние команды на системные указатели (DP или SP), регистры (R1 или R5), значение в памяти по определенному адресу и семь разрядов состояния. Значения, заданные для регистров,

включают начальный ноль для указания экспоненты в операциях с ПЗ. Для всех регистров и адресов памяти представлен перевод значений в десятичную систему. Разряды состояния перечислены в порядке, в котором они представлены в ассемблере или симуляторе (см. таблицу 10.7).

ABSF Абсолютное значение числа с плавающей запятой

Синтаксис: ABSF src, dst

Операция: |src| → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 29	28 23	22 21	20 16	15 0
0 0 0	0 0 0 0 0 0	G	dst	src

Описание: Абсолютное значение операнда src загружается в регистр dst. Операнды src и dst являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 0

N 0

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
ABSF    R4,R7
```

Before Instruction			After Instruction		
R4	5C 8000 F971	-9.90337307e+27	R4	5C 8000 F971	-9.90337307e+27
R7	7D 2511 00AE	5.48527255e+37	R7	5C 7FFF 068F	9.90337307e+27
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

ABSF||STF Абсолютное значение числа в формате с ПЗ и сохранить значение с ПЗ

Синтаксис: ABSF src2, dst1

|| STF src3, dst2

Операция: |src2| → dst1

|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)

src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 30	29 25	24 23	22 20	19 16	15 8	7 0
0 0	0 0 1 0 0	dst1	0 0 0	src3	dst1	src3

Описание: Абсолютное значение числа в формате с ПЗ и параллельно сохранить значение в формате с ПЗ. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (ABSF), записывает в тот же регистр, STF имеет на входе содержимое регистра до того, как оно было изменено командой ABSF. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2. Если src3 и dst1 указывают на один и тот же адрес, src3 считывается до записи в dst1. Переполнение возникает, если src(man) = 80000000h и src(exp) = 7Fh. Результат – dst(man) = 7FFFFFFFh и dst(exp)=7Fh.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 0

N 0

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
        ABSF    *++AR3(IR1) ,R4
||      STF     R4,*-AR7(1)
```

	Before Instruction		After Instruction	
R4	07 33C0 0000	1.79750e+02	05 74C0 0000	6.118750e+01
AR3	80 9800		8098AF	
AR7	80 98C5		8098C5	
IR1	0AF		0AF	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data Memory				
8098AF	58B4000	-6.118750e+01	58B4000	-6.118750e+01
8098C4	0		733C000	1.79750e+02

ABSI Абсолютное значение целого числа

Синтаксис: ABSI src, dst

Операция: |src| → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 29	28 24	23	22 16	19 16	15 0
0 0 0	0 0 0 0 0 1	G	dst	src3	src

Описание: Абсолютное значение операнда src загружается в регистр dst. Операнды src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 0

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

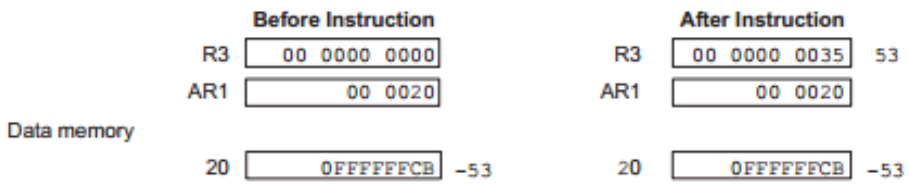
Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1:

```
ABSI    R0,R0
or
ABSI    R0
```

Пример 2:

```
ABSI    *AR1,R3
```



ABSI||STI Абсолютное значение целого числа и сохранить целое

Синтаксис: ABSI src2, dst1
|| STI src3, dst2

Операция: |src2| → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 30	29 25	24 23	22 20	16	15 8	7 0
1 1	0 0 1 0 1	dst1	0 0 0	src3	dst1	src2

Описание: Абсолютное значение числа целого и сохранить значение целого параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (ABSI), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой ABSI. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2. Переполнение возникает, если src = 80000000h.
Если ST (OVM) = 1, результат – dst = 7FFFFFFFh.
Если ST (OVM) = 0, результат – dst = 80000000h.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7-R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 0
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
        ABSI    *-AR5(1),R5
||      STI     R1,*AR2--(IR1)
```

Before Instruction			After Instruction		
R1	00 0000 0042	66	R1	00 0000 0042	66
R5	00 0000 0000		R5	00 0000 0035	53
AR2	80 98FF		AR2	80 98F0	
AR5	80 99E2		AR5	80 99E2	
IR1	0F		IR1	0F	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
8098FF	2	-53	8098FF	42	-53
8099E1	0FFFFFFCB	2	8099E1	0FFFFFFCB	66

ADDC Сложение целых с переносом

Синтаксис: ADDC src, dst

Операция: dst + src + C → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 29	28 23	24 22	16	15 8	7 0
0 0 0	0 0 0 0 1 0	G	dst	src	

Описание: Сумма операндов dst и src и флаг C (перенос) загружается в регистр dst. Операнды src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7-R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
ADDC    R1,R5
```

	Before Instruction		After Instruction	
R1	00 FFFF 5C25	-41,947	00 FFFF 5C25	-41,947
R5	00 FFFF 019E	-65,122	00 FFFE 5DC4	-107,068
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	

ADDC3 Сложение целых с переносом (3 операнда)

Синтаксис: ADDC3 src2, src1, dst

Операция: src1 + src2 + C → dst

Операнды: Трехоперандные режимы адресации src1 (T):
00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)
1 - косвенная (смещение = 0, 1, IR0, IR1)
Трехоперандные режимы адресации src2 (T):
00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)
11 - косвенная (смещение = 0, 1, IR0, IR1)
dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 23	24	22 16	15 8	7 0
0 0 1	0 0 0 0 0 0	T	dst	src3	src

Описание: Сумма операндов src1 и src2, и флаг C (перенос) загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7-R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1:

```
ADDC3     *AR5++(IR0),R5,R2
 or
ADDC3     R5,*AR5++(IR0),R2
```

Before Instruction			After Instruction		
R2	00 0000 0000		R2	00 0000 0032	50
R5	00 0000 0066	102	R5	00 0000 0066	102
AR5	80 9908		AR5	80 9918	
IR0	10		IR0	10	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	1		C	1	
Data memory					
809908	0FFFFFCB	-53	809908	0FFFFFCB	-53

Пример 2:

```
ADDC3     R2, R7, R0
```

Before Instruction			After Instruction		
R0	00 0000 0000		R0	00 0000 123F	4671
R2	00 0000 02BC	700	R2	00 0000 02BC	700
R7	00 0000 0F82	3970	R7	00 0000 0F82	3970
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	1		C	0	

ADDF Сложение значений с плавающей запятой

Синтаксис: ADDF src, dst

Операция: dst + src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 29	28 23	24 22	16	15 8	7 0
0 0 0	0 0 0 0 1 1	G	dst	src	

Описание: Сумма операндов dst и src загружается в регистр dst.

Операнды src и dst – числа в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7-R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

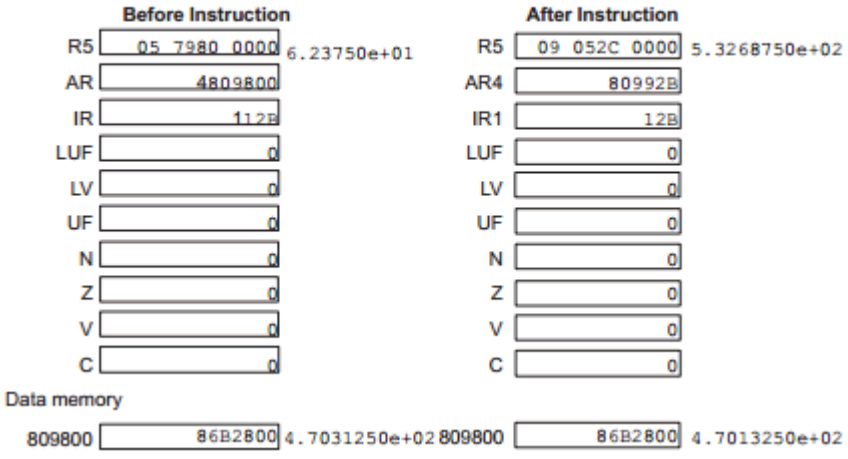
V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
ADDF *AR4++(IR1),R5
```



ADDF3 Сложение значений с плавающей запятой (3 операнда)

Синтаксис: ADDF3 src2, src1, dst

Операция: src1 + src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 29	28 23	24	22 16	15 8	7 0
0 0 1	0 0 0 0 0 1	T	dst	src1	src2

Описание: Сумма операндов src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1:

```
ADDF3     R6,R5,R1
or
ADDF3     R5,R6,R1
```

Before Instruction			After Instruction		
R1	00 0000 0000		R1	09 052C 0000	5.3268750e+02
R5	05 7980 0000	6.23750e+01	R5	05 7980 0000	6.23750e+01
R6	08 6B28 0000	4.7031250e+02	R6	08 6B28 0000	4.7031250e+02
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

Пример 2:

```
ADDF3     *+AR1(1),*AR7++(IR0),R4
```

	Before Instruction		After Instruction		
R4	00 0000 0000		R4	07 0DB2 0000	1.41695313e+02
AR1	80 9820		AR1	80 9820	
AR7	80 99F0		AR7	80 99F8	
IR0	8		IR0	8	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UV	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
809821h	700F000	1.28940e+02	809821h	700F000	1.28940e+02
8099F0h	34C2000	1.27590e+01	8099F0h	34C2000	1.27590e+01

ADDF3||STF Сложить значения в формате с ПЗ и сохранить значение с ПЗ

Операция: src1 + src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 30	29 24	23 16	15 8	7 0		
0 0	0 0 1 1 0	dst1	src1	src3	dst2	src2

Описание: Сложение в формате с ПЗ и сохранение числа в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (ADDF3), записывает в тот же регистр, STF имеет на входе содержимое регистра до того, как оно было изменено командой ADDF3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF 1 при возникновении отрицательного переполнения, иначе 0.
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример

```
        ADDF3   *+AR3(IR1),R2,R5
||      STF     R4,*AR2
```

Before Instruction			After Instruction		
R2	07 0C80 0000	1.4050e+02	R2	07 0C80 0000	1.4050e+02
R4	05 7B40 0000	6.281250e+01	R4	05 7B40 0000	6.281250e+01
R5	00 0000 0000		R5	08 2020 0000	3.20250e+02
AR2	80 98F3		AR2	80 98F3	
AR3	80 9800		AR3	80 9800	
IR1	0A5		IR1	0A5	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UV	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
8098A5h	733C000	1.79750e+02	8098A5h	733C000	1.79750e+02
8098F3h	0		8098F3h	57B4000	6.28125e+01

ADDI Сложение целых

Синтаксис: ADDI src, dst

Операция: dst + src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 22	16	15 8	7 0
0 0 0	0 0 0 0 1 1	G	dst	src	

Описание: Сумма операндов dst и src загружается в регистр dst. Операнды dst и src являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: ADDI R3, R7

Before Instruction			After Instruction		
R3	00 FFFF FFCB	-53	R3	00 FFFF FFCB	-53
R7	35	53	R7	00 0000 0000	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	

ADDI3 Сложение целых (3 операнда)

Синтаксис: ADDI3 <src2>, <src1>, <dst>

Операция: src1 + src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 1	0 0 0 0 1 0	T	dst	src1	src2

Описание: Сумма операндов src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

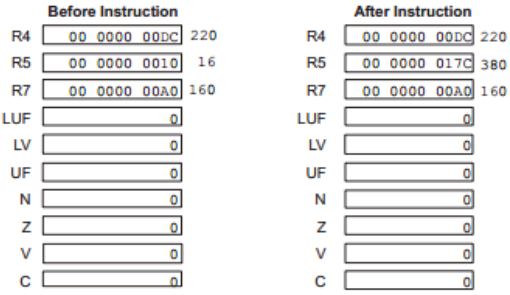
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

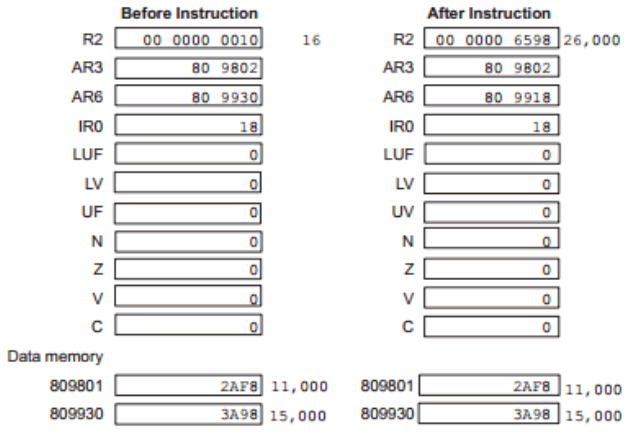
C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1: ADDI3 R4, R7, R5



Пример 2: ADDI3 * – AR3(1), * - - AR6(IR0), R2



ADDI3||STI Сложить целые и сохранить целое

Синтаксис: ADDI3 src2, src1, dst1
|| STI src3, dst2

Операция: src1 + src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
0 1 | 0 0 1 1 1	dst1 | src1	src3 | dst2	src2

Описание: Целочисленное сложение и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (ABSI), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой ABSI. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример
```
ADDI3   *AR0 –(IR0), R5, R0
|| STI   R3, *AR7
```

Before Instruction			After Instruction		
R0	00 0000 0000		R0	00 0000 0208	520
R3	00 0000 0035	53	R3	00 0000 0035	53
R5	00 0000 00DC	220	R5	00 0000 00DC	220
AR0	80 992C		AR0	80 9920	
AR7	80 983B		AR7	80 983B	
IR0	0C		IR0	0C	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UV	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
80992C	12C	300	80992C	12C	300
80983B	0		80983B	35	53

AND Поразрядное логическое И (AND)

Синтаксис: AND src, dst

Операция: dst AND src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 0	0 0 0 1 0 1	G	dst	src

Описание: Результат поразрядного логического И между операндами dst и src записывается в регистр dst. Предполагается, что src и dst являются целыми без знака.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения.

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: AND R1, R2

Before Instruction	
R1	00 0000 0080
R2	00 0000 0AFF
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

After Instruction	
R1	00 0000 0080
R2	00 0000 0080
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

AND3 Поразрядное логическое И (AND) (3 операнда)

Синтаксис: AND3 src2, src1, dst

Операция: src1 AND src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 1	0 0 0 0 1 1	T	dst	src1	src2

Описание: Результат поразрядного логического И операндов src1 и src2 загружается в регистр dst.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения.

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: AND3 *AR0—(IR0), *+AR1, R4

	Before Instruction	After Instruction
R4	00 0000 0000	00 0000 0020
AR0	80 98F4	80 98A4
AR1	80 9951	80 9951
IR0	50	50
LUF	0	0
LV	0	0
UF	0	0
N	0	0
Z	0	0
V	0	0
C	0	0
Data memory		
8098F4h	30	30
809952h	123	123

Пример 2: AND3 *-AR5, R7, R4

	Before Instruction	After Instruction
R4	00 0000 0000	00 0000 0002
R7	00 0000 0002	00 0000 0002
AR5	80 985C	80 985C
LUF	0	0
LV	0	0
UF	0	0
N	0	0
Z	0	0
V	0	0
C	0	0
Data memory		
80985Bh	0AFF	0AFF

AND3||STI Поразрядное логическое И и сохранить целое

Синтаксис: AND src2, src1, dst1
||STI src3, dst2

Операция: src1 AND src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, $0 \le n1 \le 7$)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, $0 \le n2 \le 7$)
src3 - регистр (Rn3, $0 \le n3 \le 7$)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
0 1 | 0 0 1 1 1	dst1 | src1	src3	dst2 | src2

Описание: Поразрядное логическое И и сохранение целого параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (AND3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой AND3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.
LV Не изменяется.
UF 0
N Старший значащий разряд выходного результата
Z 1 при генерации нулевого результата, иначе 0.
V 0
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:
```
AND3 *+AR1 (IR0), R4, R7
|| STI R3, *AR2
```

Before Instruction			After Instruction		
R0	00 0000 0008		R0	00 0000 0008	
R3	00 0000 0035	53	R3	00 0000 0035	53
R4	00 0000 A323		R4	00 0000 A323	
R7	00 0000 0000		R7	00 0000 0003	
AR1	80 99F1		AR1	80 99F1	
AR2	80 983F		AR2	80 983F	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
8099F9h	5C53		8099F9h	5C53	
80983Fh	0		80983Fh	35	53

ANDN Поразрядное логическое И (AND) с дополнением

Синтаксис: ANDN src, dst

Операция: dst AND ~src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод

31 24	23 16	15 8	7 0
0 00 0 0 0 1 0	G	dst	src

Описание: Поразрядное логическое И между операндом dst и поразрядным логическим дополнением (~) операнда src (→) загружается в регистр dst. Предполагается, что операнды dst и src являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного результата.

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: ANDN @980Ch, R2

Before Instruction		After Instruction	
R2	00 0000 0C2F	R2	00 0000 042D
DP	080	DP	080
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0
Data memory			
80980Ch	0A02	80980Ch	0A02

ANDN3 Поразрядное логическое И (AND) (с дополнением (3 операнда))

Синтаксис: ANDN3 src2, src1, dst

Операция: src1 AND ~src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, $0 \le n1 \le 27$)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 1	0 0 0 1 0 0 T	dst	src1	src2

Описание: Поразрядное логическое И между операндом src1 и поразрядным логическим дополнением (~) операнда src загружается в регистр dst. Предполагается, что операнды dst, src1 и src2 являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения- один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного результата.

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: ANDN3 R5, R3, R7

Before Instruction		After Instruction	
R3	00 0000 0C2F	R3	00 0000 0C2F
R5	00 0000 0A02	R5	00 0000 0A02
R7	00 0000 0000	R7	00 0000 042D
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0

Пример 2: ANDN3 R1, *AR5++ (IR0), R0

Before Instruction		After Instruction	
R0	00 0000 0000	R0	00 0000 0F30
R1	00 0000 00CF	R1	00 0000 00CF
AR5	80 9825	AR5	80 982A
IR0	5	IR0	5
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0
Data memory			
809825h	0FFF	809825h	0FFF

ASH Арифметический сдвиг

Синтаксис: ASH count, dst

Операция: Если (count ≥0):

dst << count → dst.

В противном случае:

dst >> |count| → dst.

Операнды: основные режимы адресации count (по счетчику) (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод

31 24	23 16	15 8	7 0
0 00 0 0 0 1 1 1	G	dst	count

Описание: Младшие 8 значащих разрядов оператора count используются для генерации двоичного дополнения счетчика сдвига до 32. Если операнд count больше нуля, операнд dst сдвигается влево на величину операнда count. При сдвиге младшие разряды заполняются нулями, а старшие сдвигаются в разряд переноса C регистра состояния. Арифметический левый сдвиг: C← dst ← 0. Если операнд count меньше 0, операнд dst сдвигается вправо на абсолютное значение операнда count. Старшие разряды числа сдвигаются с расширением знака вправо. Младшие разряды сдвигаются в разряд переноса C регистра состояния. Арифметический сдвиг право: знак dst → dst → C. Если операнд count равен 0, сдвиг не происходит, и разряд C (перенос) устанавливается в 0. Операнды count и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения - один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N Старший значащий разряд выходного значения.

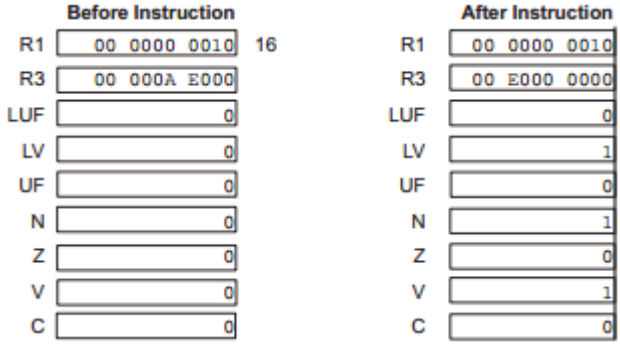
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

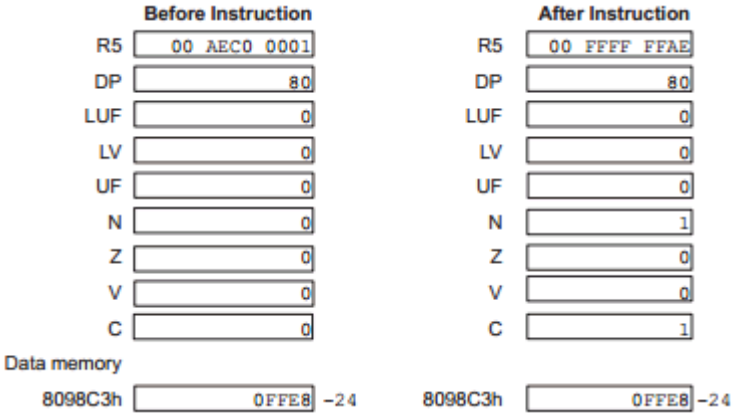
C Устанавливается по значению последнего сдвигаемого вовне разряда. Равен 0, если счетчик сдвига 0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: ASH R1, R3



Пример 2: ASH @98C3h, R5



ASH3 Арифметический сдвиг (3 операнда)

Синтаксис: ASH3 count, src, dst

Операция: Если (count ≥ 0):

src << count → dst

В противном случае:

src >> |count| → dst

Операнды: Трехоперандные режимы адресации count (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 1	0 0 0 1 0 1	T	dst	src	count

Описание: Семь младших значащих разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига до 32 разрядов. Если операнд count больше 0, операнд src сдвигается влево на число разрядов, определенных операндом count. Младшие разряды при сдвиге заполняются нулями, старшие сдвигаются в разряд переноса C регистра состояния. Арифметический левый сдвиг: C ← src ← 0.

Если операнд count меньше 0, операнд src сдвигается справа на число разрядов, определенных абсолютным значением операнда count. Старшие разряды операнда src сдвигаются вправо с расширением знака. Младшие разряды сдвигаются в разряд переноса C регистра состояния. Арифметический сдвиг вправо: знак src → src → C.

Если операнд count равен 0, сдвиг не происходит, и разряд C (перенос) устанавливается в 0. Операнды count, src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, иначе не изменяется.

UF 0

N Старший значащий разряд выходного результата

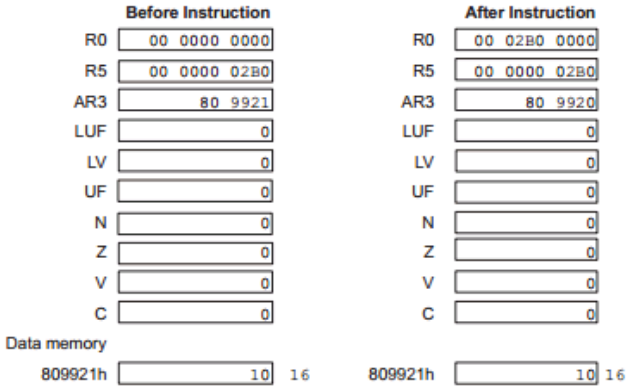
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

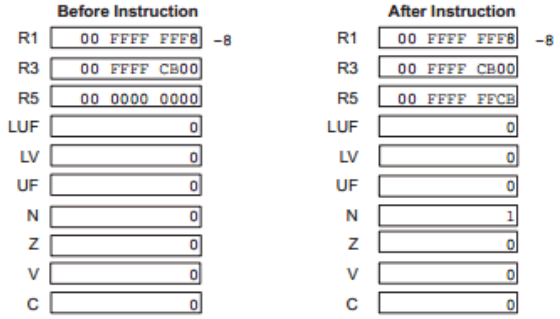
C Устанавливается по значению последнего сдвигаемого вовне разряда. Равен 0, если count = 0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: ASH3 *AR3-- (1), R5, R0



Пример 2: ASH3 R1, R3, R5



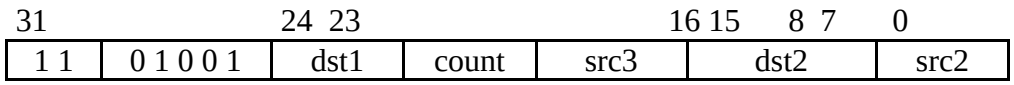
ASH3||STI Арифметический сдвиг и целое

Синтаксис: ASH3 count, src2, dst1
|| STI src3, dst2

Операция: Если (count ≥ 0):
src2 << count → dst1
В противном случае:
src2 >> |count| → dst1
|| src3 → dst2

Операнды: count - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод



Описание: Семь младших значащих разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига до 32 разрядов. Если операнд count больше 0, операнд src сдвигается влево на число разрядов, определенных операндом count. Младшие разряды при сдвиге заполняются нулями, старшие сдвигаются в разряд переноса C регистра состояния. Арифметический левый сдвиг: C←src2←0. Если операнд count меньше 0, операнд src сдвигается справа на число разрядов, определенных абсолютным значением операнда count. Старшие разряды операнда src сдвигаются вправо с расширением знака. Младшие разряды сдвигаются в разряд переноса C регистра состояния. Арифметический сдвиг вправо: знак src2 → src2 → C. Если операнд count равен 0, сдвиг не происходит, и разряд C (перенос) устанавливается в 0. Операнды count и dst являются целыми со знаком. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (ASH3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой ASH3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7-R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, иначе не изменяется.

UF 0

N Старший значащий разряд выходного результата

Z 1 при генерации нулевого результата, иначе 0.

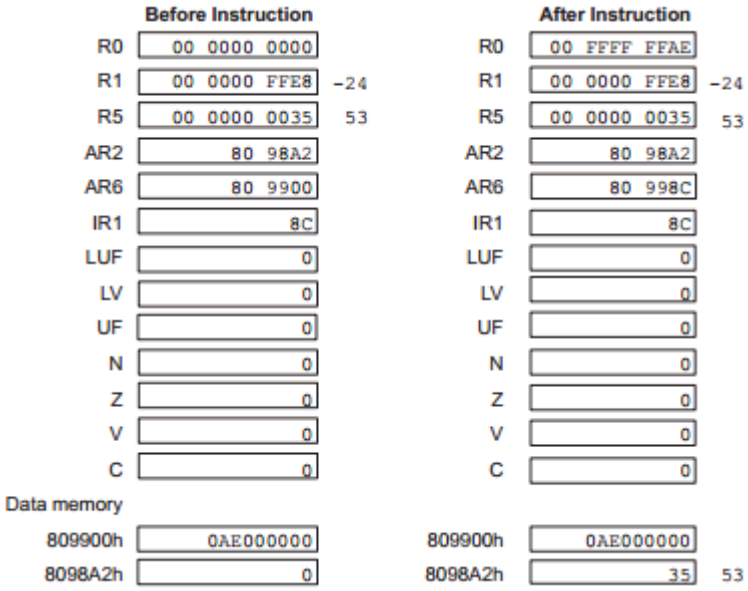
V 1 при возникновении целочисленного переполнения, иначе 0.

C Устанавливается по значению последнего сдвигаемого вовне разряда. Равен 0, если count = 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: ASH3 R1, *AR6++ (IR1), R0

|| STI R5, *AR2



Bcond Условный переход (стандартный)

Синтаксис: Bcond src

Операция: Если cond - истина: Если src в регистровом режиме адресации (Rn, 0≤n≤27), src → PC.

Если src в режиме адресации относительно PC (метка или адрес), смещение + PC + 1 → PC. В противном случае продолжение.

Операнды: Режимы адресации условного перехода src (B):

0 - регистровая

1 - относительно PC

Опкод:

31 24	23	22 16	15 8	7 0
0 1 1 0 1 0	B	0 0 0 0	0 cond	регистр или смещение

Описание: Bcond означает стандартный переход, который выполняется за четыре цикла. Переход осуществляется, если условие истинно (т.к. конвейер при этом должен быть очищен; см. подраздел 9.2). Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, ассемблер генерирует смещение; смещение = метка - (PC команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова

команды перехода. Смещение добавляется к PC команды перехода + 1 для генерации нового PC.

В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7-R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: BZ R0

Before Instruction	
R0	00 0003 FF00
PC	2B00
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

After Instruction	
R0	00 0003 FF00
PC	3 FF00
LUF	0
LV	0
UF	0
N	0
Z	1
V	0
C	0

BcondD Условный переход (с задержкой)

Синтаксис: BcondD src

Операция: Если cond - истина:
Если src в регистровом режиме адресации (Rn, 0≤n≤27), src → PC.
Если src в режиме адресации относительно PC (метка или адрес), смещение + PC + 3 → PC. В противном случае продолжение.

Операнды: Режимы адресации условного перехода src (B):
0 - регистровая
1 - относительно PC

Опкод:

31 24	23	22 16	15 8	7 0
0 1 1 0 1 0	B	0 0 0	1	cond	регистр или смещение

Описание: BcondD означает задержанный переход, при котором обеспечивается выборка трех команд до изменения PC. В результате получается одноцикловый переход, и три команды, следующие за BcondD, не влияют на cond. Переход осуществляется, если условие истинно. Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адре-

сации относительно PC, ассемблер генерирует смещение; смещение = метка – (PC команды перехода + 3). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды перехода. Смещение добавляется к PC команды перехода + 3 для генерации нового PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7-R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: BNZD 36 (36 = 24h)

Before Instruction	
PC	0050
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

After Instruction	
PC	0077
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

BR Безусловный переход (стандартный)

Синтаксис: BR src

Операция: src → PC

Операнды: src - режим длинной непосредственной адресации

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 0	0	src	

Описание: BR обеспечивает стандартный переход, который выполняется за четыре цикла, т. к. в течение выполнения перехода происходит очистка конвейера (см. подраздел 9.2). Операнд src является 24-разрядным целым без знака. Примечание – Для стандартного перехода разряд 24 равен 0.

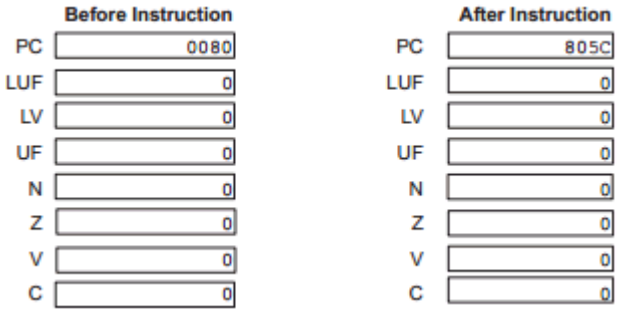
Циклы: 4

Разряды состояния: LUF Не изменяется.

LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: BR 805Ch



BRD Безусловный переход (с задержкой)

Синтаксис: BRD src

Операция: src → PC

Операнды: src - режим длинной непосредственной адресации

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 0	1	src	

Описание: BRD означает задержанный переход, при котором обеспечивается выборка трех команд после задержанного перехода до изменения PC. В результате получается одноцикловый переход. Операнд src является 24-разрядным целым без знака.

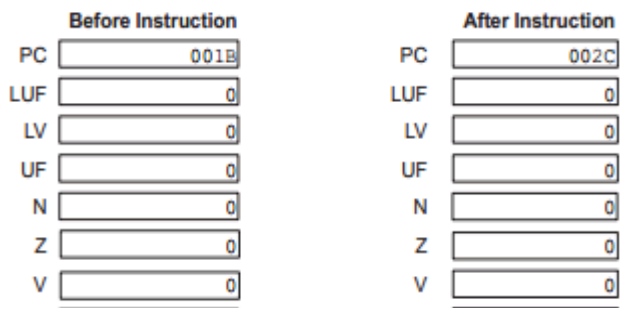
Примечание: для задержанного перехода разряд 24 равен 1.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: BRD 2Ch



CALL Вызов подпрограммы

Синтаксис: CALL src

Операция: Следующий PC → *++SP

Операнды: src - режим длинной непосредственной адресации

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 1	0	src	

Описание: Осуществляется вызов подпрограммы. Следующее значение PC записывается в системный стек. Операнд src загружается в PC. Операнд src является 24-разрядным непосредственным операндом без знака.

Циклы: 4

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

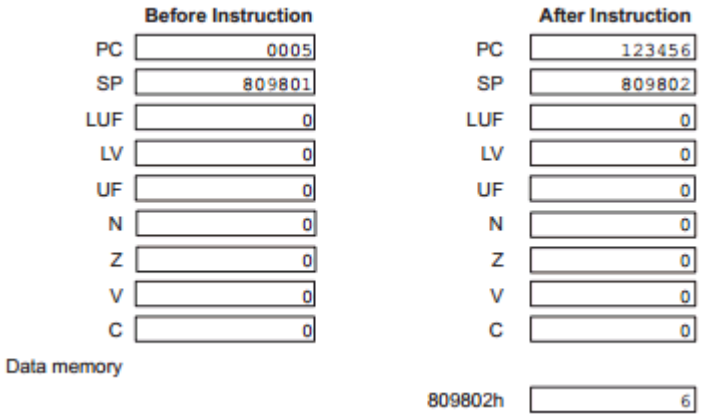
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: CALL 123456h



CALLcond Условный вызов подпрограммы

Синтаксис: CALLcond src

Операция: Если cond - истина:

Следующий PC → *++SP

Если src в регистровом режиме адресации (Rn, 0≤n≤27), src → PC.

Если src в режиме адресации относительно PC (метка или адрес),

смещение + PC + 1 → PC.

В противном случае продолжение.

Операнды: Режимы адресации условного перехода src (B):

0 - регистровая

1 - относительно PC

Опкод:

31 24	23	22 16	15 8	7 0
0 1 1 1 0 0	B	0 0 0 0	cond	регистр или смещение

Описание: Вызов осуществляется, если условие истинно. Если условие истинно, следующее значение PC загружается в системный стек. Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, ассемблер генерирует смещение; смещение = метка - (PC команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды вызова. Смещение добавляется к PC команды перехода + 1 для генерации нового PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой командой (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 5

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: CALLNZ R5

Before Instruction		After Instruction	
R5	00 0000 0789	R5	00 0000 0789
PC	0123	PC	0789
SP	809835	SP	809836
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0
Data memory			
		809836h	124

CMPF Сравнение значений с плавающей запятой

Синтаксис: CMPF src, dst

Операция: dst → src

Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, $0 \le n \le 7$)
01 - прямая
10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 0 0	G	dst	src

Описание: Операнд src вычитается из операнда dst. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значений каких бы то ни было операндов. Предполагается, что операнды dst и src являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются для всех регистров назначения (R27 - R0).

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

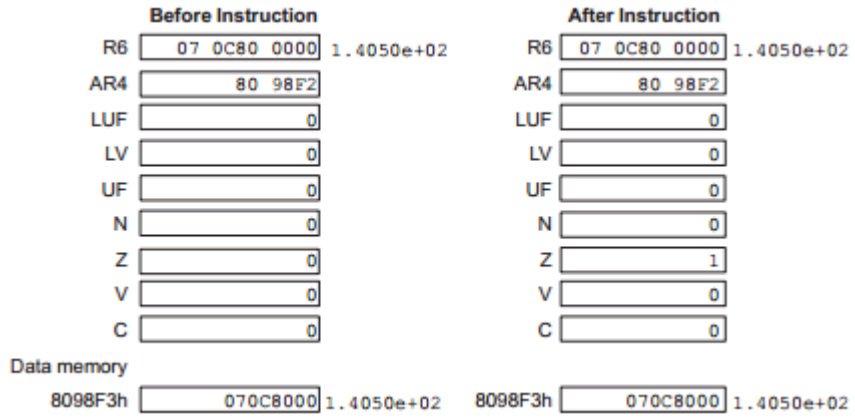
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: CMPF *+AR4, R6



CMPF3 Сравнение значений с плавающей запятой (3 операнда)

Синтаксис: CMPF3 src2, src1

Операция: src1 - src2

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 0 0 0 1 1 0	T 0 0 0 0 0	src1	src2

Описание: Операнд src2 вычитается из операнда src1. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значений каких бы то ни было операндов. Предполагается, что операнды dst и src являются числами в формате с ПЗ. Хотя команда имеет только два операнда, она имеет вид трехоперандной, т. к. операнды определены в трехоперандном формате.

Циклы: 1

Разряды состояния: Флаги состояния изменяются для всех регистров назначения (R27 - R0).

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

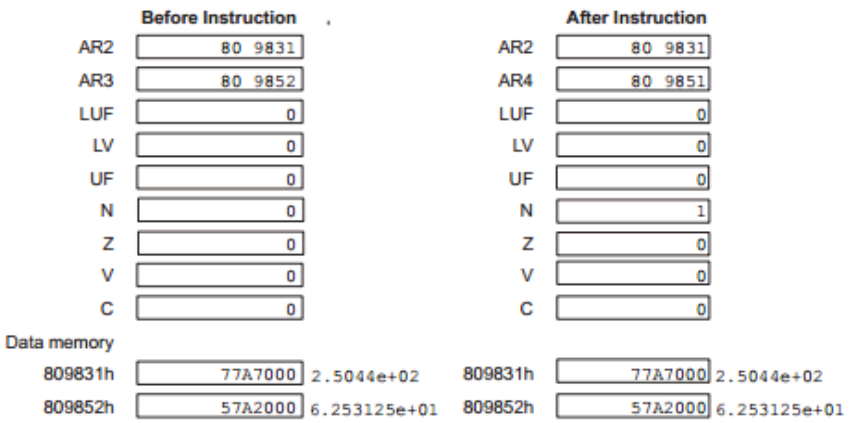
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: CMPF3 *AR2, *AR3-- (1)



CMPI Сравнение целых

Синтаксис: CMPI src, dst

Операция: dst - src

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 0 1	G	dst	src

Описание: Операнд src вычитается из операнда dst. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значения каких бы то ни было операндов. Предполагается, что операнды dst и src являются целыми без знака.

Циклы: 1

Разряды состояния: Флаги состояния изменяются для всех регистров назначения (R27 - R0).

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

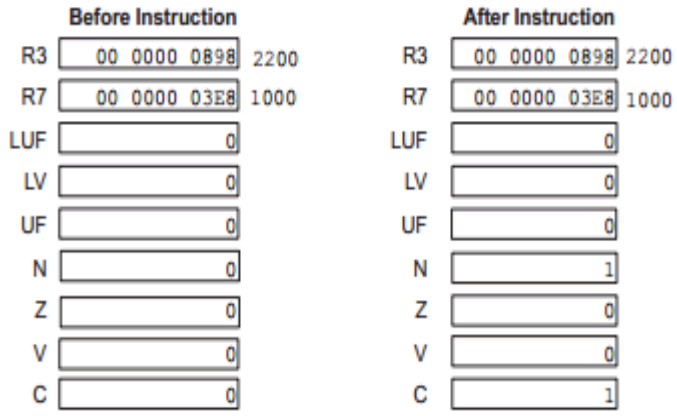
V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
CMPI    R3, R7
```



CMPI3 Сравнение целых (3 операнда)

Синтаксис: CMPI3 src2, src1

Операция: src1 - src2

Операнды Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 0 0 0 1 1 1 | T	0 0 0 0 0	src1	src2

Описание: Операнд src2 вычитается из операнда src1. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значения каких бы то ни было операндов. Предполагается, что операнды dst и src являются целыми со знаком. Хотя команда имеет только два операнда, она имеет вид трехоперандной, т.к. операнды определены в трехоперандном формате.

Циклы: 1

Разряды состояния: Флаги состояния изменяются для всех регистров назначения (R27-R0).

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

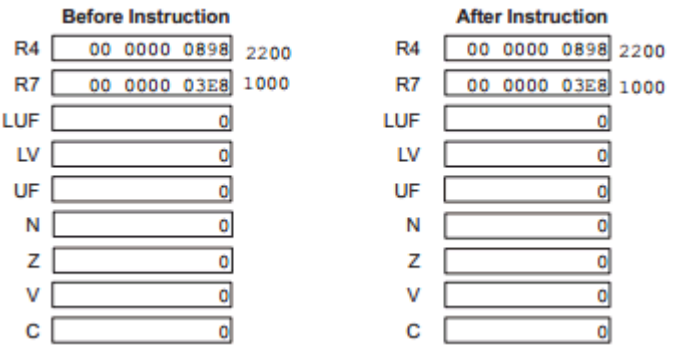
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: CMPI3 R7, R4



DBcond Декремент и условный переход (стандартный)

Синтаксис: DBcond ARn, src

Операция: ARn - 1 → ARn

Если cond - истина и ARn≥0:

Если src в регистровом режиме адресации (Rn, 0≤ n ≤27), src → PC.

Если src в режиме адресации относительно PC (метка или адрес),

смещение + PC + 1 → PC.

В противном случае продолжение.

Операнды: Режимы адресации условного перехода src (B):

0 - регистровая

1 - относительно PC

ARn - регистр (0 ≤ n ≤ 7)

Опкод:

31 24	23	22 21	20 16	15 8	7 0
0 1 1 0 1 1	B	ARn	0	cond	регистр или смещение

Описание: DBcond означает стандартный переход, который выполняется за четыре цикла, т.к. конвейер должен быть очищен, если cond – «истина». Определенный вспомогательный регистр декрементируется, и выполняется переход, если cond – «истина», и если значение в определенном вспомогательном регистре больше или равно 0. Флаги условий те же самые, которые установлены последней командой, влияющей на разряды состояния. Вспомогательный регистр интерпретируется как 24-разрядное знаковое целое. Самые старшие 8 разрядов не изменяются при декрементировании. При сравнении вспомогательного регистра используется 24 младших разряда вспомогательного регистра. Обратите внимание, что условие перехода не зависит от декремента вспомогательного регистра.

Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, ассемблер генерирует смещение; смещение = метка – (PC команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды перехода. Смещение добавляется к PC команды перехода + 1 для генерации нового PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 4

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

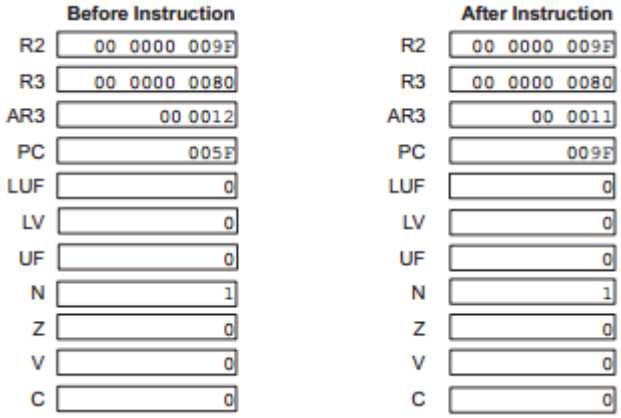
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:
```
CMPI    200, R3
DBLT    AR3, R2
```



DBcondD Декремент и условный переход (задержанный)

Синтаксис: DBcondD ARn, src

Операция: ARn - 1 → ARn

Если cond - истина и ARn ≥ 0:

Если src в регистровом режиме адресации (Rn, 0 ≤ n ≤ 27), src → PC.

Если src в режиме адресации относительно PC (метка или адрес), смещение + PC + 3 → PC. Иначе: продолжение.

Операнды: Режимы адресации условного перехода src (B):

0 - регистровая

1 - относительно PC

ARn - регистр (0 ≤ n ≤ 7)

Опкод:

31 24	23	16	15 8	7 0
0 1 1 0 1 1 | B | ARn	1	cond	регистр или смещение	

Описание: DBcondD означает задержанный переход, который выполняется за один цикл в результате того, что позволяется осуществить выборку трех команд до того, как изменится PC. Определенный вспомогательный регистр декрементируется, и выполняется переход, если cond – «истина», и если значение в определенном вспомогательном регистре больше или равно 0. Флаги условий те же самые, которые установлены последней командой, влияющей на разряды состояния. Три команды, следующие за DBcondD, не влияют на разряды состояния. Вспомогательный регистр интерпретируется как 24-разрядное знаковое целое. Самые старшие 8 разрядов не изменяются при декрементировании. При

сравнении вспомогательного регистра используется 24 младших разряда вспомогательного регистра. Обратите внимание, что условие перехода не зависит от декремента вспомогательного регистра. Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, ассемблер генерирует смещение; смещение = метка – (PC команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды перехода. Смещение добавляется к PC команды перехода + 1 для генерации нового PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:
```
CMPI    0, R2
DBZD AR5, $+110h
```

	Before Instruction	After Instruction
R2	00 0000 0026	00 0000 0026
AR5	00 0067	00 0066
PC	0100	0210
LUF	0	0
LV	0	0
UF	0	0
N	0	0
Z	0	1
V	0	0
C	0	0

FIX Преобразование значений с плавающей запятой в целое

Синтаксис: FIX src, dst

Операция: fix(src) → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 0 1 0 1 0	G	dst	src

Описание: Операнд src в формате с ПЗ преобразуется к ближайшему целому, меньшему или равному по значению, и результат записывается в регистр dst. Операнд src имеет значение в формате с ПЗ, операнд dst – целое со знаком. Поле экспоненты регистра результата (если он один) не изменяется. Целочисленное переполнение возникает, когда число в формате с ПЗ слишком велико, чтобы быть представлено как 32-разрядное в дополнительном коде. В случае целочисленного переполнения результат будет заполнен в направлении переполнения.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения - один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

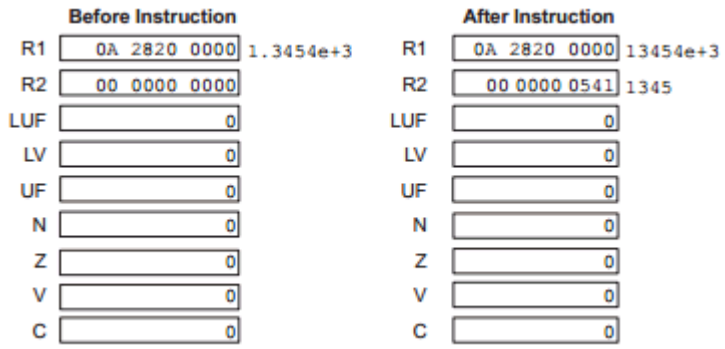
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: FIX R1, R2



FIX||STI Преобразовать значение числа в формате с ПЗ в целое и сохранить целое

Синтаксис: FIX src2, dst1
|| STI src3, dst2

Операция: fix(src2) → dst1
|| src3 → dst2

Операнды : src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, $0 \le n1 \le 7$)
src3 - регистр (Rn2, $0 \le n2 \le 7$)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23	16	15 8	7 0
1 1 0 1 0 1 0	dst1	0 0 0	src3	dst2 src2

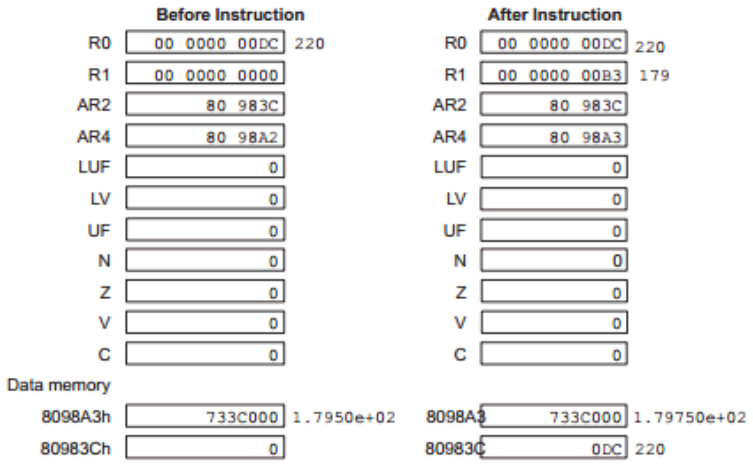
Описание: Преобразование числа в формате с ПЗ в целое. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (FIX), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой FIX. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2. Целочисленное переполнение возникает, кода число в формате с ПЗ слишком велико, чтобы быть представлено как 32-разрядное в дополнительном коде. В случае целочисленного переполнения результат будет заполнен в направлении переполнения.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:
```
FIX     *++AR4 (1), R1
|| STI R0, *AR2
```



FLOAT Преобразование целых значений в значения с плавающей запятой

Синтаксис: FLOAT src, dst

Операция: float(src) → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 1 1	G	dst	src

Описание: Целочисленный операнд src преобразуется в значение в формате с ПЗ, равное ему, и записывается в регистр dst. Операнд src является целым числом со знаком, операнд dst – число в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения - один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

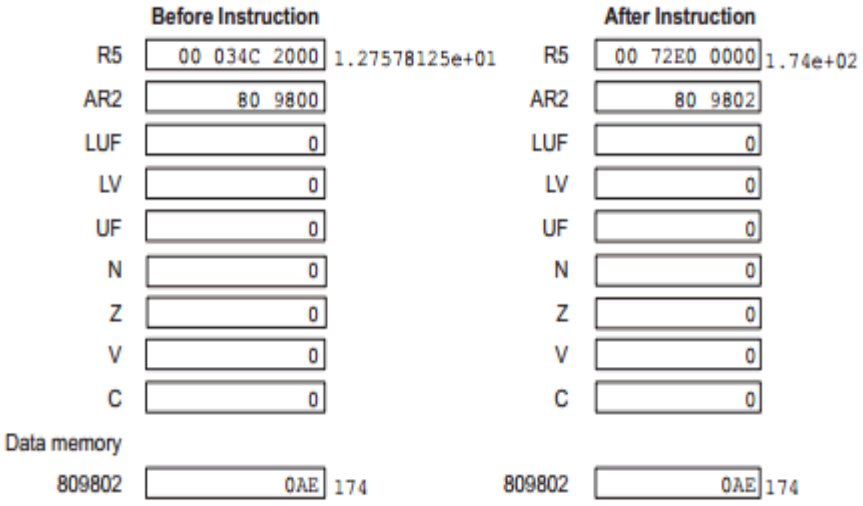
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: FLOAT *++AR2 (2), R5



FLOAT||STF Преобразовать целое в значение числа в формате с ПЗ и сохранить в формате с ПЗ

Синтаксис: FLOAT src2, dst1
|| STF src3, dst2

Операция: float(src2) → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
1 1 0 1 1 1 dst1	0 0 0 src3	dst2	src2

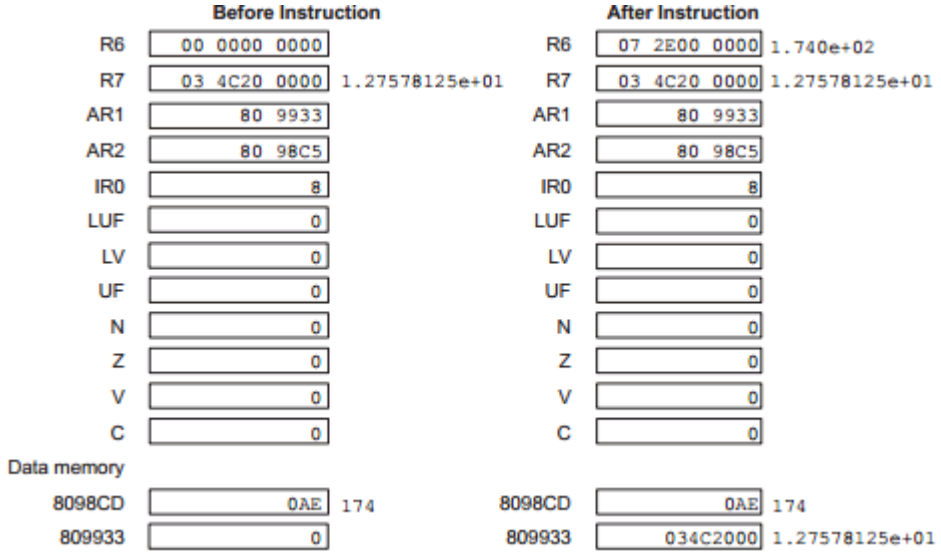
Описание: Выполняется преобразование из целого в формат с ПЗ. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (FLOAT), записывает в тот же регистр, STF имеет на входе содержимое регистра до того, как оно было изменено командой FLOAT. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV Не изменяется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 0
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:
```
FLOAT *+AR2 (IR0), R6
|| STF R7, *AR1
```



IACK Подтверждение прерывания

Синтаксис: IACK src

Операция: Выполняет пустую операцию чтения с IACK = 0.
По завершении пустого чтения устанавливает IACK в 1.

Операнды: основные режимы адресации src (G):
01 - прямая
10 - косвенная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 1 1 0	G	0 0 0 0 0	src

Описание: Выполняется пустая операция чтения с IACK = 0. По завершении пустого чтения IACK устанавливается в 1. Эта команда может быть использована для подтверждения внешнего прерывания. Если указанный адрес относится к внекристальной памяти, осуществляется операция чтения по данному адресу. Сигнал IACK и адрес могут использоваться для сигнализации подтверждения прерывания внешним устройствам. Чтение данных процессором не используется.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: IACK *AR5

Before Instruction	
$\overline{IACK}$	1
PC	300
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

After Instruction	
$\overline{IACK}$	1
PC	301
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

IDLE Ожидание прерывания

Синтаксис: IDLE

Операция: 1 → ST (GIE)
Следующий PC → PC
Ожидание до прерывания

Операнды: Нет

Опкод:

31 24	23 16	15 8	7 0
0 0 0	0 0 1 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Описание: Устанавливается глобальное разрешение прерывания, следующее значение PC загружается в PC и ЦПУ ожидает получения прерывания. Когда прерывание получено, содержимое PC загружается в активный системный стек.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

LDE Загрузка значения экспоненты с плавающей запятой

Синтаксис: LDE src, dst

Операция: src(exp) → dst(exp)

Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, $0 \le n \le 7$)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0	0 0 1 1 0 1	G dst	src

Описание: Поле экспоненты операнда src загружается в поле экспоненты регистра dst. Поле мантиссы регистра dst не изменяется, если только значение загруженной экспоненты не является зарезервированным для нуля значением экспоненты, что определяется точностью операнда src. Тогда поле мантиссы операнда dst устанавливается в ноль. Операнды src и dst являются числами в формате с ПЗ. Непосредственные значения выражаются в коротком формате с ПЗ.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.

N Не изменяется.

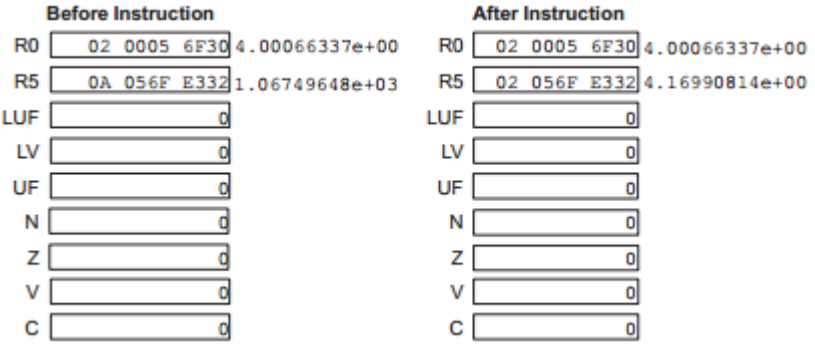
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDE R0, R5



LDF Загрузка значения с плавающей запятой

Синтаксис: LDF src, dst

Операция: src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 0 0 1 1 1 0	G	dst	src

Описание: Операнд src загружается в регистр dst. Операнды src и dst являются числами в формате с плавающей запятой.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

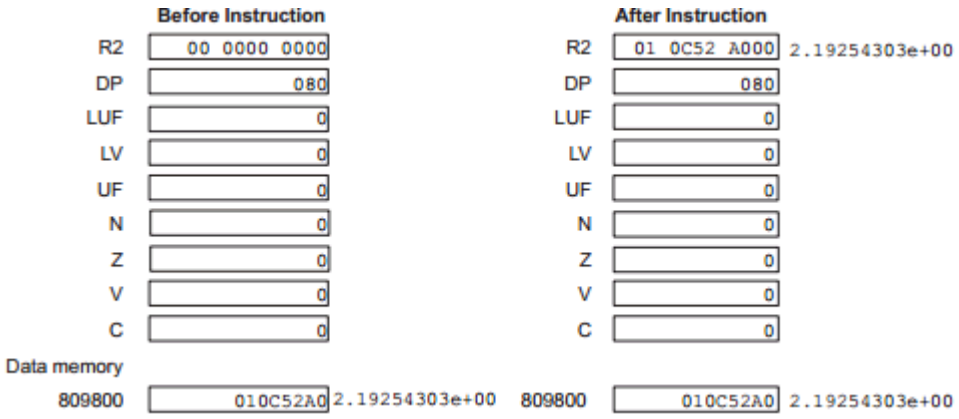
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDF @9800h, R2



LDFcond Условная загрузка значения с плавающей запятой

Синтаксис: LDFcond src, dst

Операция: Если cond – «истина»:

src → dst.

Иначе:

dst не изменяется

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 1 0 0 cond	G dst	src	

Описание: Если cond – «истина», операнд src загружается в регистр dst. В противном случае регистр dst не изменяется. Операнды src и dst являются числами в формате с ПЗ. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Обратите внимание, что команда LDFU (загрузить ПЗ-число безусловно) используется для загрузки R7-R0 без влияния на флаги условий. Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

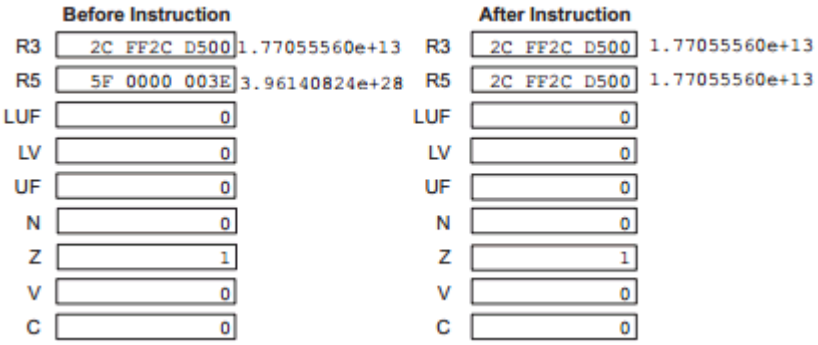
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDFZ R3, R5



LDFI Загрузка значения с плавающей запятой, блокировка. Сигнализирует об операции блокировки

Синтаксис: LDFI src, dst

Операция: Сигнализирует об операции блокировки.

src → dst

Операнды: основные режимы адресации src (G):

01 - прямая

10 - косвенная

dst - регистр (Rn, $0 \leq n \leq 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 1 1 1 1	G	dst	src

Описание: Операнд src загружается в регистр dst. Операция блокировки (межпроцессорного взаимодействия) сигнализируется через XF0 и XF1. Операнды src и dst являются числами в формате с ПЗ. Обратите внимание, что определена только прямая и косвенная адресация. Детальное описание см. в подразделе 6.4.

Циклы: 1, если XF1 = 0 (см. подраздел 6.4)

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

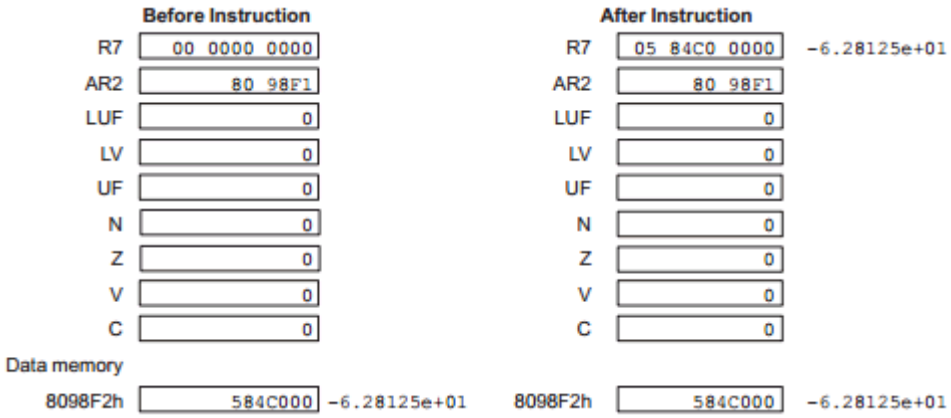
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDFI *+AR2, R7



LDF||LDF Загрузить значение в формате с ПЗ

Синтаксис: LDF src2, dst2
|| LDF src1, dst1

Операция: src2 → dst2
|| src1 → dst1

Операнды: src1 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, $0 \le n1 \le 7$)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst2 - регистр (Rn2, $0 \le n2 \le 7$)

Опкод

31 24	23 16	15 8	7 0
1 1 0 0 0 1 0	dst2 dst1	0 0 0	src1 src2

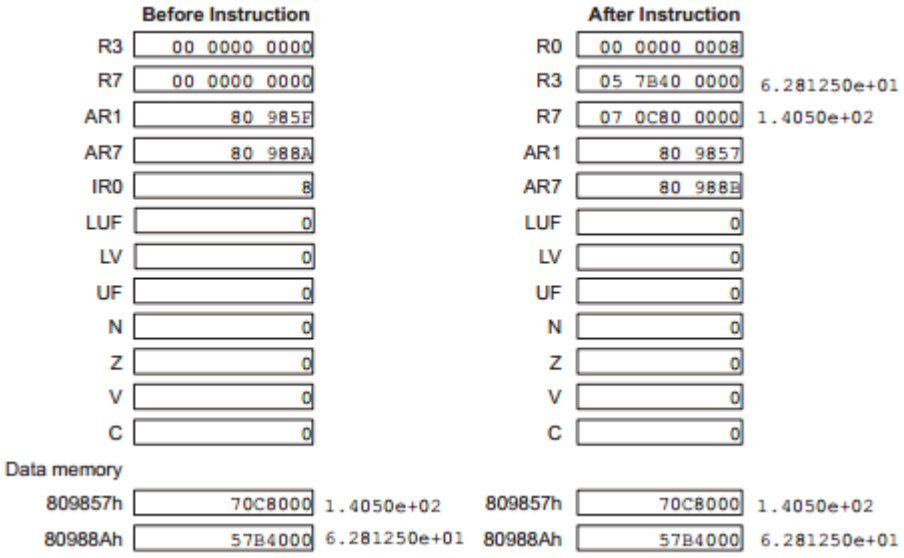
Описание: Загрузка двух чисел выполняется параллельно. Если обе LDF загружают в один регистр, ассемблер выдает предупреждение. В результате этого получается LDF src2, dst2

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:
```
LDF     *--AR1 (IR0), R7
|| LDF   *AR7++ (1), R3
```



LDF||STF Параллельное выполнение LDF и STF

Синтаксис: LDF src2, src1
|| STF src3, dst2

Операция: src2 → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31	24	23	16	15	8	7	0
1 1	0 1 1 0 0	dst1	0 0 0	src3	dst2		src2

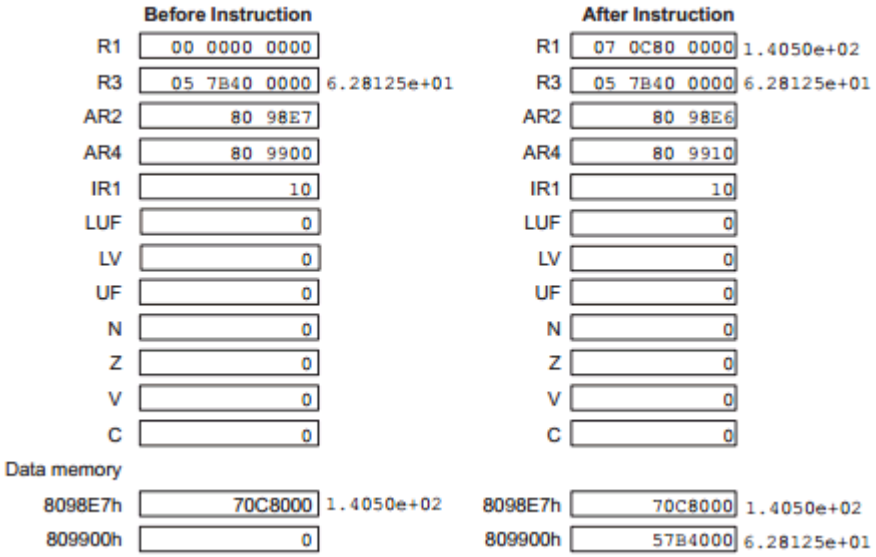
Описание: Параллельно выполняются операции загрузки и сохранения чисел в формате с ПЗ. Если операнды src2 и dst2 указывают на один и тот же адрес, src2 считывается до того, как будет записан dst2.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDF *AR2-- (1), R1
|| STF R3, *AR4++ (IR1)



LDI Загрузка целого

Синтаксис: LDI src, dst

Операция: src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 0 0 0	G	dst	src

Описание: Операнд src загружается в регистр dst. Операнды src и dst являются целыми со знаком. Альтернативная форма LDI, LDP, используется для загрузки регистра указателя страницы памяти (DP) или любого другого регистра восьмью старшими разрядами перемещаемого адреса. Смотри команду LDP и 10.3.2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

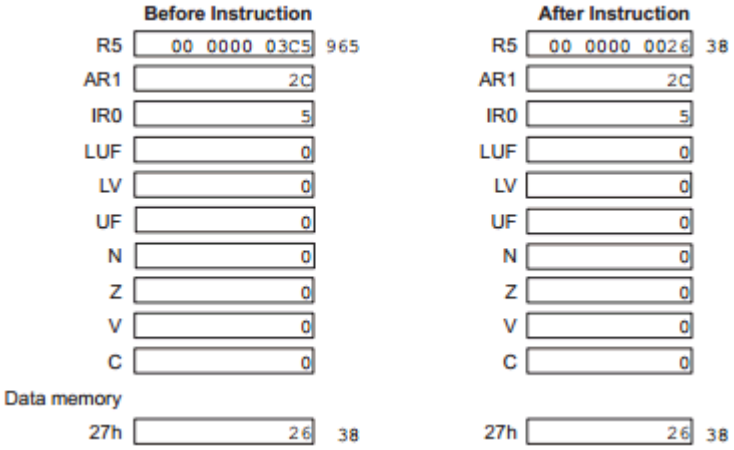
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDI *-AR1 (IR0), R5



LDIcond Условная загрузка целых

Синтаксис: LDIcond src, dst

Операция: Если cond - "истина":

src → dst

Иначе:

dst не изменяется

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 1 0 1 cond	G dst	src	

Описание: Если cond – «истина», операнд src загружается в регистр dst. В противном случае регистр dst не изменяется. Вне зависимости от условия, имеет место чтение регистра src. Операнды src и dst являются целыми со знаком. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Обратите внимание, что команда LDIU (безусловная загрузка целого) используется для загрузки R7-R0 без изменения флагов условий. Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

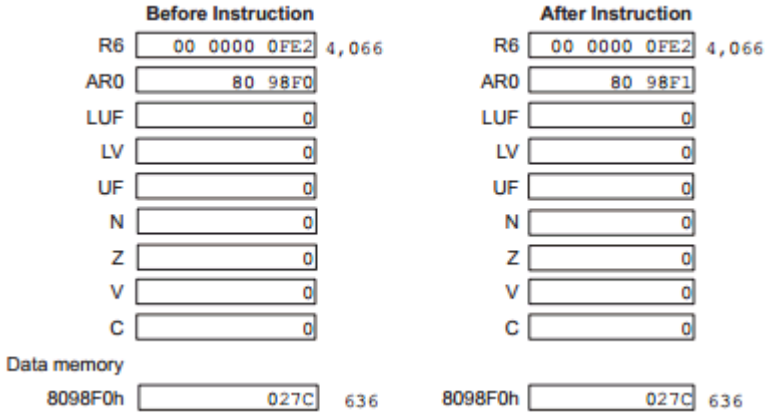
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: LDIZ R4, R6



LDII Загрузка целых, блокировка. Сигнализирует о команде блокировки

Синтаксис: LDII src, dst

Операция: Сигнализирует об операции блокировки.

src → dst

Операнды: основные режимы адресации src (G):

01 - прямая

10 - косвенная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 0 0 1	G	dst	src

Описание: Операнд src загружается в регистр dst. Операция блокировки (межпроцессорного взаимодействия) сигнализируется через XF0 и XF1. Операнды src и dst являются целыми со знаком. Необходимо обратить внимание, что определены только прямая и косвенная адресации. Детальное описание см. в подразделе 6.4.

Циклы: 1, если XF1 = 0 (см. подраздел 6.4)

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

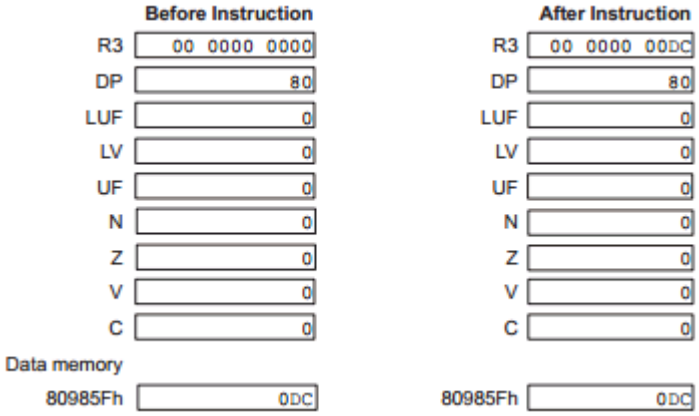
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDII @985Fh, R3



LDI||LDI Загрузить целое

Синтаксис: LDI src2, dst2
|| LDI src1, dst1

Операция: src2 → dst2
|| src1 → dst1

Операнды: src1 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, $0 \le n1 \le 7$)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst2 - регистр (Rn2, $0 \le n2 \le 7$)

Опкод

31 24	23 16	15 8	7 0
1 1 0 0 0 1 1	dst2	dst1	0 0 0 src1	src2

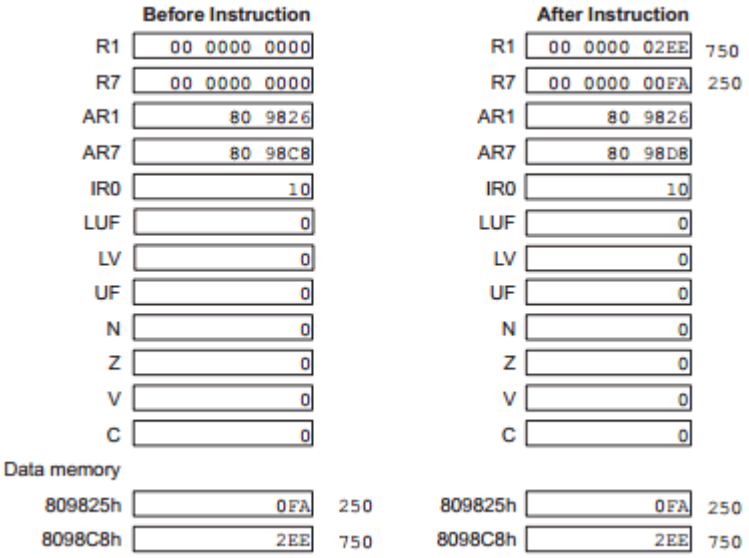
Описание: Загрузка двух целых выполняется параллельно. Если обе LDI загружают в один регистр, ассемблер выдает предупреждение. В результате этого получается LDI src2, dst2.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDI *--AR1 (1), R7
|| LDI *AR7++ (IR0), R1



LDI||STI Загрузить целое и сохранить целое

Синтаксис: LDI src2, src1
|| STI src3, dst2

Операция: src2 → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
1 1 0 1 1 0 1	dst1	0 0 0	src3	dst2	src2

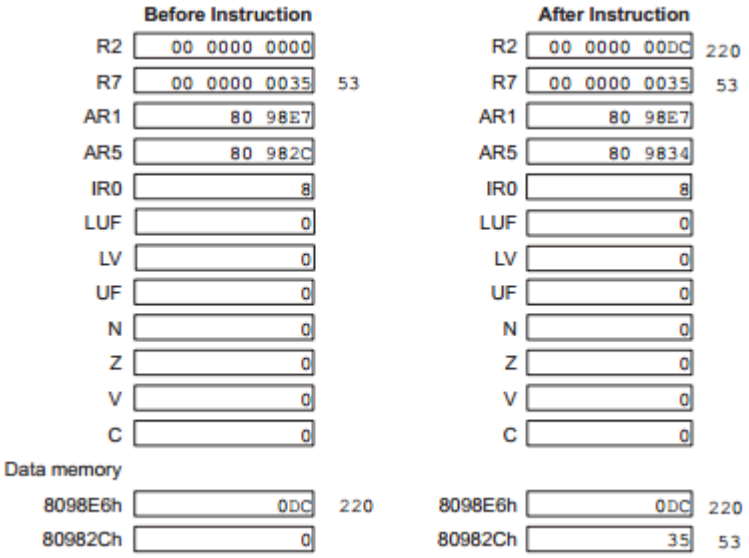
Описание: Параллельно выполняются операции загрузки и сохранения целых. Если операнды src2 и dst2 указывают на один и тот же адрес, src2 считывается до того, как будет записан dst2.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:
```
LDI     *-AR1 (1), R2
|| STI   R7, *AR5++ (IR0)
```



LDM Загрузка мантиссы с плавающей запятой

Операция: src(man) → dst(man)

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 0 1 0	G	dst	src

Описание: Поле мантиссы операнда src загружается в поле мантиссы регистра dst. Поле экспоненты dst не изменяется. Операнды src и dst являются числами в формате с ПЗ. Если операнд src загружается из памяти, соответствующее содержимое памяти загружается как мантисса. При использовании непосредственного режима адресации разряды 15-12 слова команды устанавливаются в 0 ассемблером.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

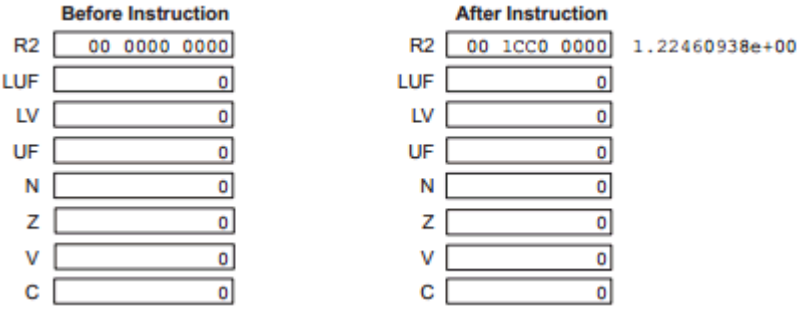
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDM 156.75, R2 (156.75 = 071CC00000h)



LDP Указатель загрузки данных страницы

Синтаксис: LDP src, DP

Операция: src → Указатель страницы данных.

Операнды: src - 8 старших разрядов абсолютного 24-разрядного адреса источника (src). ",DP" в операнде является дополнительным.

dst регистр (Rn, 0 ≤ n ≤ 7)

Опкод

31	24	23	16	15	8	7	0
0 0 0	0 1 0 0 0 0	1	1 0 0 0 0	0 0 0 0 0 0 0 0		src2	

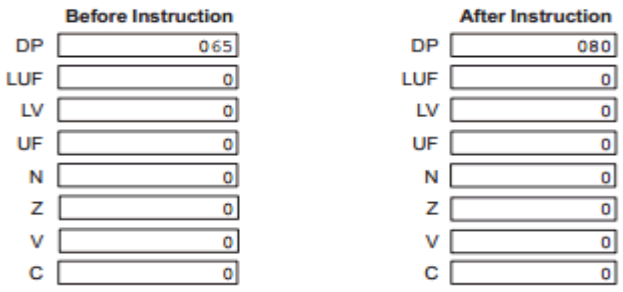
Описание: Эта псевдооперация является альтернативной формой команды LDI, за исключением того, что команда LDP определена только для непосредственного режима адресации. Поле операнда src содержит восемь старших разрядов абсолютного 24-разрядного адреса src (т. е. только 24-16 разряды src используются). Эти восемь разрядов загружены в младшие восемь разрядов указателя страницы данных. Младшие восемь разрядов указателя используются в прямой адресации как указатели страницы данных, к которой будет осуществляться адресация. Всего имеется 256 страниц, каждая из которых размером 64К слов. Разряды 31-8 указателя зарезервированы и хранятся как 0.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDP @809900h, DP
или
LDP @809900h



LSH Логический сдвиг

Синтаксис: LSH count, dst

Операция: Если count ≥0:

dst << count → dst.

Иначе:

dst >> |count| → dst.

Операнды: основные режимы адресации count (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 29	24 23		16	15 8	7 0
0 0 0	0 1 0 0 1 1	G	dst	count	

Описание: Семь младших разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига. Если операнд count больше 0, операнд dst сдвигается влево на количество разрядов, определенное значением операнда count. Младшие разряды заполняются нулями, старший разряд сдвигаем переносится в разряд C (carry) регистра состояния. Логический сдвиг влево: C ← dst ←0. Если операнд count меньше 0, dst сдвигается вправо на число разрядов, определенное абсолютным значением операнда count. Старшие разряды операнда dst заполняются 0 при сдвиге вправо. Младшие разряды сдвигаются вовне через разряд переноса C. Логический сдвиг вправо: 0 → dst → C. Если операнд count = 0, сдвиг не происходит и разряд переноса C устанавливается в 0. Предполагается, что операнд count является целым со знаком, а оператор dst – беззнаковое целое.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения.

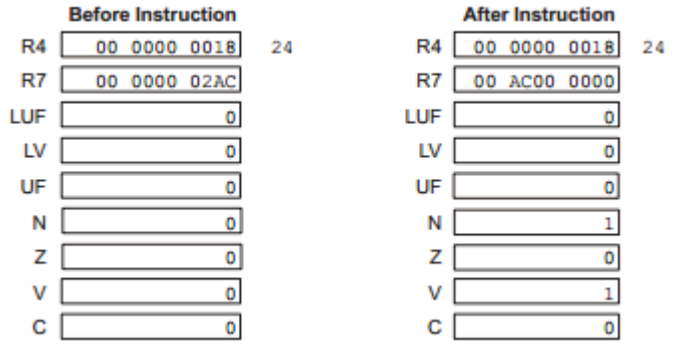
Z 1 при генерации нулевого результата, иначе 0.

V 0

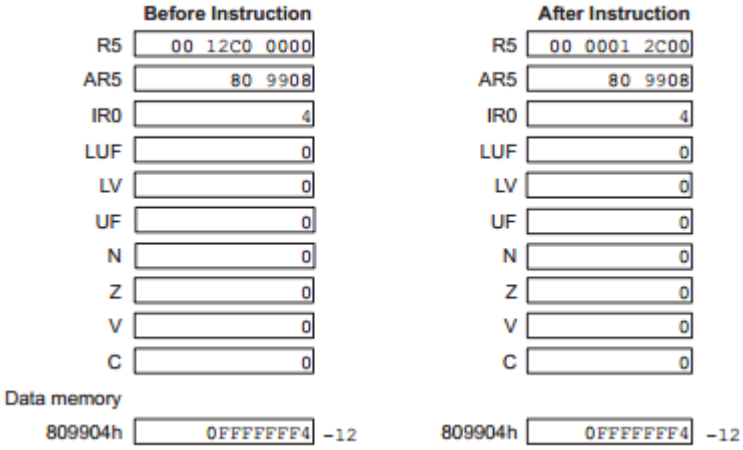
C Устанавливается по значению последнего сдвинутого вовне разряда.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: LSH R4, R7



Пример 2: LSH *-AR5 (IR1), R5



LSH3 Логический сдвиг (трехоперандная)

Синтаксис: LSH3 count, src, dst

Операция: Если count ≥0:
src << count → dst.
Иначе:
src >> |count| → dst.

Операнды: Трехоперандные режимы адресации src (T):
00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)
11 - косвенная (смещение = 0, 1, IR0, IR1)
Трехоперандные режимы адресации count (T):
00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)
11 - косвенная (смещение = 0, 1, IR0, IR1)
dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 0 0 1 0 0 0	T	dst	src	count

Описание: Семь младших разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига. Если операнд count больше 0, копия операнда src сдвигается влево на число разрядов, определенное значением операнда count и результат записывается в dst (src не изменяется). Младшие разряды заполняются нулями, старшие разряды сдвига-

ются в разряд переноса C регистра состояния. Логический сдвиг влево: C← dst← 0. Если операнд count меньше 0, src сдвигается вправо на число разрядов, определенное абсолютным значением операнда count. Старшие разряды операнда dst заполняются 0 при сдвиге вправо. Младшие разряды сдвигаются в разряд переноса C регистра состояния. Логический сдвиг вправо: 0 → dst → C. Если операнд count = 0, сдвиг не происходит и разряд переноса C устанавливается в 0. Предполагается, что операнд count является целым со знаком, а операнды dst и src – беззнаковые целые.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Младший разряд выходного значения.

Z 1 при генерации нулевого выхода, иначе 0.

V 0

C Устанавливается по значению последнего сдвинутого в разряд переноса C регистра состояния. 0 для счетчика сдвига, равного 0. Не изменяется, если dst не является одним из R7-R0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: LSH3 R4, R7, R2

Before Instruction			After Instruction		
R2	00 0000 0000		R2	00 AC00 0000	
R4	00 0000 0018	24	R4	00 0000 0018	24
R7	00 0000 02AC		R7	00 0000 02AC	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	1	
Z	0		Z	0	
V	0		V	1	
C	0		C	0	

Пример 2: LSH3 *-AR4 (IR1), R5, R3

Before Instruction			After Instruction		
R3	00 0000 0000		R3	00 0001 2C00	
R5	00 12C0 0000		R5	00 12C0 0000	
AR4	80 9908		AR4	80 9908	
IR1	4		IR1	4	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
809904h	0FFFFFFF4	-12	809904h	0FFFFFFF4	-12

LSH3||STI Логический сдвиг и сохранить целое

Синтаксис:
LSH3 count, src2, dst1
|| STI src3, dst2

Операция:
Если count ≥0:
src2 << count → dst1.
Иначе:
src2 >> |count| → dst1
|| src3 → dst2

Операнды:
count - регистр (Rn1, 0 ≤ n1 ≤ 7)
src1 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn3, 0 ≤ n3 ≤ 7)
src2 - регистр (Rn4, 0 ≤ n4 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31		24 23		16	15	8 7	0
1 1	0 1 1 1 0	dst1	count	src3	dst2	src2	

Описание:
Семь младших разрядов операнда count используются для генерации двоичного дополнения счетчика сдвига. Если операнд count больше 0, копия операнда src2 сдвигается влево на число разрядов, определенное значением операнда count и результат записывается в dst1 (src2 не изменяется). Младшие разряды заполняются нулями, старшие разряды сдвигаются в разряд переноса C регистра состояния. Логический сдвиг влево: C ← dst2 ← 0. Если операнд count меньше 0, dst сдвигается вправо на число разрядов, определенное абсолютным значением операнда count. Старшие разряды операнда dst заполняются 0 при сдвиге вправо. Младшие разряды сдвигаются в разряд переноса C регистра состояния. Логический сдвиг вправо: 0 → dst2 → C. Если операнд count = 0, сдвиг не происходит и разряд переноса C устанавливается в 0. Предполагается, что операнд count является 7-разрядным целым со знаком, а операнды dst1 и src2 – беззнаковые целые. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая проводится параллельно (LSH3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой LSH3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы:
1

Разряды состояния:
Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV Не изменяется.
UF 0
N Младший разряд выходного значения.
Z 1 при генерации нулевого выхода, иначе 0.
V 0

C Устанавливается по значению последнего сдвинутого во вне разряда. 0 для счетчика сдвига, равного 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1: LSH3 R2, *++AR3 (1), R0

|| STI R4, *-AR5

	Before Instruction		After Instruction	
R0	00 0000 0000		00 AC00 0000	
R2	00 0000 0018	24	00 0000 0018	24
R4	00 0000 00DC	220	00 0000 00DC	220
AR3	80 98C2		80 98C3	
AR5	80 98A3		80 98A3	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
8098C3h	0AC		0AC	
8098A2h	0		0DC	220

Пример 2: LSH3 R7, *AR2-- (1), R2

	Before Instruction		After Instruction	
R0	00 0000 012C	300	00 0000 012C	300
R2	00 0000 0000		00 0002 C000	
R7	00 FFFF FFF4	-12	00 FFFF FFF4	-12
AR0	80 98B7		80 98B7	
AR2	80 9863		80 9862	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
809863h	2C000000		2C000000	
8098B8h	0		12C	300

MPYF Умножение значений с плавающей запятой

Синтаксис: MPYF src, dst

Операция: dst × src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 1 0 0	G	dst	src

Описание: Результат умножения операндов dst и src загружается в регистр dst. Операнд src является числом в формате с ПЗ с одинарной точностью, операнд dst является числом в формате с ПЗ с расширенной точностью.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

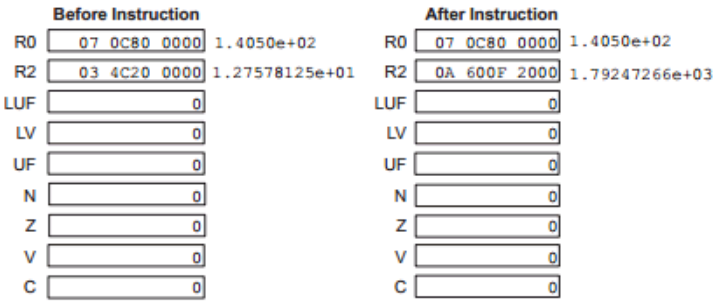
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: MPYF R0, R2



MPYF3 Умножение значений с плавающей запятой (трехоперандная)

Синтаксис: MPYF3 src2, src1, dst

Операция: src1 ×src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23	16	15 8	7 0
0 0 1 0 0 1 0 0 1	T	dst	src1	src2

Описание: Результат умножения операндов dst1 (src1) и src загружается в регистр dst. Операнды src1 и src2 являются числом в формате с ПЗ с одинарной точностью, операнд dst является числом в формате с ПЗ с расширенной точностью.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

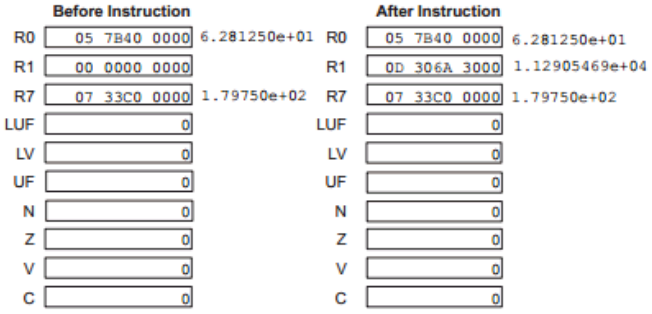
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

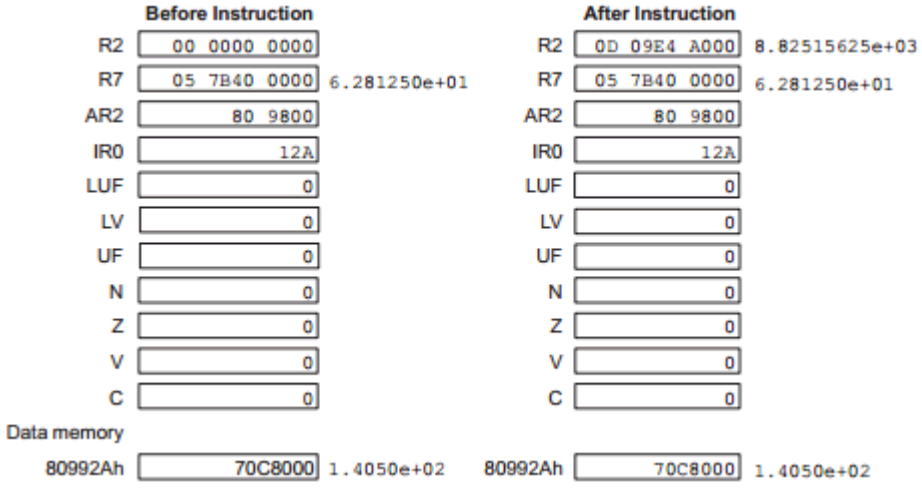
Пример 1: MPYF3 R0, R7, R1



Пример 2: MPYF3 *+AR2 (IR0), R7, R2

или

MPYF3 R7, *+AR2 (IR0), R2



MPYF3||ADDF3 Умножить и прибавить значение в формате с ПЗ

Синтаксис: MPYF3 srcA, srcB, dst1

|| ADDF3 srcC, srcD, dst2

Операция: srcA × srcB → dst1

|| srcC + srcD → dst2

Операнды: srcA |

srcB | Любые два регистры ($0 \leq Rn \leq 7$)

srcC | Любые два косвенная (смещение = 0, 1, IR0, IR1)

srcD |

dst1 - регистр (d1):

0 = R0

1 = R1

dst2 - регистр (d2):

0 = R2

1 = R3

src1 - регистр (Rn, $0 \le n \le 7$)

src2 - регистр (Rn, $0 \le n \le 7$)

src3 - косвенная (смещение = 0, 1, IR0, IR1)

src4 - косвенная (смещение = 0, 1, IR0, IR1)

P - параллельные режимы адресации ($0 \le P \le 3$)

Операция (Поле P)

00 - src3 × src4, src1 + src2

01 - src3 × src1, src4 + src2

10 - src1× src2, src3 + src4

11 - src3 × src1, src2 + src4

Опкод:

31 24	23 16	15 8	7 0
1 0 | 0 0 0 0 | P	d1 | d2 | src1 | src2	src3	src4

Описание: Умножение в формате с ПЗ и сложение в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYF3) считывает из регистра, и операция, которая производится параллельно (ADDF3), записывает в тот же регистр, MPYF3 имеет на входе содержимое регистра до того, как оно было изменено командой ADDF3. Для четырех возможных операндов источника могут быть записаны любые комбинации режимов адресации, при этом два записываются как косвенные, а два – как регистры. Присвоение операндов источника srcA - srcD полям src1 - src4 изменяется в зависимости от комбинации использованных режимов адресации, и поле P кодируется соответственно. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 0

Z 0

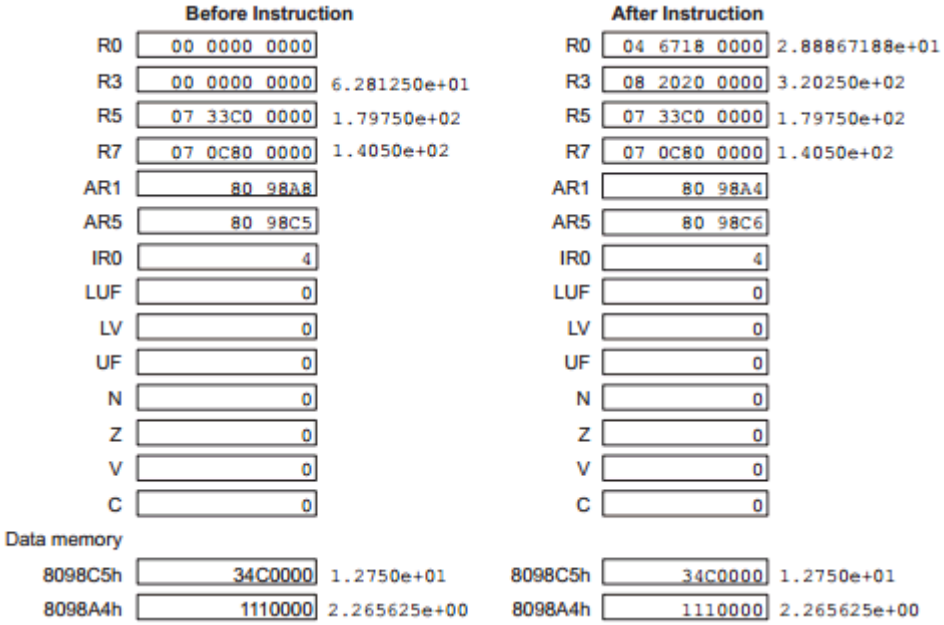
V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: MPYF3 *AR5++ (1),:--AR1 (IR0), R0

|| ADDF3 R5, R7, R3



MPYF3||STF Умножить значения в формате с ПЗ и сохранить значение с ПЗ

Синтаксис: MPYF3 src2, src1, dst

|| STF src3, dst2

Операция: src1× src2 → dst1

|| src3 → dst2

Операнды: src1 - регистр (Rn1, $0 \le n1 \le 7$)

src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn3, $0 \le n3 \le 7$)

src3 - регистр (Rn4, $0 \le n4 \le 7$)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 0 1 1 1 1 | dst1	src1 | src3	dst2	src2

Описание: Умножение в формате с ПЗ и сохранение числа в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYF3) записывает в регистр, и операция, которая производится параллельно (STI), считывает из того же регистра, STF имеет на входе содержимое регистра до того, как оно было изменено командой MPYF3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

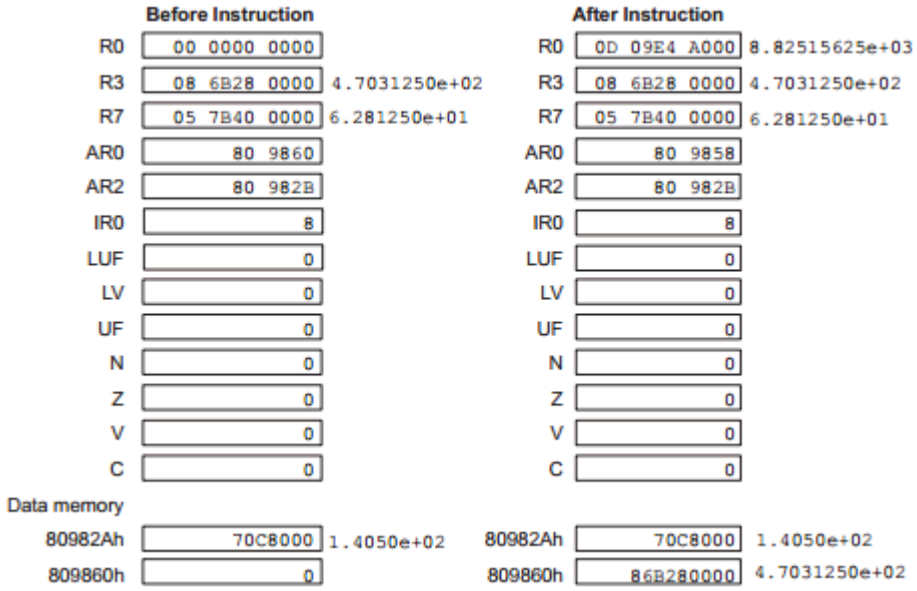
V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: MPYF3 *-AR2 (1), R7, R0

|| STF R3, *AR0-- (IR0)



MPYF3||SUBF3 Умножить и вычесть значение в формате с ПЗ

Синтаксис: MPYF3 srcA, srcB, dst1

|| SUBF3 srcC, srcD, dst2

Операция: srcA ×srcB → dst1

|| srcD - srcC → dst2

Операнды: srcA |

srcB | Любые два регистры (0 ≤ Rn ≤ 7)

srcC | Любые два косвенная (смещение = 0, 1, IR0, IR1)

srcD |

dst1 - регистр (d1):

0 = R0

1 = R1

dst2 - регистр (d2):

0 = R2

1 = R3

src1 - регистр (Rn, $0 \le n \le 7$)

src2 - регистр (Rn, $0 \le n \le 7$)

src3 - косвенная (смещение = 0, 1, IR0, IR1)

src4 - косвенная (смещение = 0, 1, IR0, IR1)

P - параллельные режимы адресации ($0 \le P \le 3$)

Операция (Поле P)

00 - src3 × src4, src1 - src2

01 - src3 × src1, src4 - src2

10 - src1× src2, src3 - src4

11 - src3 × src1, src2 - src4

Опкод:

31 24	23				16 15	8 7	0
1 0	0 0 0 1	P	d1	d2	src1	src2	src3	src4

Описание: Умножение в формате с ПЗ и вычитание в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYF3) считывает из регистра, и операция, которая производится параллельно (SUBF3), записывает в тот же регистр, MPYF3 имеет на входе содержимое регистра до того, как оно было изменено командой SUBF3. Для четырех возможных операндов источника могут быть записаны любые комбинации режимов адресации, при этом два записываются как косвенные, а два – как регистры. Присвоение операндов источника srcA - srcD полям src1 - src4 изменяется в зависимости от комбинации использованных режимов адресации, и поле P кодируется соответственно.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 0

Z 0

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

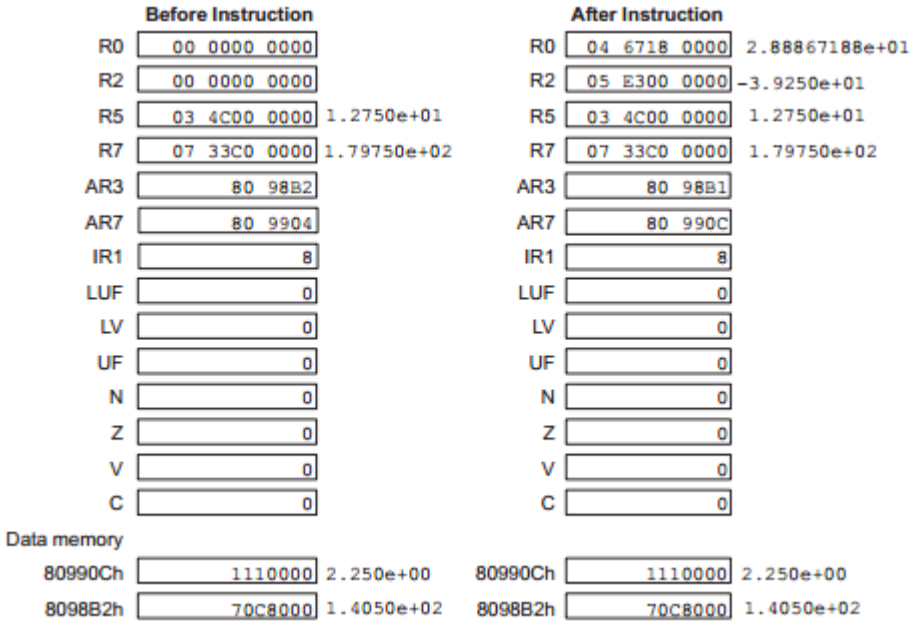
Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
MPYF3 R5, *++AR7 (IR1), R0
|| SUBF3 R7, *AR3-- (1), R2
```

или

```
MPYF3 *++AR7 (IR1), R5, R0
|| SUBF3 R7, *AR3-- (1), R2
```



MPYI Умножение целых

Синтаксис: MPYI src, dst

Операция: dst × src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 1 0 1 0 1	G	dst	src

Описание: Результат произведения операндов dst и src загружается в регистр dst. При чтении операнды src и dst являются 24-разрядными целыми со знаком. Результат является 48-разрядным целым со знаком. В регистр dst помещается 32 младших значащих разряда результата. Целочисленное переполнение возникает, когда хотя бы один из старших 16 разрядов 48-разрядного результата отличается от старшего разряда 32-разрядного выходного значения.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

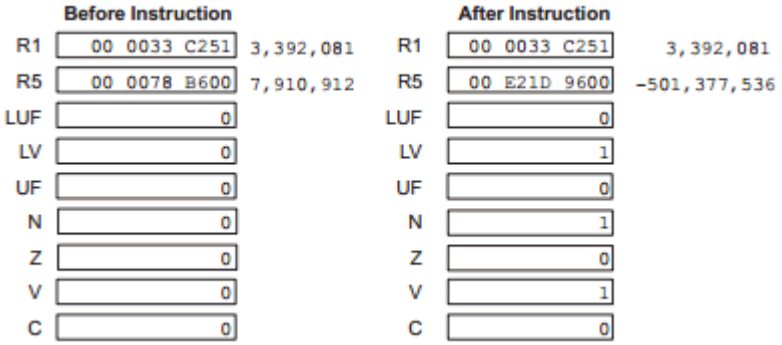
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: MPYI R1, R5



MPYI3 Умножение целых (трехоперандная)

Синтаксис: MPYI3 src2, src1, dst

Операция: src1 × src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, n2 ≤27)

01 - регистровая (Rn2, n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 0 0 1 0 1 0	T	dst	src1	Src2

Описание: Результат произведения операндов src1 и src2 загружается в регистр dst. Операнды src1 и src2 являются 24-разрядными целыми со знаком. Результат является 48-разрядным целым со знаком. В регистр dst помещается 32 младших значащих разряда результата. Целочисленное переполнение возникает, когда хотя бы один из старших 16 разрядов 48-разрядного результата отличается от старшего разряда 32-разрядного выходного значения.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

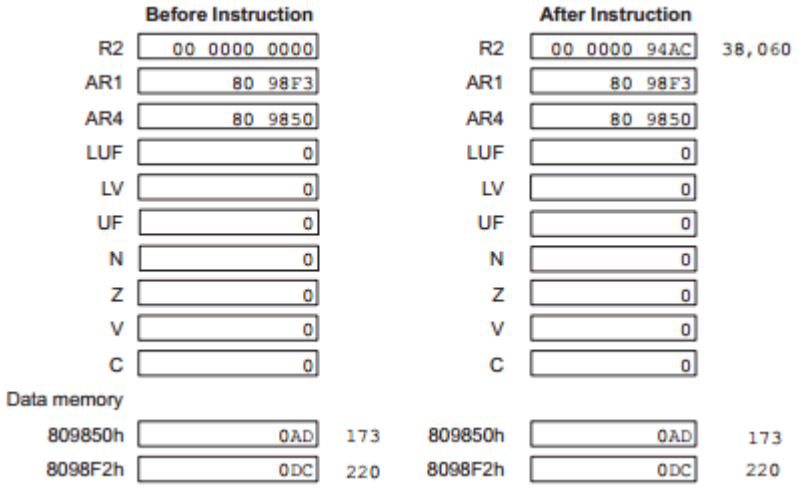
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

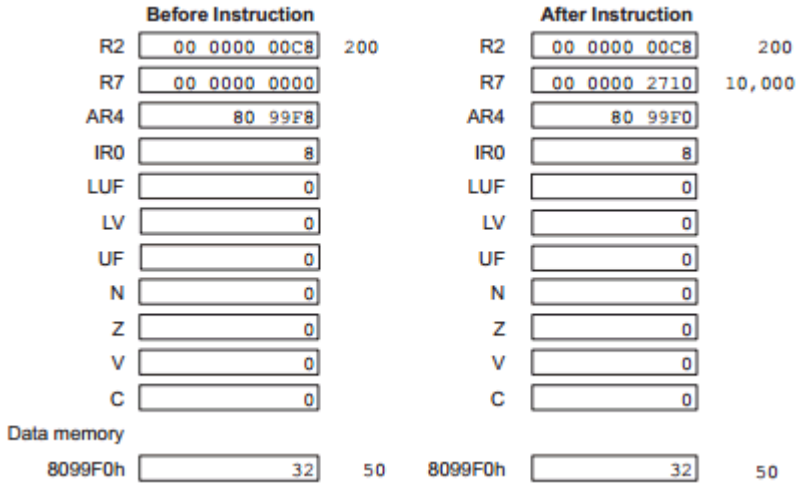
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1: MPYI3 *AR4, *-AR1 (1), R2



Пример 2: MPYI3 *--AR4 (IR0), R2, R7



MPYI3||ADDI3 Умножить и прибавить целое

Синтаксис: MPYI3 srcA, srcB, dst1

|| ADDI3 srcC, srcD, dst2

Операция: srcA × srcB → dst1

|| srcD + srcC → dst2

Операнды: srcA |

srcB | Любые два регистры (0 ≤ Rn ≤ 7)

srcC | Любые два косвенная (смещение = 0, 1, IR0, IR1)

srcD |

dst1 - регистр (d1):

0 = R0

1 = R1

dst2 - регистр (d2):

0 = R2

1 = R3

src1 - регистр (Rn, $0 \leq n \leq 7$)

src2 - регистр (Rn, $0 \leq n \leq 7$)

src3 - косвенная (смещение = 0, 1, IR0, IR1)

src4 - косвенная (смещение = 0, 1, IR0, IR1)

P - параллельные режимы адресации ($0 \leq P \leq 3$)

Операция (Поле P)

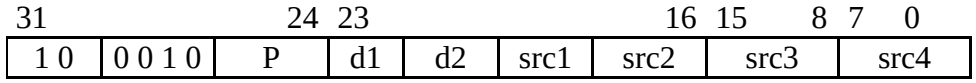
00 - src3 × src4, src1 + src2

01 - src3 × src1, src4 + src2

10 - src1× src2, src3 + src4

11 - src3 × src1, src2 + src4

Опкод:



Описание: Целочисленное умножение и целочисленное сложение производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYI3) считывает из регистра, и операция, которая производится параллельно (ADDI3), записывает в тот же регистр, MPYI3 имеет на входе содержимое регистра до того, как оно было изменено командой ADDI3. Для четырех возможных операндов источника могут быть записаны любые комбинации режимов адресации, при этом два записываются как косвенные, а два – как регистры. Присвоение операндов источника srcA - srcD полям src1 - src4 изменяется в зависимости от комбинации использованных режимов адресации, и поле P кодируется соответственно.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 0

Z 0

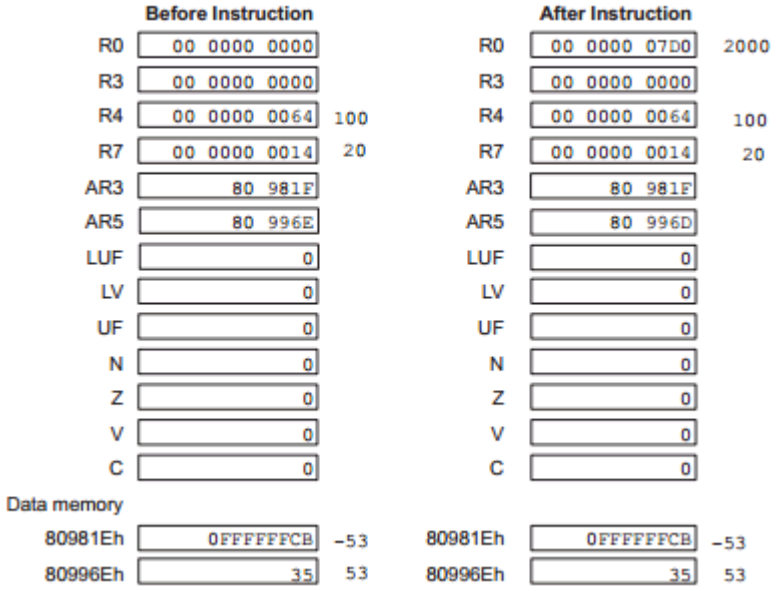
V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
MPYI3 R7, R4, R0
|| ADDI3 *-AR3, *AR5-- (1), R3
```



MPYI3||STI Умножить целое и сохранить целое

Синтаксис: MPYI3 src2, src1, dst
|| STI src3, dst2

Операция: src1× src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, $0 \le n1 \le 7$)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn3, $0 \le n3 \le 7$)
src3 - регистр (Rn4, $0 \le n4 \le 7$)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 0 0 0 0 | dst1	src1	src3	dst2 | src2

Описание: Целочисленное умножение и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (MPYI3) записывает в регистр, и операция, которая производится параллельно (STI), считывает из того же регистра, STI имеет на входе содержимое регистра до того, как оно было изменено командой MPYI3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

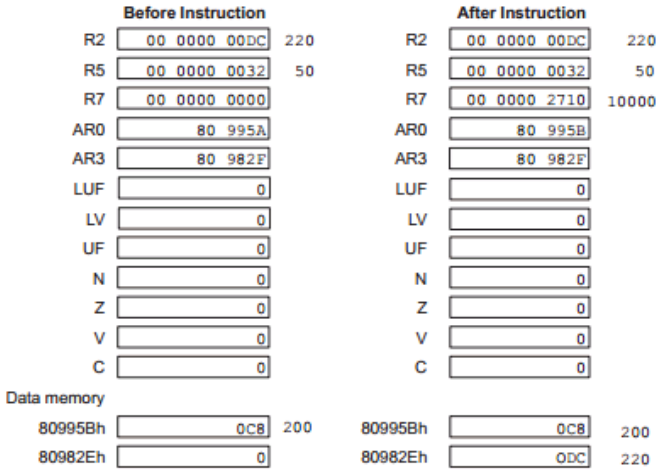
V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: MPYI3 *-AR0 (1), R5, R7

|| STI R2, *-AP3 (1)



MPYI3||SUBI3 Умножить и вычесть целое

Синтаксис: MPYI3 srcA, srcB, dst1

|| SUBI3 srcC, srcD, dst2

Операция: srcA × srcB → dst1

|| srcD - srcC → dst2

Операнды: srcA |

srcB | Любые два регистры (0 ≤ Rn ≤ 7)

srcC | Любые два косвенная (смещение = 0, 1, IR0, IR1)

srcD |

dst1 - регистр (d1):

0 = R0

1 = R1

dst2 - регистр (d2):

0 = R2

1 = R3

src1 - регистр (Rn, 0 ≤ n ≤ 7)

src2 - регистр (Rn, 0 ≤ n ≤ 7)

src3 - косвенная (смещение = 0, 1, IR0, IR1)

src4 - косвенная (смещение = 0, 1, IR0, IR1)

P - параллельные режимы адресации (0 ≤ P ≤ 3)

Операция (Поле P)

00 - src3 × src4, src1 - src2

01 - src3 × src1, src4 - src2

10 - src1 × src2, src3 - src4

11 - src3 × src1, src2 - src4

Опкод:

31 24	23				16 15 8	7 0
1 0	0 0 1 1	P	d1	d2	src1	src2	src3	src4

Описание: Целочисленное умножение и целочисленное вычитание производится параллельно. Все регистры считываются в начале и загружаются в конце

цикла исполнения. Это означает, что если одна из параллельных операций (MPYI3) считывает из регистра, и операция, которая производится параллельно (SUBI3), записывает в тот же регистр, MPYI3 имеет на входе содержимое регистра до того, как оно было изменено командой SUBI3. Для четырех возможных операндов источника могут быть записаны любые комбинации режимов адресации, при этом два записываются как косвенные, а два – как регистры. Присвоение операндов источника srcA - srcD полям src1 - src4 изменяется в зависимости от комбинации использованных режимов адресации, и поле P кодируется соответственно. Когда это не очень важно, ассемблер может изменить порядок в коммутативных операциях с целью упрощения обработки.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного целочисленного переполнения, в противном случае не меняется.

N 0

Z 0

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
MPYI3 R2, *++AR0 (1), R0
|| SUBI3 *AR5-- (IR1), R4, R2
```

или

```
MPYI3 *++AR0 (1), R2, R0
|| SUBI3 *AR5-- (IR1), R4, R2
```

	Before Instruction		After Instruction	
R0	00 0000 0000		00 0000 1324	4900
R2	00 0000 0032	50	00 0000 0320	800
R4	00 0000 07D0	2000	00 0000 07D0	2000
AR0	80 98E3		80 98E4	
AR5	80 99FC		80 99F0	
IR1	0C		0C	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
8098E4h	62	98	62	98
8099FCh	4B0	1200	4B0	1200

NEGB Отрицание целого с заемом

Синтаксис: NEGB src, dst

Операция: 0 - src - C → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 1 1 0 G	dst	src	

Описание: Разница от 0 операнда src и C загружаются в регистр dst. Предполагается, src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

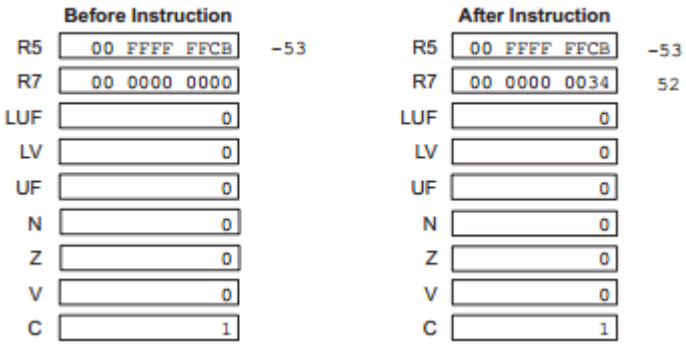
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при генерации заема, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: NEGB R5,R7



NEGF Отрицание значения с плавающей запятой

Синтаксис: NEGF src, dst

Операция: 0 - src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 7)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 1 1 1	G	dst	src

Описание: Операнд, представляющий собой отличие от нуля операнда src загружается в регистр dst. Предполагается, src и dst являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

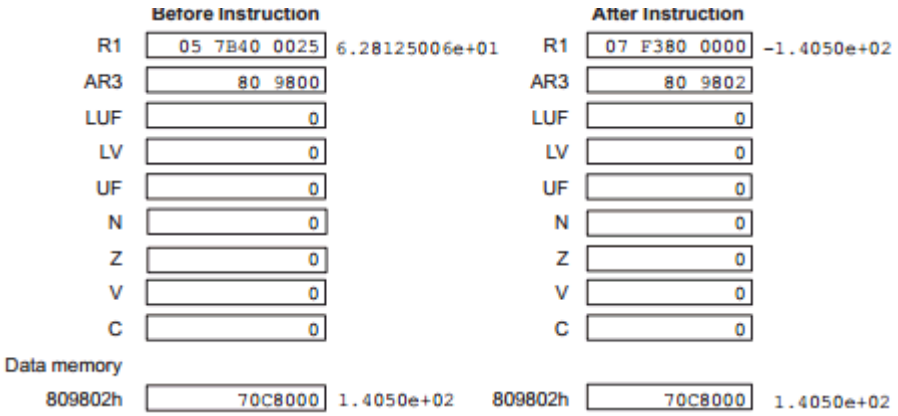
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: NEGF *++AR3 (2), R1



NEGF||STF Обратное значение в формате с ПЗ и сохранить значение с ПЗ

Синтаксис: NEGF src2, dst1

|| STF src3, dst2

Операция: 0 - src2 → dst1

|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn1, $0 \le n1 \le 7$)

src3 - регистр (Rn2, $0 \le n2 \le 7$)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0		
1 1	1 0 0 0 1	dst1	0 0 0	src3	dst2	src2

Описание: Отрицание числа в формате с ПЗ и сохранение числа в формате с ПЗ производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (NEGF), записывает в тот же регистр,

STF имеет на входе содержимое регистра до того, как оно было изменено командой NEGF. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

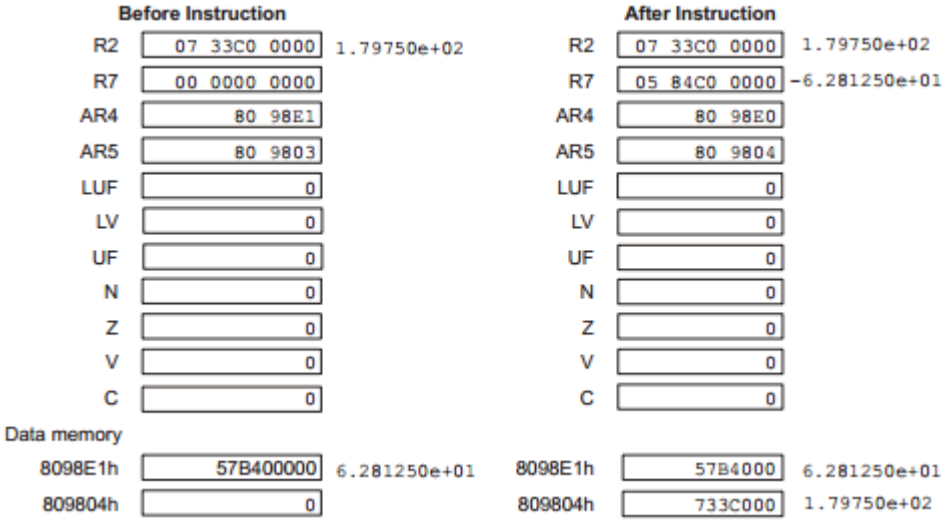
V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
NEGF *AR4-- (1), R7
|| STF R2,*++AR5 (1)
```



NEGI Отрицает целое число

Синтаксис: NEGI src, dst

Операция: 0 - src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 0 0	G	dst	src

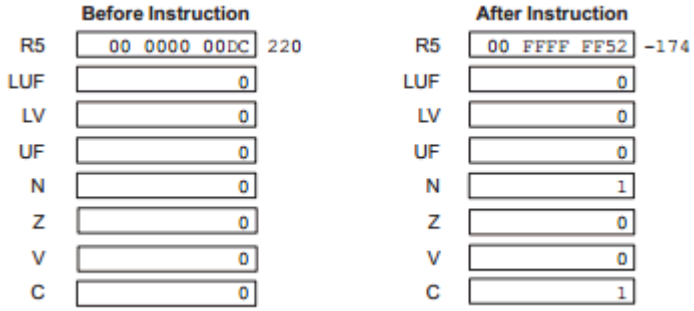
Описание: Операнд, представляющий собой отличие от нуля операнда src, загружается в регистр dst. Предполагается, src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: NEGI 174, R5 (174 = 0AEh)



NEG||STI Обратное целое и сохранить целое

Синтаксис: NEGI src2, dst1
|| STI src3, dst2

Операция: 0 - src2 → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 0 0 1 0 | dst1	0 0 1 | src3	dst2	src2

Описание: Отрицание целого числа в формате и сохранение целого числа производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (NEGI), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой NEGI. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

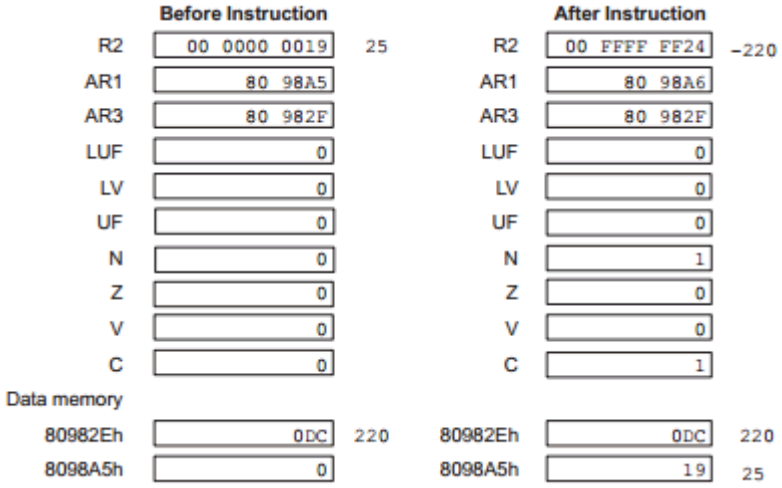
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:
```
NEGI *-AR3, R2
 || STI R2, *AR1++
```



NOP Нет операции

Синтаксис: NOP src

Операция: Нет никаких операций АЛУ или умножителя. ARn изменяются, если src определен в режиме косвенной адресации.

Операнды: основные режимы адресации src (G):

00 - регистровая (нет операции)

10 - косвенная (изменяет ARn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0	0 1 1 0 0 1 G	0 0 0 0 0	src

Описание: Если операнд src определен в косвенном режиме адресации, выполняется определенная операция адресации и имеет место пустое чтение памяти. Если операнд src опущен, не выполняется никакие операции.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: NOP

Пример 2: NOP *AR3-- (1)

NORM Нормирование значений с плавающей запятой

Синтаксис: NORM src, dst

Операция: norm(src) → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 1 0	G	dst	src

Описание: Предполагается, что операнд src является ненормализованным числом в формате с ПЗ, т. е. неявный (следующий за знаковым) разряд установлен по значению знакового разряда. Операнд dst равен нормализованному операнду src с удаленным неявным разрядом. Экспонента операнда dst устанавливается равной экспоненте операнда src минус величина левого сдвига, необходимого для нормализации src. Операнд dst является нормализованным числом в формате с ПЗ. Если src(exp) = -128 и src(man) = 0, тогда dst = 0, Z = 1 и UF = 0. Если src(exp) = -128 и src(man) № 0, тогда dst = 0, Z = 0 и UF = 1. При любых других значениях src при возникновении отрицательного переполнения (т.е. потере значимости), dst(man) устанавливается в 0 и dst(exp) = -128. Если src(man) = 0, тогда dst(man) = 0 и dst(exp) = - 128. См. подраздел 4.6.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV Не изменяется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: NORM R1, R2

Before Instruction		After Instruction		
R1	04 0000 3AF5	R1	04 0000 3AF5	
R2	07 0C80 0000	R2	F2 6BD4 0000	1.12451613e - 04
LUF	0	LUF	0	
LV	0	LV	0	
UF	0	UF	0	
N	0	N	0	
Z	0	Z	0	
V	0	V	0	
C	0	C	0	

NOT Поразрядное логическое дополнение

Синтаксис: NOT src, dst

Операция: ~src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 1 1	G	dst	src

Описание: Поразрядное логическое дополнение операнда src загружается в регистр dst. Дополнение формируется путем инверсии каждого разряда операнда src. Операнды src и dst являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения.

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: NOT @982Ch, R4

Before Instruction		After Instruction	
R4	00 0000 0000	R4	00 FFFF A1D0
DP	080	DP	080
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	1
Z	0	Z	0
V	0	V	0
C	0	C	0
Data memory			
80982Ch	5E2F	80982Ch	5E2F

NOT||STI Логическое дополнение (поразрядная инверсия) значения и сохранить целое

Синтаксис: NOT src2, dst1
|| STI src3, dst2

Операция: ~src2 → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 0 0 1 1 | dst1	0 0 0 | src3	dst2	src2

Описание: Поразрядное логическое НЕ(NOT) и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (NOT), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой NOT. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.
LV Не изменяется.
UF 0
N Старший разряд выходного значения.
Z 1 при генерации нулевого результата, иначе 0.
V 0
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:
```
NOT *AR2, R3
|| STI R7, *--AR4 (IR1)
```

	Before Instruction		After Instruction	
R3	00 0000 0000		00 FFFF F3D0	
R7	00 0000 00DC	220	00 0000 00DC	220
AR2	80 99CB		80 99CB	
AR4	80 9850		80 9840	
IR1	10		10	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		1	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
8099CCh	0C2F		0C2F	
809840h	0		0DC	220

OR Поразрядное логическое OR (ИЛИ)

Синтаксис: OR src, dst

Операция: dst OR src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 0 0 0 0 0 0	G	dst	src

Описание: Поразрядное логическое ИЛИ (OR) между операндами src и dst загружается в регистр dst. Операнды src и dst являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: OR *++AR1 (IR1), R2

Before Instruction		After Instruction	
R2	00 1256 0000	R2	00 1256 2BCD
AR1	80 9800	AR1	80 9804
IR1	4	IR1	4
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0
Data memory			
809804h	2BCD	809804h	2BCD

OR3 Поразрядное логическое – OR (ИЛИ) (трехоперандная)

Синтаксис: OR3 src2, src1, dst

Операция: src1 OR src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, $0 \le n1 \le 27$)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, $0 \le n1 \le 27$)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 0 0 1 0 1 1	T	dst	src1	src2

Описание: Поразрядное логическое ИЛИ(OR) между операндами src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: OR3 *++AR1(IR1), R2, R7

	Before Instruction	After Instruction
R2	00 1256 0000	00 1256 0000
R7	00 0000 0000	0 1256 2BCD
AR1	80 9800	80 9804
IR1	4	4
LUF	0	0
LV	0	0
UF	0	0
N	0	0
Z	0	0
V	0	0
C	0	0
Data memory		
809804h	2BCD	2BCD

OR3||STI Поразрядное логическое ИЛИ и сохранить целое

Синтаксис: OR3 src2, src1, dst1

|| STI src3, dst2

Операция: src1 OR src2 → dst1

|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)

src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)

src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31		24	23		16	15		8	7		0
0 0	1 0 1 0 0		dst1	src1	src3	dst2				src2	

Описание: Поразрядное логическое ИЛИ (OR) и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (OR3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой OR3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
OR3 *++AR2, R5, R2
|| STI R6, *AR1--
```

	Before Instruction		After Instruction	
R2	00 0000 0000		00 0080 9800	
R5	00 0080 0000		00 0080 0000	
R6	00 0000 00DC	220	00 0000 00DC	220
AR1	80 9883		80 9882	
AR2	80 9830		80 9831	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
809831h	9800		809831h	9800
809883h	0		809883h	0DC 220

POP Выталкивание значений с плавающей запятой из стека

Синтаксис: POP dst

Операция: *SP-- → dst

Операнды: dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 … 24	23 … 16	15 … 8	7 … 0
0 0 0 0 1 1 1 0 0	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Описание: Вершина системного стека выталкивается и загружается в регистр dst (в 32 младших разряда). Вершина стека – целое со знаком. Команда POP выполняется с постдекрементом указателя стека. Разряды экспоненты в регистрах расширенной точности (R7-R0) слева не изменяются.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

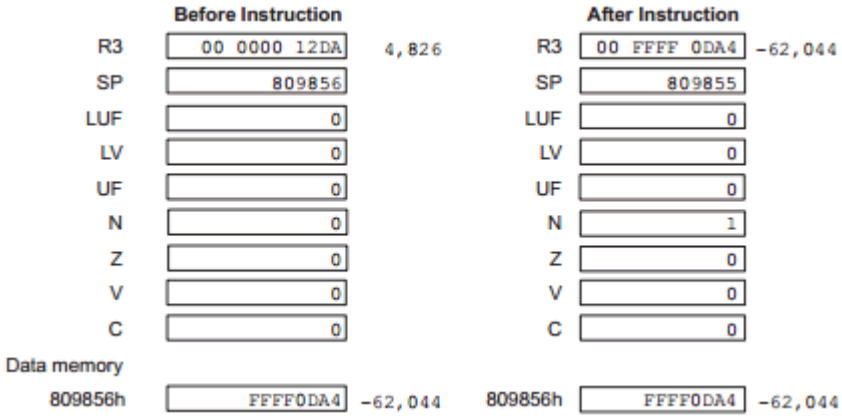
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: POP R3



POPF Выталкивание значений с плавающей запятой из стека

Синтаксис: POPF dst

Операция: *SP-- → dst1

Операнды: dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 … 24	23 … 16	15 … 8	7 … 0
0 0 0 0 1 1 1 0 0	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Описание: Вершина системного стека выталкивается и загружается в регистр dst (в 32 старших разряда). Вершина стека – число в формате с ПЗ. Команда POPF выполняется с постдекрементом указателя стека. 8 младших разрядов в регистрах расширенной точности (R7-R0) заполняются нулями.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 0

UF Не изменяется.

N 1 при генерации отрицательного результата, иначе 0.

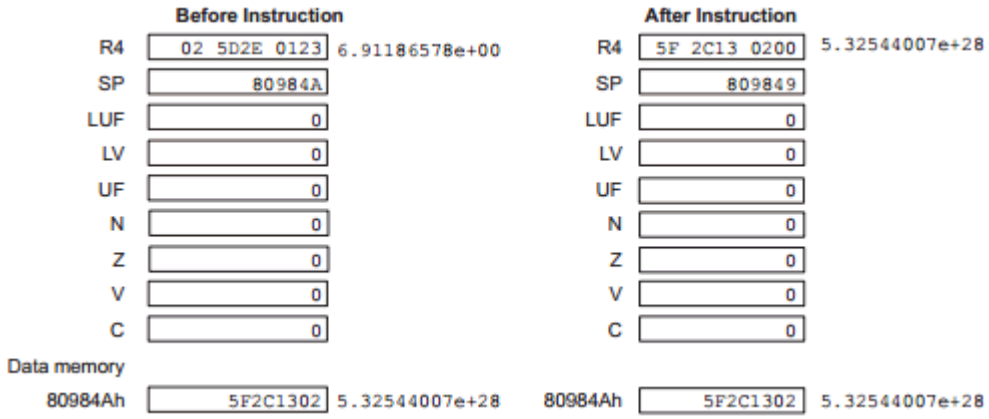
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: POPF R4



PUSH Загрузка в стек целого

Синтаксис: PUSH src

Операция: src → *++SP

Операнды: src - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 1 1 1 0	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

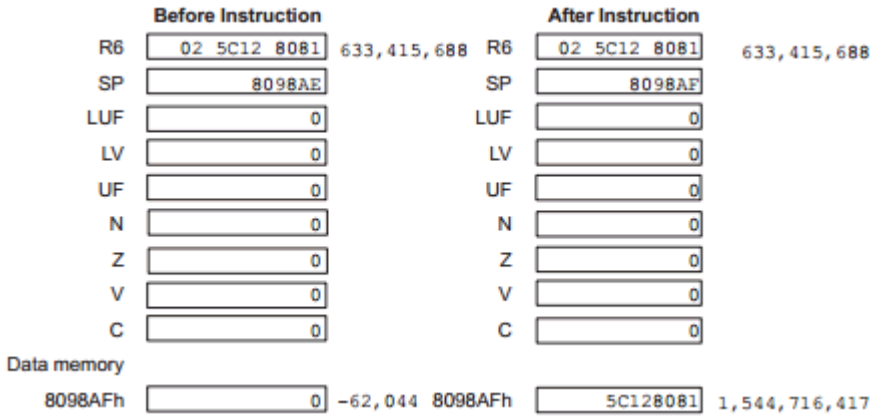
Описание: Содержимое регистра src (32 младших значащих разряда) загружается в текущий системный стек. Предполагается, что src является целым со знаком. Команда PUSH выполняется с прединкрементом указателя стека. Целая часть или мантисса регистров с расширенной точностью (R7-R0) сохраняется при этой команды.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: PUSH R6



PUSHF Загрузка значений с плавающей запятой в стек

Синтаксис: PUSHF src

Операция: src → *++SP

Операнды: src - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23	16	15 8	7 0
0 0 0	0 1 1 1 1 0	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

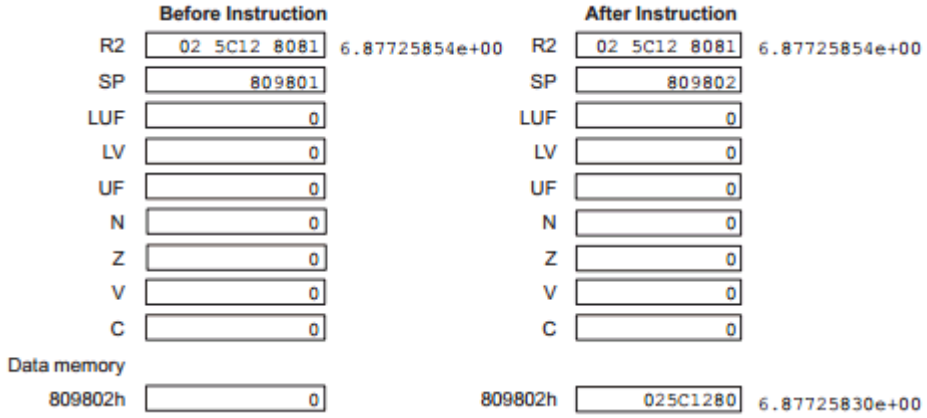
Описание: Содержимое регистра src (32 старших значащих разряда) загружается в текущий системный стек. Предполагается, что src является числом в формате с ПЗ. Команда PUSHF выполняется с прединкрементом указателя стека. 8 младших значащих разрядов мантиссы не сохраняются (обратите внимание на отличие значений в R2 и в стеке в приведенном ниже примере).

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: PUSHF R2



RETIcond Условный выход из прерывания

Синтаксис: RETIcond

Операция: Если cond - "истина":
*SP-- → PC
1 → ST (GIE).
Иначе, продолжение.

Операнды: Нет

Опкод:

31 24	23	16	15 8	7 0
0 1 1 1 1	0 0 0 0	0 0	cond	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Описание: Выполняется условный возврат. Если условие истинно, вершина стека выталкивается в PC, и 1 записывается в разряд GIE (глобальное разрешение прерывания) регистра состояния. Это имеет эффектом разреше-

ние всех прерываний, для которых соответствующий разряд разрешения прерывания установлен в 1. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2, где приведен список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: RETINZ

	Before Instruction		After Instruction
PC	0456	PC	0123
SP	809830	SP	80982F
ST	0	ST	2000
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0
Data memory			
809830h	123	809830h	123

RETScond Условный выход из подпрограммы

Синтаксис: RETScond

Операция: Если cond - "истина":
*SP-- → PC
Иначе, продолжение.

Операнды: Нет

Опкод:

31 24	23 16	15 8	7 0
0 1 1 1 1 0 0 0 1	0 0 cond	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Описание: Выполняется условный возврат. Если условие истинно, вершина стека выталкивается в PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполня-

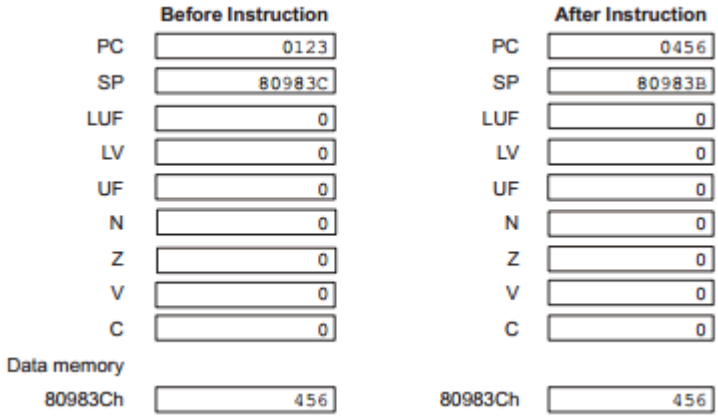
ется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: RETSGE



RND Округление значения с плавающей запятой

Синтаксис: RND src, dst

Операция: rnd (src) → dst

Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, $0 \le n \le 7$)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 0 0 1 0 | G	dst	src	

Описание: Результат округления операнда src загружается в регистр dst. Операнд src округляется до ближайшего значения с одинарной точностью в формате с ПЗ. Если значение операнда src находится точно посредине между двумя значениями с одинарной точностью в формате с ПЗ, то оно округляется к большему положительному из этих двух значений.

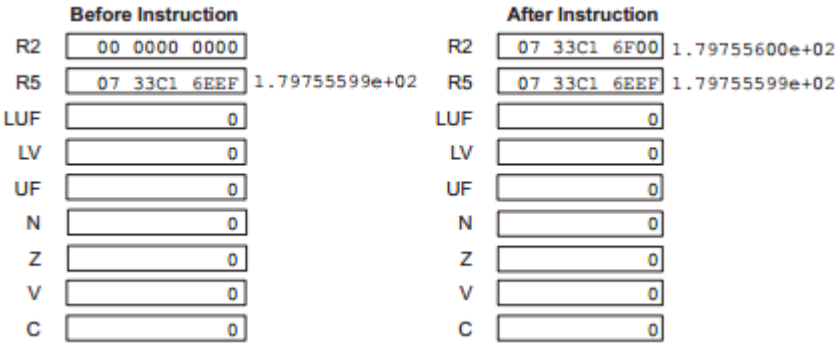
Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: RND R5, R2



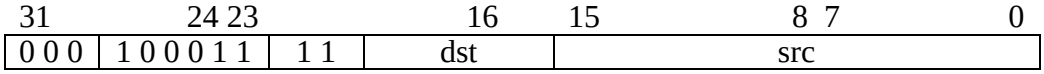
ROL Циклический сдвиг влево

Синтаксис: ROL dst

Операция: dst циклически сдвинутый влево на 1 разряд → dst

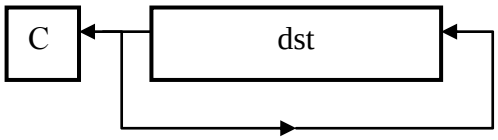
Операнды: dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:



Описание: Содержимое операнда dst циклически сдвигается влево на один разряд и загружается в регистр dst. Это циклический сдвиг с пересылкой старшего значащего разряда в младший.

Циклический левый сдвиг:



Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.
LV Не изменяется.
UF 0
N Старший значащий разряд выходного значения
Z 1 при генерации нулевого результата, иначе 0.
V 0
C Устанавливается по значению циклически сдвигаемого вовне старшего разряда. Не изменяется, если dst не является одним из R7-R0

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: ROL R3

Before Instruction	
R3	00 8002 5CD4
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

After Instruction	
R3	00 0004 B9A9
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

ROLC Циклический сдвиг влево через перенос

Синтаксис: ROLC dst

Операция: dst циклически сдвинутый влево на 1 разряд через разряд переноса →dst

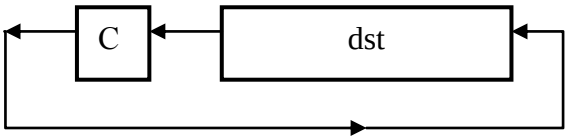
Операнды: dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 0 1 0 0 | 1 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Описание: Содержимое операнда dst циклически сдвигается влево на один разряд через разряд переноса и загружается в регистр dst. Это циклический сдвиг с пересылкой старшего значащего разряда в разряд переноса, в то время как разряд переноса пересылается в младший значащий разряд.

Циклический левый сдвиг:



Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения.

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Устанавливается по значению циклически сдвигаемого вовне старшего разряда.

Если dst не является одним из R7-R0, C сдвигается в dst, но не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: ROLC R3

Before Instruction	
R3	00 0000 0420
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

After Instruction	
R3	00 0000 0841
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

Пример 2: ROLC R3

Before Instruction	
R3	00 8000 4281
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

After Instruction	
R3	00 0000 8502
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

ROR Циклический сдвиг вправо

Синтаксис: ROR dst

Операция: dst циклически сдвинутый вправо на 1 разряд через разряд переноса → dst

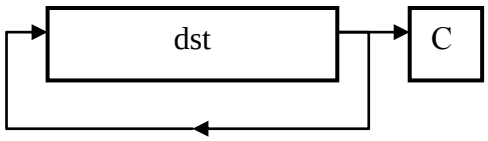
Операнды: dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 0 1 0 1 | 1 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Описание: Содержимое операнда dst циклически сдвигается вправо на один разряд и загружается в регистр dst. Младший значащий разряд циклически сдвигается в разряд переноса, а также пересылается в старший значащий разряд.

Циклический правый сдвиг:



Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Устанавливается по значению циклически сдвигаемого вовне старшего разряда. Не изменяется, если dst не является одним из R7-R0

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: ROR R7

Before Instruction	
R7	00 0000 0421
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

After Instruction	
R7	00 8000 0210
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	1

RORC Циклический сдвиг вправо

Синтаксис: RORC dst

Операция: dst циклически сдвинутый вправо на 1 разряд через разряд переноса →dst

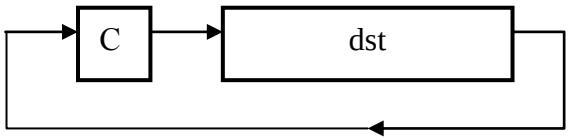
Операнды: dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 0 1 1 0 | 1 1	dst	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	

Описание: Содержимое операнда dst циклически сдвигается вправо на один разряд через разряд переноса регистра состояния и загружается в регистр dst. Это выглядит как 33-разрядный сдвиг. Разряд переноса пересылается в старший значащий разряд dst, в то время как младший значащий разряд dst циклически сдвигается в разряд переноса.

Циклический левый сдвиг:



Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Устанавливается по значению циклически сдвигаемого вовне старшего разряда. Если dst не является одним из R7-R0, C сдвигается в dst, но не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: RORC R4

Before Instruction		After Instruction	
R4	00 8000 0081	R4	00 4000 0040
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	1	N	0
Z	0	Z	0
V	0	V	0
C	0	C	1

RPTB Повторение блока команд

Синтаксис: RPTB src

Операция: src → RE

1 → ST (RM)

Следующий PC → RS

Операнды: src длинный непосредственный режим адресации

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 1 0 0	src(смещение)		

Описание: RPTB позволяет обеспечить повторение блока команд несколько раз без потерь на зацикливание. Эта команда активизирует режим повторения при изменении PC. Операнд src – 24-разрядное непосредственное значение, которое загружается в регистр адреса конца повторений (RE). В разряд режима повторения регистра состояния ST (RM) записывается 1, указывая, что PC будет изменяться в режиме повторения. Адрес следующей команды загружается в регистр адреса начала повторения (RS).

Циклы: 4

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: RPTB 127h

	Before Instruction
PC	0123
RE	0
RS	0
ST	0
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

	After Instruction
PC	0124
RE	127
RS	124
ST	100
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

RPTS Повторение одной инструкции

Синтаксис: RPTS src

Операция: src → RC
1 → ST (RM)
1 → S
Следующий PC → RS
Следующий PC → RE

Операнды: основные режимы адресации src (G):
00 - регистровая
01 - прямая
10 - косвенная
11 - непосредственная

Опкод:

31 24	23 16	15 8	7 0
000 100111	G 11011	src	

Описание: RPTB позволяет обеспечить повторение отдельной команды несколько раз без потерь на зацикливание. Выборка из регистра команд (IR) также может производиться, что позволяет предотвращать доступ к памяти повторений. Src загружается в счетчик повторений (RC). В разряд режима повторения регистра состояния ST (RM). 1 также записывается в разряд одиночного повторения (S). Это указывает на то, что выборка программы будет выполняться только из регистра команд. Следующий PC загружается в регистр адреса конца повторения (RE) и в регистр адреса начала повторения (RS). Для непосредственного режима операнд src – беззнаковое целое без расширения знака.

Циклы: 4

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: RPTS AR5

	Before Instruction		After Instruction
AR5	00 00FF	AR5	00 00FF
PC	0123	PC	0124
RC	0	RC	0FF
RE	0	RE	124
RS	0	RS	124
ST	0	ST	100
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0

SIGI Сигнал блокировки сигнализирует об операции блокировки

Синтаксис: SIGI

Операция: Операция сигнализации блокировки. Ожидание подтверждения блокировки. Очистка блокировки.

Операнды: Нет

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 1 0 0	G	dst	src

Описание: Операция межпроцессорного взаимодействия (блокировки) сигнализируется через XF0 и XF1. После подтверждения операции блокировки, операция блокировки завершается. SIGI игнорирует внешние сигналы

готовности. См. подраздел 6.4 для получения более детальной информации.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

STF Сохранение значения с плавающей запятой

Синтаксис: STF src, dst

Операция: src → dst

Операнды: src - регистр (Rn, $0 \leq n \leq 7$)
основные режимы адресации dst (G):
01 - прямая
10 - косвенная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 1 0 0 0	G	src	dst

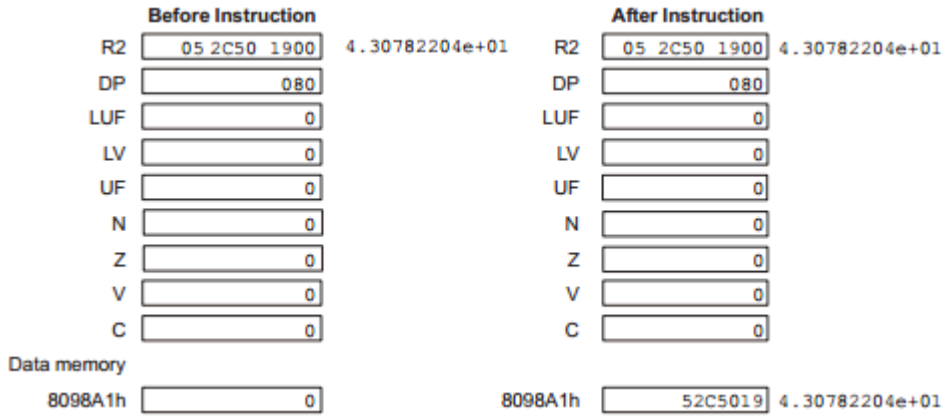
Описание: Операнд src загружается в память по адресу dst. Операнды src и dst являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: STF R2,@98A1h



STFI Сохранение значения с плавающей запятой, блокировка

Синтаксис: STFI src, dst

Операция: src → dst

Сигнализирует об окончании операции блокировки

Операнды: src - регистр (Rn, 0 ≤ n ≤ 7)

основные режимы адресации dst (G):

01 - прямая

10 - косвенная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 1 0 0 1	G	src	dst

Описание: Операнд src загружается в память по адресу dst. Операция блокировки сигнализируется через выводы XF0 и XF1. Операнды src и dst являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

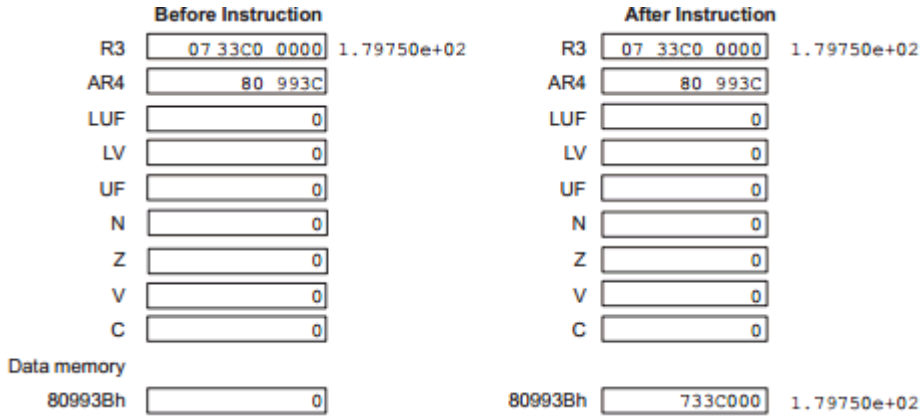
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: STFI R3,*-AR4



STF||STF Сохранить значения в формате с ПЗ

Синтаксис: STF src2, dst2

|| STF src1, dst1

Операция: src2 → dst2

|| src1 → dst1

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)

dst1 косвенная (смещение = 0, 1, IR0, IR1)

src2 - регистр (Rn2, 0 ≤ n2 ≤ 7)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31	24	23	16	15	8	7	0
1 1	0 0 0 0 0	src2	0 0 0	src1	dst1	dst2	

Описание: Две команды STF выполняются параллельно. Оба операнда src1 и src2 являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:
```
STF R4,*AR3--
|| STF R3,*++AR5
```

Before Instruction			After Instruction		
R3	07 33C0 0000	1.79750e+02	R3	07 33C0 0000	1.79750e+02
R4	07 0C80 0000	1.4050e+02	R4	07 0C80 0000	1.4050e+02
AR3	80 9835		AR3	80 9834	
AR5	80 99D2		AR5	80 99D3	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
809835h	0		809835h	070C8000	1.4050e+02
8099D3h	0		8099D3h	0733C000	1.79750e+02

STI Сохранение целого

Синтаксис: STI src, dst

Операция: src → dst

Операнды: src - регистр (Rn, 0 ≤ n ≤ 27)
основные режимы адресации dst (G):
01 - прямая
10 - косвенная

Опкод:

31	24	23	16	15	8	7	0
0 0 0	1 0 1 0 1 0	G	src	dst			

Описание: Операнд src загружается в память по адресу dst. Операнды src и dst являются целыми со знаком.

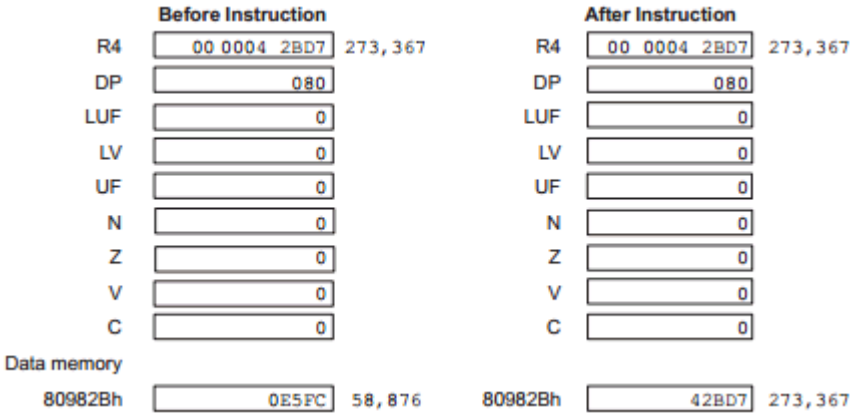
Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.

Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: STI R4,@982Bh



STII Сохранение целых, блокировка

Синтаксис: STII src, dst

Операция: src → dst сигнализирует об окончании операции блокировки

Операнды: src - регистр (Rn, $0 \le n \le 27$)
основные режимы адресации dst (G):
01 - прямая
10 - косвенная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 1 1	G	src	dst

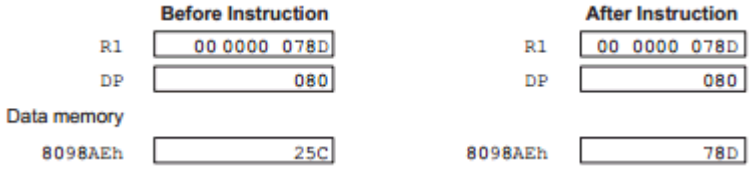
Описание: Операнд src загружается в память по адресу dst. Операция блокировки сигнализируется через выводы XF0 и XF1. Операнды src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: STII R1,@98AEh



STI||STI Сохранить целые

Синтаксис: STI src2, dst2
|| STI src1, dst1

Операция: src2 → dst2
|| src1 → dst1

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
dst1 косвенная (смещение = 0, 1, IR0, IR1)
src2 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 0 0 0 0 1 | src2	0 0 0 | src1	dst1	dst2

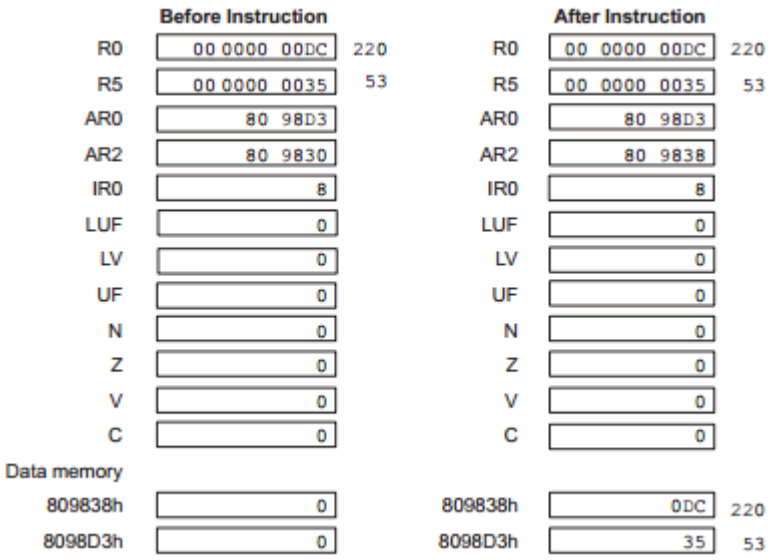
Описание: Сохранение двух целых производится параллельно. Если обе операции сохранения выполняются по одному адресу, значение записывается как при выполнении операции STI src2, src2.

Циклы: 1

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: STI R0, *++AR2 (IR0)
|| STI R5, *AR0



SUBB Вычитание целых с заемом

Синтаксис: SUBB src, dst

Операция: dst-src-C → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 1 1 0 1 | G	dst	src	

Описание: Разность между операторами dst, src и C загружается в регистр dst. Предполагается, что операнды dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

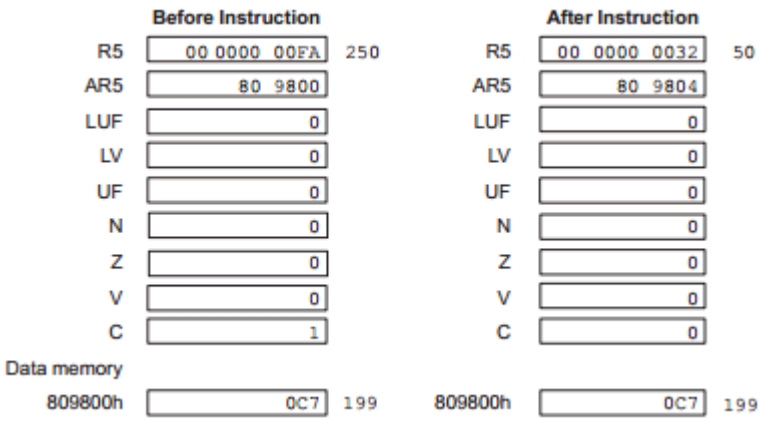
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBB *AR5++ (4), R5



SUBB3 Вычитание целых с заемом (трехоперандная)

Синтаксис: SUBB3 src2, src1, dst

Операция: src1-src2-C → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 0 0 1 1 0 0 | T	dst	src1	src2

Описание: Разность между операндами src1 и src2, и флагом C (перенос) загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

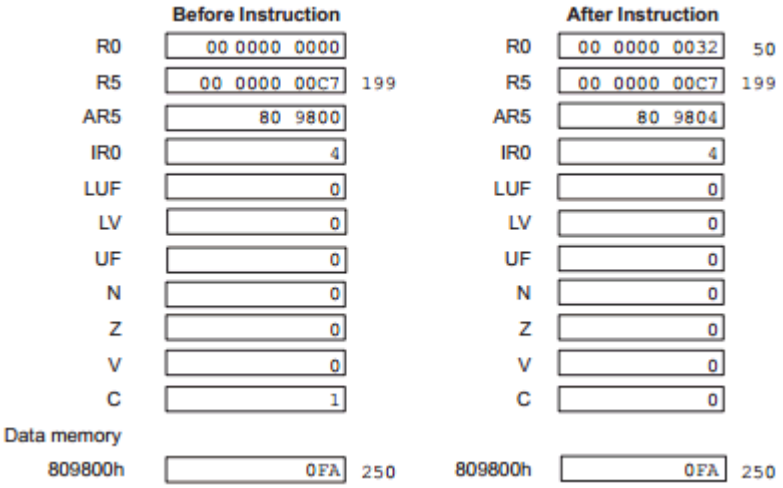
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBB3 R5, *AR5++ (IR0), R0



SUBC Условное вычитание целых

Синтаксис: SUBC src, dst

Операция: Если (dst - src ≥ 0):

(dst - src << 1) или 1 → dst.

Иначе:

dst << 1 → dst.

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 1 0 1 1 1 0	G	dst	src

Описание: Операнд src вычитается из операнда dst. Операнд dst загружается со значением, зависящим от результата вычитания. Если (dst - src) больше или равно 0, (dst - src) сдвигается влево на один разряд, младший значащий разряд устанавливается в 1, и результат загружается в регистр dst. Если (dst - src) меньше или равно нулю, dst сдвигается влево на один разряд и загружается в регистр dst. Операнды dst и src являются целыми без знака. SUBC может быть использована для выполнения за один шаг многоразрядного целочисленного деления. См. 11.3.4 для детального описания.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

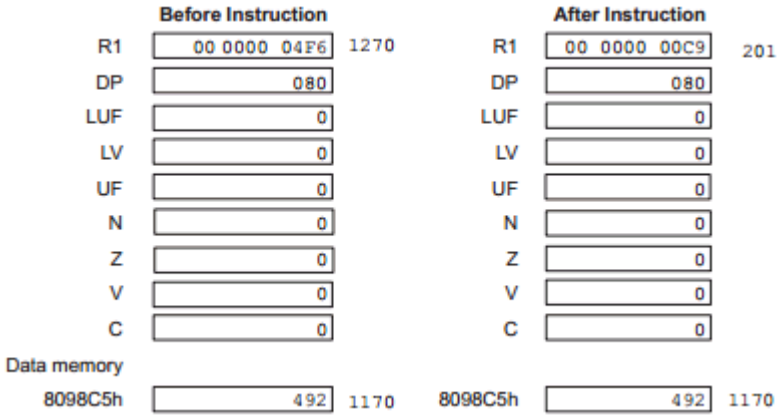
Z Не изменяется.

V Не изменяется.

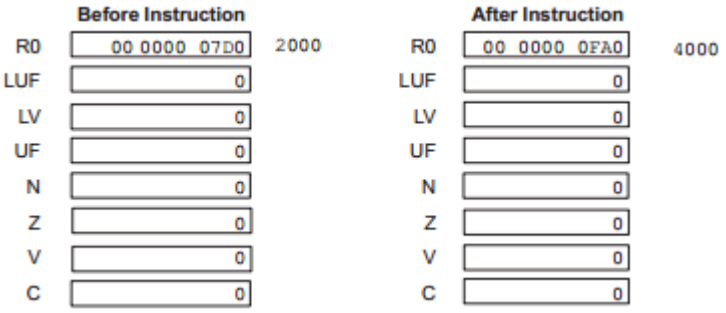
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: SUBC @98C5h, R1



Пример 2: SUBC 3000, R0 (3000 = 0BB8h)



SUBF Условное вычитание целых

Синтаксис: SUBF src, dst

Операция: dst - src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 1 1 1 G	dst	src	

Описание: Разность операнда dst минус операнд src загружается в регистр dst. Операнды dst и src являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

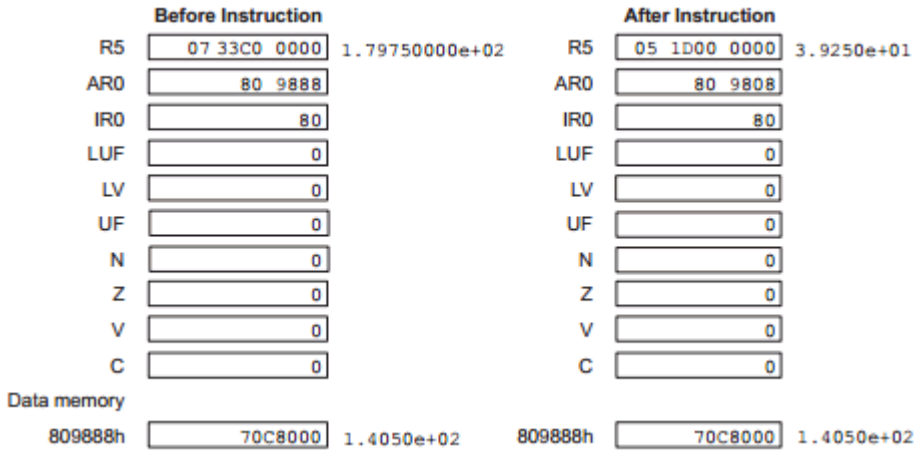
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: SUBF *AR0-- (IR0), R5



SUBF3 Вычитание значений с плавающей запятой (трехоперандная)

Синтаксис: SUBF3 src2, src1, dst

Операция: src1 - src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):
00 - регистровая (Rn1, 0 ≤ n1 ≤ 7)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn1, 0 ≤ n1 ≤ 7)
11 - косвенная (смещение = 0, 1, IR0, IR1)
Трехоперандные режимы адресации src2 (T):
00 - регистровая (Rn2, 0 ≤ n2 ≤ 7)
01 - регистровая (Rn2, 0 ≤ n2 ≤ 7)
10 - косвенная (смещение = 0, 1, IR0, IR1)
11 - косвенная (смещение = 0, 1, IR0, IR1)
dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 0 0 1 1 0 1 | T	dst	src1	src2

Описание: Разность между операндами src1 и src2 загружается в регистр dst. Операнды dst, src1 и src2 являются числами в формате с ПЗ.

Циклы: 1

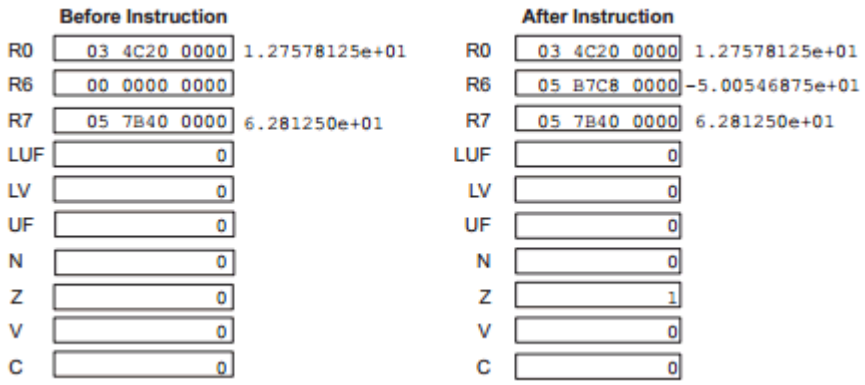
Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF 1 при возникновении отрицательного переполнения, иначе 0.
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: SUBF3 *AR0-- (IR0), *AR1, R4

	Before Instruction		After Instruction	
R4	00 0000 0000		05 1D00 0000	3.9250e+01
AR0	80 9888		80 9808	
AR1	80 9851		80 9851	
IR0	80		80	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
809888h	70C8000	1.4050e+02	70C8000	1.4050e+02
809851h	733C000	1.79750e+02	733C000	1.79750e+02

Пример 2: SUBF3 R7, R0, R6



SUBF3||STF Вычесть значение в формате с ПЗ и сохранить значение в формате с ПЗ

Синтаксис: SUBF3 src1, src2, dst1
|| STF src3, dst2

Операция: src2 - src1 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 | 0 1 0 1 | dst1 | src1	src3	dst2	src2

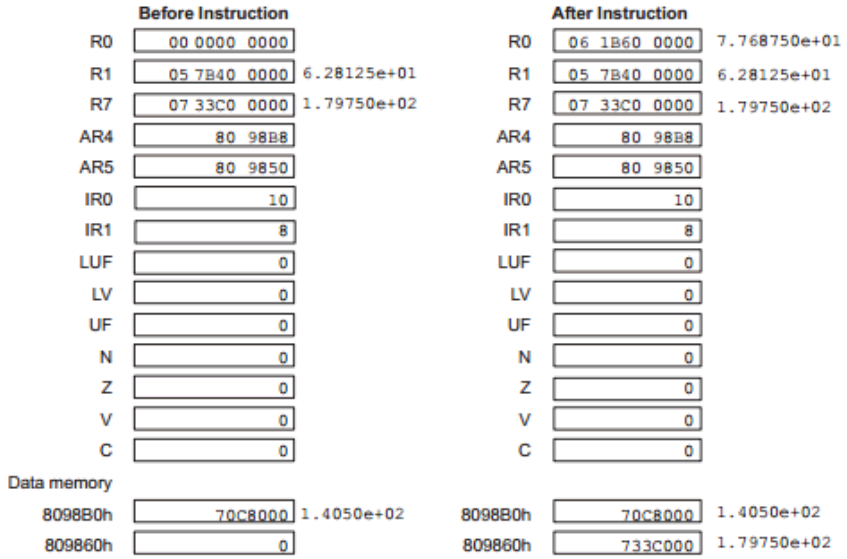
Описание: Вычитание в формате с ПЗ и сохранение числа в формате с ПЗ выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (SUBF3), записывает в тот же регистр, STF имеет на входе содержимое регистра до того, как оно было изменено командой SUBF3. Если src3 и dst1 указывают на один и тот же адрес, src3 считывается до записи в dst1.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF 1 при возникновении отрицательного переполнения, иначе 0.
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: SUBF3 R1, *-AR4 (IR1), R0
|| STF R7, *+AR5 (IR0)



SUBI Вычитание целых

Синтаксис: SUBI src, dst

Операция: dst-src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 0 0 0 0	G	dst	src

Описание: Разность операнда dst минус операнд src загружается в регистр dst. Операнды src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

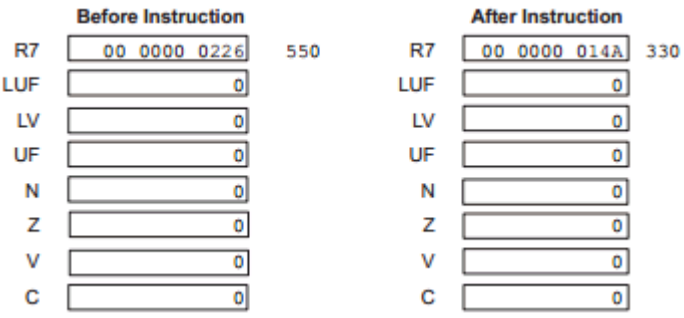
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBI 220, R7



SUBI3 Вычитание целых (3-х операндная)

Синтаксис: SUBI3 src2, src1, dst

Операция: src1 - src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23	16	15 8	7 0
0 0 1 | 0 0 1 1 1 0	T	dst	src	src

Описание: Разность операнда src1 минус операнд src2 загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

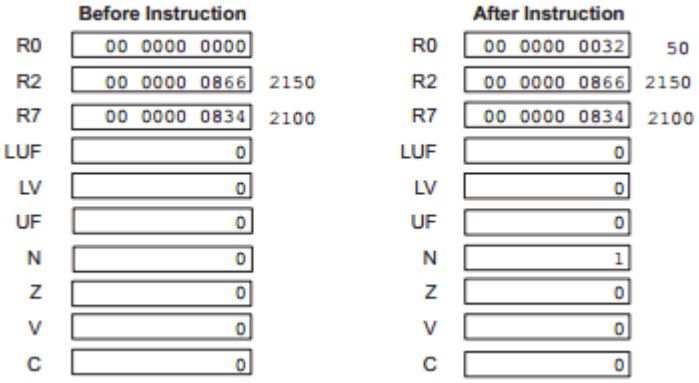
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

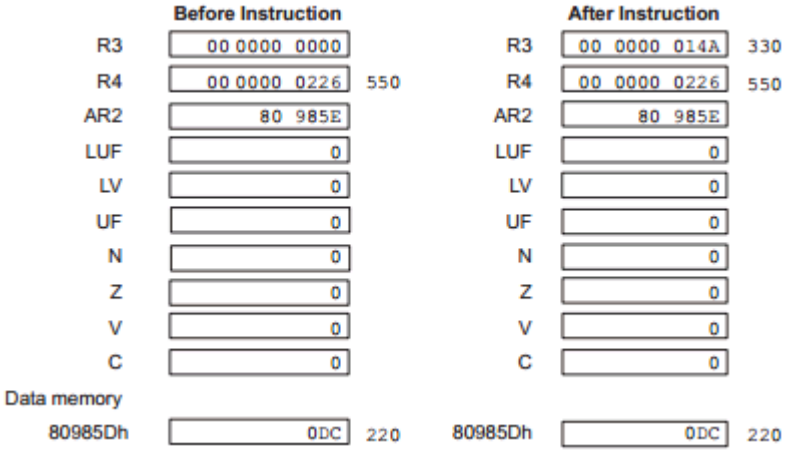
C 1 при возникновении замена, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1: SUBI3 R7, R2, R0



Пример 2: SUBI3 *-AR2 (1), R4, R3



SUBI3||STI Вычесть целое и сохранить целое

Синтаксис: SUBI3 src1, src2, dst1
|| STI src3, dst2

Операция: src2 - src1 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 1 1 0 1 1 0	dst1 src1	src3 dst2	src2

Описание: Целочисленное вычитание и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (SUBI3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой SUBI3. Если src3 и dst1 указывают на один и тот же адрес, src3 считывается до записи в dst1.

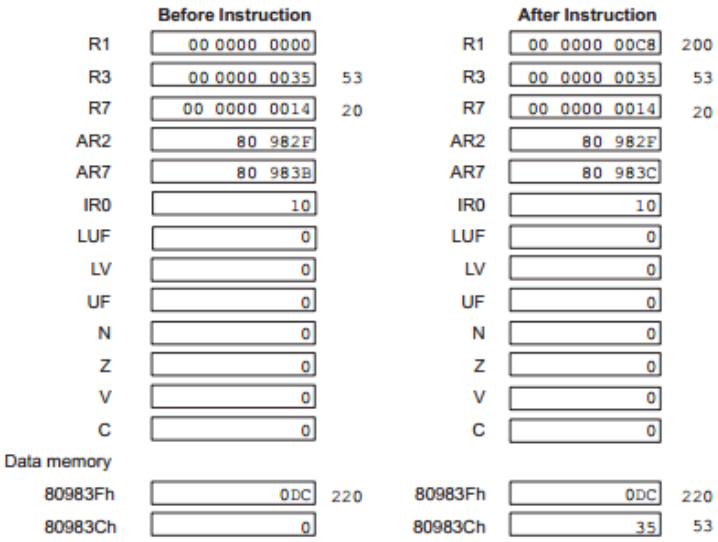
Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:
```
SUBI3 R7, *+AR2 (IR0), R1
|| STI R3, *++AR7
```



SUBRB Вычесть целое и сохранить целое

Синтаксис: SUBRB src, dst

Операция: src - dst - C → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 1 0 0 0 1 | G	dst	src	

Описание: Разность операндов src, dst и C загружается в регистр dst. Операнды dst и src являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

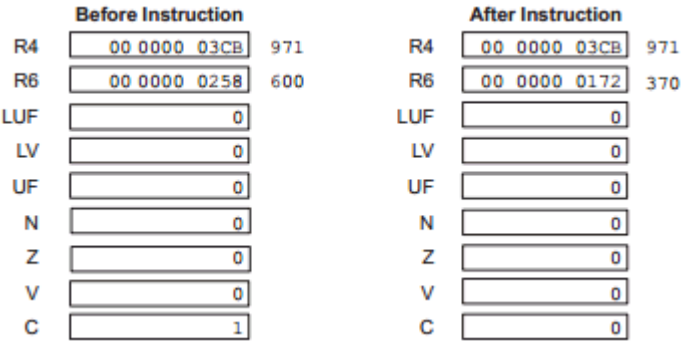
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBRB R4,R6



SUBRF Вычитание значений с плавающей запятой в обратном порядке

Синтаксис: SUBRF src, dst

Операция: src - dst → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 1 0 0 1 0 | G	dst	src	

Описание: Разность операнда src минус операнд dst загружается в регистр dst. Операнды dst и src являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

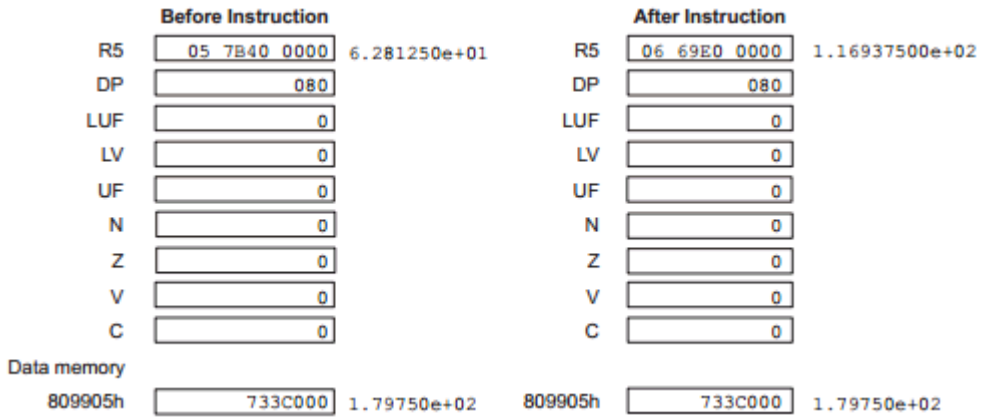
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: SUBRF @9905h, R5



SUBRI Вычитание целых в обратном порядке

Синтаксис: SUBRI src, dst

Операция: src - dst → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 0 1 1	G	dst	src

Описание: Разность операнда src минус операнд dst загружается в регистр dst. Операнды dst и src являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

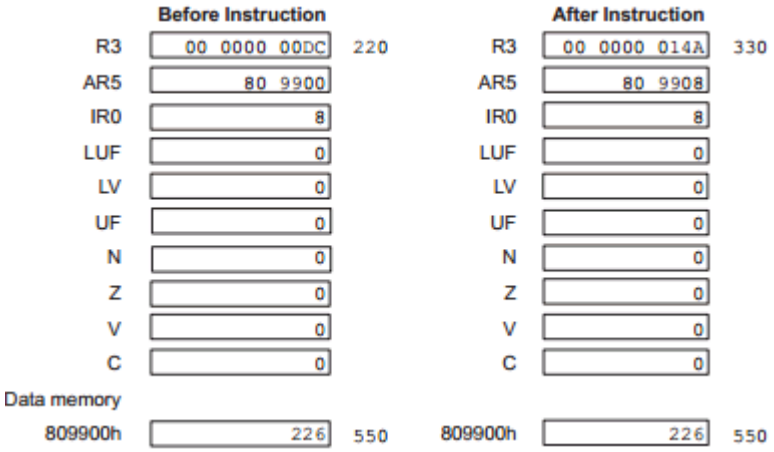
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заема, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBRI *AR5++ (IR0), R3



SWI Программное прерывание

Синтаксис: SWI

Операция: Выполняет прерывание эмуляции

Операнды: Нет

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 1 1 0 0	0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Описание: Команда SWI выполняет прерывание эмуляции. Эта команда является резервной и не может быть использована в обычном программировании.

Циклы: 4

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

TRAPcond Программное прерывание

Синтаксис: TRAPcond N

Операция: 0 → ST (GIE)
Если условие(cond) "истина":
Следующий PC → *++SP,
Вектор системного прерывания N → PC.
Иначе:
Установить GIE в начальное состояние.
Продолжение.

Операнды: N ($0 \le N \le 31$)

Опкод:

31 24	23 16	15 8	7 1	0
0 1 1 1 0 1 0	0 0 0 0	cond	0 0 0 0 0 0 0 0	0 0 0	N

Описание: Все прерывания запрещаются, когда 0 записывается в ST(GIE). Если условие истинно, содержимое счетчика команд (PC) загружается в си-

стемный стек, а в PC загружается содержимое указанного вектора системного прерывания (N). Если условие не "истина", ST(GIE) устанавливается по значению, которое он имел до того, как команда TRAPcond его изменила. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе 10.2 приведен список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 5

Разряды состояния: LUF Не изменяется.
LV Не изменяется.
UF Не изменяется.
N Не изменяется.
Z Не изменяется.
V Не изменяется.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: TRAPZ 16

	Before Instruction		After Instruction
PC	0123	PC	0010
SP	809870	SP	809871
ST	0	ST	0
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	0	N	0
Z	0	Z	0
V	0	V	0
C	0	C	0
Data memory			
Trap V.16	10	809871h	124

TSTB Проверка битовых полей

Синтаксис: TSTB src, dst

Операция: dst AND src (dst И src)

Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, $0 \le n \le 27$)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 1 0 1 0 0 | G	dst	src	

Описание: Поразрядное логическое И операндов dst и src формируется, но результат не загружается ни в один регистр. Таким образом, обеспечивается

неразрушающее сравнение. Предполагается, что операнды являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения

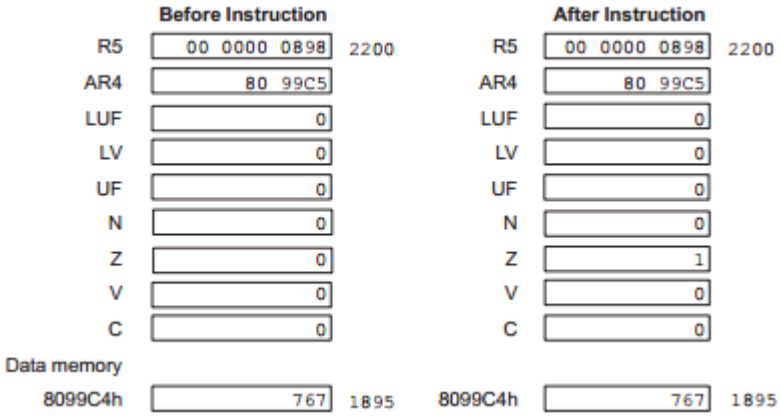
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: TSTB *-AR4 (1), R5



TSTB3 Проверка битовых полей (трехоперандная)

Синтаксис: TSTB3 src2, src1

Операция: src1 AND src2

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 0 0 1 1 1 1 | T	0 0 0 0 0	src1	src1

Описание: Поразрядное логическое И операндов src1 и src1 формируется, но результат не загружается ни в один регистр. Таким образом, обеспечивается сравнение без потери информации. Предполагается, что операнды являются беззнаковыми целыми. Хотя данная команда имеет только два

операнда, она обозначается как трехоперандная, т. к. операнды определены в трехоперандном формате.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения

Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: TSTB3 *AR5-- (IR0), *+AR0 (1)

	Before Instruction		After Instruction	
AR0	80 992C		80 992C	
AR5	80 9885		80 9805	
IR0	80		80	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		1	
V	0		0	
C	0		0	
Data memory				
809885h	898	2200	898	2200
80992Dh	767	1895	767	1895

Пример 2: TSB3 R4, *AR6-- (IR0)

	Before Instruction	After Instruction
R4	00 0000 FBC4	00 0000 FBC4
AR6	80 99F8	80 99F0
IR0	8	8
LUF	0	0
LV	0	0
UF	0	0
N	0	0
Z	0	0
V	0	0
C	0	0
Data memory		
8099F8h	1568	1568

XOR Поразрядное исключающее ИЛИ

Синтаксис: XOR src, dst

Операция: dst XOR src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 1 0 1	G	dst	src

Описание: Поразрядное исключающее-ИЛИ операндов src и dst загружается в регистр dst. Операнды dst и src являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения

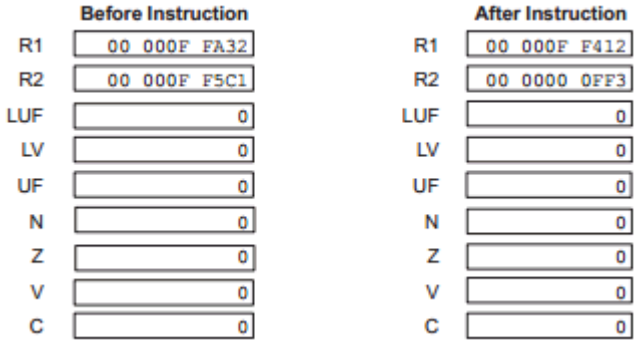
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: XOR R1, R2



XOR3 Поразрядное исключающее ИЛИ (трехоперандная)

Синтаксис: XOR3 src2, src1, dst

Операция: src1 XOR src2 → dst

Операнды: Трехоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трехоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

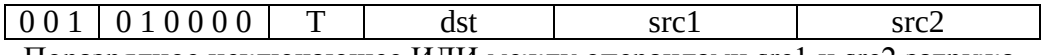
01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:



Описание: Поразрядное исключающее ИЛИ между операндами src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.
UF 0
N Старший значащий разряд выходного значения
Z 1 при генерации нулевого результата, иначе 0.
V 0
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: XOR3 *AR3++ (IR0), R7, R4

	Before Instruction	After Instruction
R4	00 0000 0000	00 0000 A53C
R7	00 0000 FFFF	00 0000 FFFF
AR3	80 9800	80 9810
IR0	10	10
LUF	0	0
LV	0	0
UF	0	0
N	0	0
Z	0	0
V	0	0
C	0	0
Data memory		
809800h	5AC3	5AC3

Пример 2: XOR3 R5, *-AR1 (1), R1

	Before Instruction	After Instruction
R1	00 0000 0000	00 0000 0F33
R5	00 000F FA32	00 000F FA32
AR1	80 9826	80 9826
LUF	0	0
LV	0	0
UF	0	0
N	0	0
Z	0	0
V	0	0
C	0	0
Data memory		
809825h	0FF5C1	0FF5C1

Примечание – См. 9.5.2 для выяснения влияния следования операций на счетчик циклов.

XOR3||STI Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ значений и сохранить целое

Синтаксис: XOR3 src2, src1, dst1
|| STI src3, dst2

Операция: src1 XOR src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 0 1 1 1 | dst1	src1	src3	dst2 | src2

Описание: Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR3) и сохранение целого(STI) выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (XOR3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой XOR3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения

Z 1 при генерации нулевого результата, иначе 0.

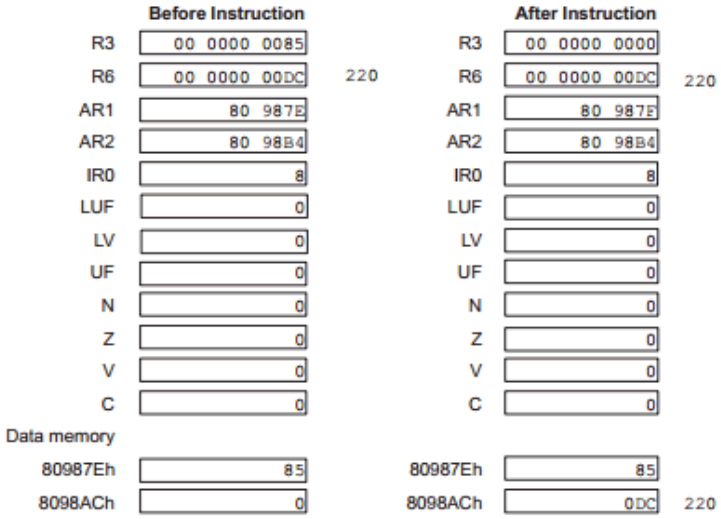
V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
XOR3        *AR1++, R3, R3
|| STI R6,   *-AR2 (IR0)
```



11 Назначение и расположение выводов, описание сигналов ПЦОС

В данном разделе приведено расположение выводов, описание сигналов ПЦОС.

11.1 Описание сигналов ПЦОС

Сигналы ПЦОС представлены в таблице 11.1, в которой приведено описание каждого сигнала, количество выводов и режимы работы, т. е. входы, выходы или высокоимпедансное состояние, обозначенные, соответственно, I, O или Z. Все выводы, обозначенные «NC», должны остаться неподключенными. Знак «#» после названия сигнала (например, RESET#) обозначает, что сигнал имеет активный низкий уровень – логический ноль.

Сигналы в таблице 11.1 сгруппированы по функциям.

Таблица 11.1 – Описание сигналов ПЦОС

Сигнал	Количество выводов	I/O/Z*	Описание	Состояние сигнала Z**
1	2	3	4	5
Интерфейс основной шины (61 вывод)				
D31-D0	32	I/O/Z	32-разрядный порт данных интерфейса основной шины	S H R
A23-A0	24	O/Z	24-разрядный порт адреса интерфейса основной шины	S H R
R/W#	1	O/Z	Сигнал чтение/запись для интерфейса в высоком уровне, если выполняется чтение, и в низком уровне, если выполняется запись через параллельный интерфейс	S H R
STRB#	1	O/Z	Строб внешнего доступа для интерфейса основной шины	S H
RDY#	1	I	Сигнал готовности. Этот вывод показывает, что внешнее устройство готово для завершения транзакции интерфейса основной шины	S
HOLD#	1	I	Сигнал удержания интерфейса основной шины. Когда HOLD# в низком уровне, любое входное сообщение завершается. Сигналы A23-A0, D31-D0, STRB# и R/W# устанавливаются в высокоимпедансное состояние, и все входные сообщения по интерфейсу основной шины удерживаются до тех пор, пока HOLD# не установится в высокий уровень, или пока не будет установлен бит NOHOLD в регистре управления основной шины	S
HOLDA#	1	O/Z	Сигнал подтверждения удержания интерфейса основной шины. Этот сигнал вырабатывается в ответ на логически низкий HOLD#. Сигналы A23-A0, D31-D0, STRB# и R/W# находятся в высокоимпедансном состоянии, и все транзакции по шине будут удерживаться. HOLDA# будет установлен в высокий уровень в ответ на установку высокого уровня HOLD#	S

Продолжение таблицы 11.1

1	2	3	4	5
Интерфейс расширенной шины (49 выводов)				
XD(31-0)	32	I/O/Z	32-разрядный порт данных интерфейса расширенной шины	S R
XA(12-0)	13	O/Z	13-разрядный порт адреса интерфейса расширенной шины	S R
XR/W#	1	O/Z	Сигнал чтение/запись интерфейса шины расширения. При выполнении чтения этот вывод удерживается в высоком уровне; при выполнении записи этот вывод будет находиться в низком уровне	S R
MSTRB#	1	O/Z	Строб доступа к внешней памяти интерфейса шины расширения	S
IOSTRB#	1	O/Z	Строб доступа внешнего ввода-вывода интерфейса расширенной шины	S
XRDY#	1	I	Сигнал готовности. Этот вывод показывает, что внешнее устройство готово для завершения транзакции интерфейса расширенной шины	S
Сигналы управления (9 выводов)				
RESET#	1	I	Сброс. Если этот вывод в низком уровне, устройство устанавливается в состояние сброса. После установки в сброс выполнение начинается с адреса, определяемого вектором сброса	
INT3# – INT0#	4	I	Вход сигнала внешнего прерывания 3-0	
IACK#	1	O/Z	Сигнал подтверждения прерывания. IACK# устанавливается в 0 во время выполнения команды IACK. Это может быть использовано для того, чтобы указать начало или конец программы обработки прерывания	S
MC/MP#	1	I	Режим микроконтроллер/режим микропроцессор	
XF1, XF0	2	I/O/Z	Выводы внешних флагов. Они используются как входы/выходы общего назначения или поддерживают команды блокировки процессора	S R
Сигналы последовательного порта 0 (6 выводов)				
CLKX0	1	I/O/Z	Тактирование передачи последовательного порта 0. Этот вывод служит в качестве последовательного синхроимпульса сдвига передатчика последовательного порта 0	S R
DX0	1	I/O/Z	Выход передачи данных. Последовательный порт 0 передает последовательные данные через этот вывод	S R
FSX0	1	I/O/Z	Кадровый синхроимпульс передачи. Импульс FSX0 начинает процесс передачи данных через вывод DX0	S R
CLKR0	1	I/O/Z	Синхронизация приема последовательного порта 0. Этот вывод служит в качестве последовательного синхроимпульса сдвига приемника последовательного порта 0	S R
DR0	1	I/O/Z	Прием данных. Последовательный порт 0 получает последовательные данные через вывод DR0	S R
FSR0	1	I/O/Z	Кадровый синхроимпульс приема. Импульс FSR0 начинает процесс приема данных через вывод DX0	S R

Продолжение таблицы 11.1

1	2	3	4	5
Сигналы последовательного порта 1 (6 выводов)				
CLKX1	1	I/O/Z	Тактирование передачи последовательного порта 1. Этот вывод служит в качестве последовательного синхроимпульса сдвига передатчика последовательного порта 1	S R
DX1	1	I/O/Z	Выход передачи данных. Последовательный порт 1 передает последовательные данные через этот вывод	S R
FSX1	1	I/O/Z	Кадровый синхроимпульс передачи. Импульс FSX1 начинает процесс передачи данных через вывод DX1	S R
CLKR1	1	I/O/Z	Синхронизация приема последовательного порта 1. Этот вывод служит в качестве последовательного синхроимпульса сдвига приемника последовательного порта 1	S R
DR1	1	I/O/Z	Прием данных. Последовательный порт 1 получает последовательные данные через вывод DR1	S R
FSR1	1	I/O/Z	Кадровый синхроимпульс приема. Импульс FSR1 начинает процесс приема данных через вывод DX1	S R
Сигналы таймера 0 (1 вывод) и таймера 1 (1 вывод)				
TCLK0	1	I/O/Z	Синхронизация таймера: как вход – TCLK0 используется таймером 0 для счета внешних импульсов; как выход – TCLK0 выдает импульсы, генерируемые таймером 0	S R
TCLK1	1	I/O/Z	Синхронизация таймера: как вход – TCLK1 используется таймером 1 для счета внешних импульсов; как выход – TCLK1 выдает импульсы, генерируемые таймером 1	S R
Выводы питания/земли (24 вывода)				
#VCC1	8	–	Вывод питания периферийных буферов ввода-вывода ***	
#VCC2	4	–	Вывод питания ядра процессора ***	
#GND1	5	–	Вывод «земли» периферийных буферов ввода-вывода	
#GND2	7	–	Вывод «земли» ядра процессора	
Сигналы тактирования процессора (5 выводов)				
CLKMD	1	I	Сигнал выбора режима тактирования: если CLKMD=0, то CLKCPU=1/2 × X2/CLKIN; если CLKMD=1, то CLKCPU=1 × X2/CLKIN	
X1	1	O/Z	Выход внутреннего генератора для подключения кварцевого резонатора. Если резонатор не используется – вывод должен остаться неподключенным	S
X2/CLKIN	1	I	Вход внутреннего генератора для кварца или генератора тактовых импульсов	S
H1	1	O/Z	Внешний сигнал синхронизации H1. Тактовый генератор имеет период, равный двум CLKIN	S
H3	1	O/Z	Внешний сигнал синхронизации H3. Тактовый генератор имеет период, равный двум CLKIN	S

Окончание таблицы 11.1

1	2	3	4	5
Резервные сигналы (18 выводов)				
EMU0-EMU2	3	I	Зарезервированный вывод. PullUp	
EMU3	1	O	Зарезервированный вывод	
EMU4/SHZ#	1	I	Зарезервированный вывод	
EMU5, EMU6	2	O	Зарезервированный вывод	
RSV0-RSV4	5	I	Зарезервированный вывод (должен быть подключен к #VCC1)	
RSV5-RSV10	6	–	Зарезервированный вывод (должен остаться неподключенным)	

Примечания

1 Все выводы питания должны быть подключены к общему питанию (плюс 3,3 В), и все выводы земли должны быть подключены к общей земле.

2 Все резисторы нагрузки должны быть от 18 до 20 кОм.

_____

* Принятые условные обозначения: I – вход, O – выход, Z – высокоимпедансное состояние.

** S = активен SHZ#, H = активен HOLD#, R = активен RESET#.

*** Рекомендуется подключать емкость 0,1 мкФ.

11.2 Расположение и назначение выводов ПЦОС

Расположение выводов приведено на рисунке 11.1.

Назначение выводов приведено в таблице 11.2.

На рисунке 11.2 представлено условное графическое обозначение ИС 1867ВЦ11Ф.

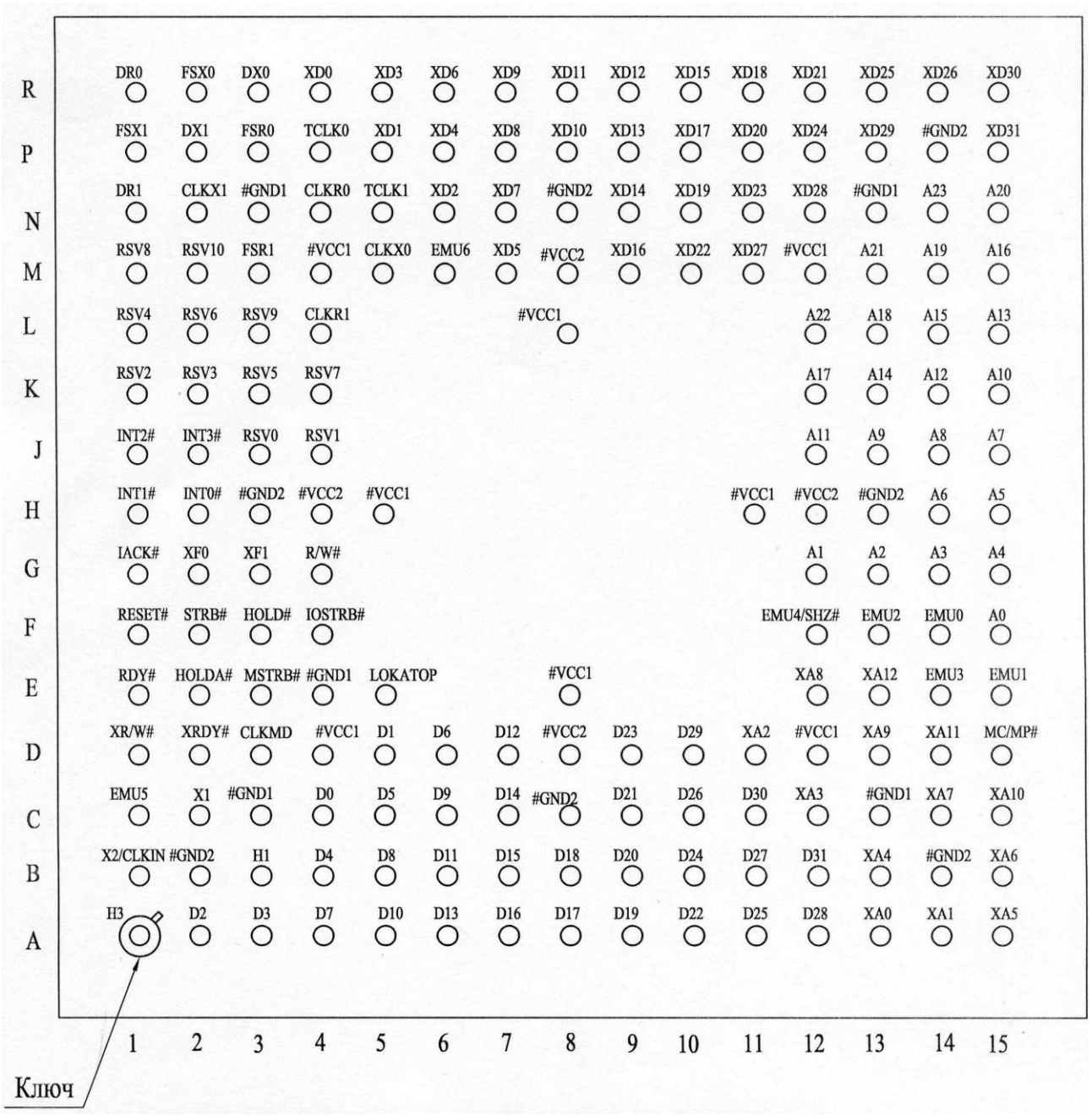


Рисунок 11.1 – Расположение выводов ИС 1867ВЦ11Ф

Таблица 11.2 – Функциональное назначение выводов

Номер вывода корпуса	Обозначение вывода	Функциональное назначение вывода		Тип вывода
C4, D5, A2, A3, B4, C5, D6, A4, B5, C6, A5, B6, D7, A6, C7, B7, A7, A8, B8, A9, B9, C9, A10, D9, B10, A11, C10, B11, A12, D10, C11, B12	D0-D31	Интерфейс основной шины	Вход/выход 32-разрядной шины данных	I/O/Z
F15, G12, G13, G14, G15, H15, H14, J15, J14, J13, K15, J12, K14, L15, K13, L14, M15, K12, L13, M14, N15, M13, L12, N14	A0-A23		Выход 24-разрядной шины адреса	O/Z
G4	R/W#		Сигнал чтения/записи	O/Z
F2	STRB#		Сигнал внешнего доступа	O/Z
El	RDY#		Сигнал готовности	I
F3	HOLD#		Сигнал удержания	I
E2	HOLDA#		Сигнал подтверждения удержания	O/Z
R4, P5, N6, R5, P6, M7, R6, N7, P7, R7, P8, R8, R9, P9, N9, R10, M9, P10, R11, N10, P11, R12, M10, N11, P12, R13, R14, M11, N12, P13, R15, P15	XD0-XD31	Интерфейс расширенной шины	Вход/выход 32-разрядной шины данных	I/O/Z
A13, A14, D11, C12, B13, A15, B15, C14, E12, D13, C15, D14, E13	XA0-XA12		Выход 13-разрядной шины адреса	O/Z
Dl	XR/W#		Сигнал чтения/записи	O/Z
E3	MSTRB#		Строб доступа к внешней памяти	O/Z
F4	IOSTRB#		Строб доступа внешнего ввода-вывода	O/Z
D2	XRDY#		Сигнал готовности	I
F1	RESET#	Выводы управления	Сброс	I
H2, H1, J1, J2	INT0#-INT3#		Вход сигнала внешнего прерывания 0-3	I
G1	IACK#		Сигнал подтверждения прерывания	O/Z
D15	MC/MP#		Режим микроконтроллер/ микропроцессор	I
G2, G3	XF0, XF1		Вывод внешних флагов	I/O/Z
M5	CLKX0	Последовательный порт 0	Тактовый вход передачи данных	I/O/Z
R3	DX0		Выход для передачи последовательных данных	I/O/Z
R2	FSX0		Вход/выход сигнала кадровой синхронизации для передачи данных	I/O/Z
N4	CLKR0		Тактовый вход приема данных	I/O/Z
R1	DR0		Вход приема данных	I/O/Z
P3	FSR0		Вход/выход сигнала кадровой синхронизации для приема данных	I/O/Z
N2	CLKX1	Последовательный порт 1	Тактовый вход передачи данных	I/O/Z
P2	DX1		Выход для передачи последовательных данных	I/O/Z
PI	FSX1		Вход/выход сигнала кадровой синхронизации для передачи данных	I/O/Z
L4	CLKR1		Тактовый вход приема данных	I/O/Z
N1	DR1		Вход приема данных	I/O/Z
M3	FSR1		Вход/выход сигнала кадровой синхронизации для приема данных	I/O/Z

Окончание таблицы 11.2

Номер вывода корпуса	Обозначение вывода	Функциональное назначение вывода		Тип вывода
P4	TCLK0	ТПМ	Синхронизация таймера 0	I/O/Z
N5	TCLK1		Синхронизация таймера 1	I/O/Z
C2	X1	Тактирование процессора	Вывод 1 для кварца, выход тактового генератора	O
B1	X2/CLKIN		Вывод 2 для кварца/ вход внешнего тактирования	I
D3	CLKMD		Сигнал выбора режима тактирования: если CLKMD=0, то CLKCPU=1/2 × X2/CLKIN; если CLKMD=1, то CLKCPU=1 × X2/CLKIN	I
B3	H1		Внешний сигнал синхронизации	O/Z
A1	H3			O/Z
F14, E15, F13	EMU0-EMU2	Зарезервированные сигналы	Зарезервированный вывод. Pull-Up	I
E14, C1, M6	EMU3, EMU5, EMU 6		Зарезервированный вывод	O
F12	EMU4/SHZ#		Зарезервированный вывод	I
J3, J4, K1, K2, L1	RSV0-RSV4		Зарезервированный вывод (должен быть подключен к #VCC1)	I
K3, L2, K4, M1, L3, M2	RSV5-RSV10		Зарезервированный вывод (должен остаться неподключенным)	–
H5, M4, L8, M12, H11, D12, E8, D4	#VCC1	Выводы питания/земли	Вывод питания периферийных буферов ввода-вывода (perif_vdd)	–
H4, M8, H12, D8	#VCC2		Вывод питания ядра процессора (vdd)	–
C3, E4, N3, N13, C13	#GND1		Вывод «земли» периферийных буферов ввода-вывода (perif_gnd)	–
B2, H3, N8, P14, H13, B14, C8	#GND2		Вывод «земли» ядра процессора (core_gnd)	–

Примечания

1 Знак # после обозначения вывода в графе «Обозначение вывода» означает инверсию (сигнал активен нулем).

2 В графе «Тип вывода»: I – вход; O – выход; Z – третье состояние.

3 Выводы питания ядра процессора D8, H4, H12, M8 объединены шиной #VCC2 внутри корпуса и выведены на верх основания корпуса на две контактные площадки 3a, 3b по диагонали A1-R15.

4 Выводы «земли» ядра процессора B2, B14, C8, H3, H13, N8, P14 объединены шиной #GND2 внутри корпуса и выведены на верх основания корпуса на две контактные площадки 2a, 2b по диагонали A1-R15.

5 Выводы питания периферийных буферов ввода-вывода H5, M4, L8, M12, H11, D12, E8, D4 объединены шиной #VCC1 внутри корпуса и выведены на верх основания корпуса на две контактные площадки 1a, 1b по диагонали A15-R1.

6 Выводы «земли» периферийных буферов ввода-вывода C3, C13, E4, N3, N13 объединены шиной #GND1 внутри корпуса и выведены на верх основания корпуса на две контактные площадки 4a, 4b по диагонали A15-R1.

7 Допускается на площадки 1a-4a, 1b-4b устанавливать безвыводные фильтрующие конденсаторы.

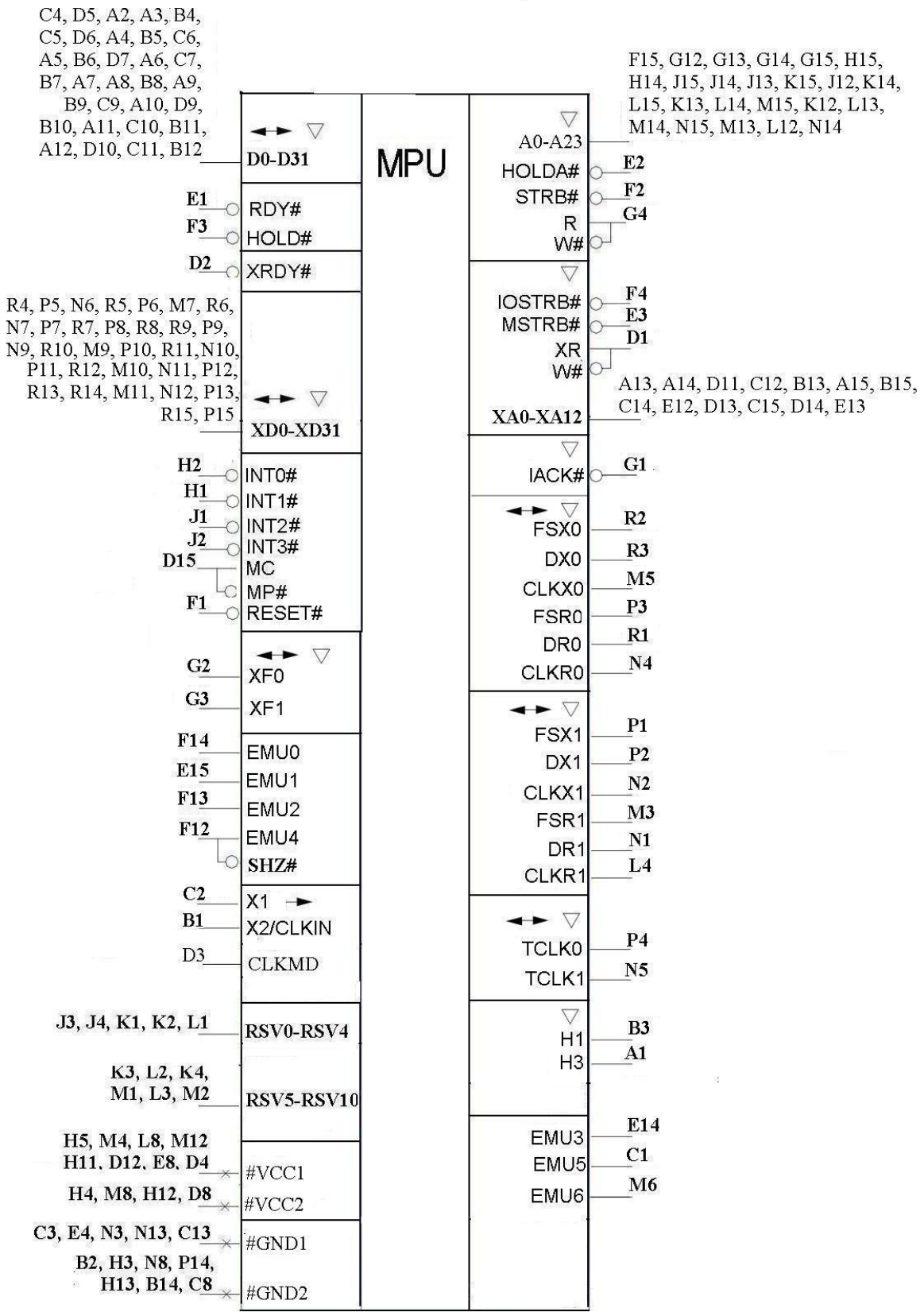


Рисунок 11.2 – Условное графическое обозначение ИС 1867ВЦ11Ф

12 Электрические параметры микросхемы

12.1 Электрические параметры микросхемы 1867ВЦ11Ф должны соответствовать нормам, приведенным в таблице 12.1.

Таблица 12.1 – Электрические параметры микросхемы

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, ºС
		не менее	не более	
1	2	3	4	5
1 Выходное напряжение низкого уровня по выводам A0 – A23, D0 – D31, EMU3, EMU5, EMU6, XA0 – XA12, XD0 – XD31, CLKR0, CLKR1, CLKX0, CLKX1, DR0, DR1, DX0, DX1, FSR0, FSR1, FSX0, FSX1, TCLK0, TCLK1, HOLDA#, R/W#, XR/W#, XF0, XF1, IACK#, IOSTRB#, MSTRB#, STRB#, H1, H3, B, $U_{CC1} = U_{CC2} = 3{,}0$ В, $I_{OL} = 2{,}0$ мА	U_{OL}	–	0,4	
2 Выходное напряжение высокого уровня по выводам A0 – A23, D0 – D31, EMU3, EMU5, EMU6, XA0 – XA12, XD0 – XD31, CLKR0, CLKR1, CLKX0, CLKX1, DR0, DR1, DX0, DX1, FSR0, FSR1, FSX0, FSX1, TCLK0, TCLK1, HOLDA#, R/W#, XR/W#, XF0, XF1, IACK#, IOSTRB#, MSTRB#, STRB#, H1, H3, B, $U_{CC1} = U_{CC2} = 3{,}0$ В, $I_{OH} = -2{,}0$ мА	U_{OH}	2,4	–	
3 Ток утечки низкого уровня на входах RDY#, HOLD#, XRDY#, RESET#, EMU4/SHZ#, MC/MP#, INT0# – INT3#, D0 – D31, XD0 – XD31, XF0, XF1, CLKR0, CLKR1, CLKX0, CLKX1, DR0, DR1, DX0, DX1, FSR0, FSR1, FSX0, FSX1, TCLK0, TCLK1, X2/CLKIN, CLKMD[1)], мкА, $U_{CC1} = U_{CC2} = 3{,}6$ В, $U_{IL} = 0$ В	I_{ILL}	–30	–	–60 ± 3 25 ± 10 85 ± 3
4 Ток утечки высокого уровня на входах RDY#, HOLD#, XRDY#, RESET#, EMU4/SHZ#, MC/MP#, INT0# – INT3#, D0 – D31, XD0 – XD31, XF0, XF1, CLKR0, CLKR1, CLKX0, CLKX1, DR0, DR1, DX0, DX1, FSR0, FSR1, FSX0, FSX1, TCLK0, TCLK1, X2/CLKIN, CLKMD, EMU0 – EMU2[1)], мкА, $U_{CC1} = U_{CC2} = 3{,}6$ В, $U_{IH} = U_{CC1}$	I_{ILH}	–	30	
5 Входной ток низкого уровня по входам EMU0 – EMU2, мкА, $U_{CC1} = U_{CC2} = 3{,}6$ В, $U_{IL} = 0$ В	I_{IL}	–600	–	
6 Динамический ток потребления[2)], мА, $U_{CC1} = U_{CC2} = 3{,}6$ В, $f_{CI} = 50$ МГц	I_{OCC}	–	800	
7 Функциональный контроль $U_{CC1} = U_{CC2} = (3{,}0; 3{,}6)$ В, $f_{CI} = (4; 50)$ МГц	ФК	–	–	

Окончание таблицы 12.1

_____

[1)] Параметры I_{ILL}, I_{ILH} при температуре минус 60 ºС не измеряются, а гарантируются нормами при температуре (25 ± 10) ºС.

[2)] Параметр измеряется при отключенных от нагрузок выходах и подключенных к уровням U_{IL} и U_{IH} входах.

12.2 Значения предельно допустимых и предельных режимов эксплуатации в диапазоне рабочих температур среды должны соответствовать нормам, приведенным в таблице 12.2.

Таблица 12.2 – Предельно допустимые и предельные режимы эксплуатации в диапазоне рабочих температур

Наименование параметра режима, единица измерения	Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1 Напряжение питания буферов ввода-вывода, В	U_{CC1}	3,0	3,6	–0,3	4,0
2 Напряжение питания ядра, В	U_{CC2}	3,0	3,6	–0,3	4,0
3 Входное напряжение низкого уровня, В	U_{IL}	–0,3	0,8	–0,5	–
4 Входное напряжение высокого уровня, В	U_{IH}	$0,7U_{CC1}$	U_{CC1}	–	$U_{CC1}+0,3$
5 Напряжение на выходе с третьим состоянием в состоянии «Выключено», В	U_{OZ}	0	U_{CC1}	–0,2	$U_{CC1}+0,3$
6 Выходной ток низкого уровня, мА	I_{OL}	–	2,0	–	4,0
7 Выходной ток высокого уровня, мА	I_{OH}	–2,0	–	–4,0	–
8 Частота следования импульсов тактового сигнала, МГц	f_{CI}	4	50	–	–
9 Емкость нагрузки выходов, пФ	C_L	–	40	–	–
Примечание – Время работы в одном из предельных режимов должно быть не более 5 с.					

12.3 Порядок подачи и снятия напряжений питания и входных сигналов на микросхемы должен быть следующим: сначала подается напряжение питания буферов ввода-вывода U_{CC1}, затем напряжение питания ядра U_{CC2} (или одновременно), а затем входные напряжения низкого уровня U_{IL} и высокого уровня U_{IH} (или одновременно).

Порядок снятия напряжений с выводов микросхемы при выключении должен быть обратным подаче: первыми снимаются входные сигналы, а затем напряжения питания.

12.4 Микросхемы 1867ВЦ11Ф должны быть устойчивы к воздействию статического электричества с потенциалом не менее 2 000 В.

12.5 Электрические параметры микросхемы, изменяющиеся в процессе и после воздействия специальных факторов, в том числе в диапазоне рабочих температур, должны соответствовать нормам, приведенным в таблице 12.3.

Таблица 12.3 – Электрические параметры микросхемы, изменяющиеся в процессе и после воздействия специальных факторов

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, ºС
		не менее	не более	
1 Выходное напряжение низкого уровня по выводам A0 – A23, D0 – D31, EMU3, EMU5, EMU6, XA0 – XA12, XD0 – XD31, CLKR0, CLKR1, CLKX0, CLKX1, DR0, DR1, DX0, DX1, FSR0, FSR1, FSX0, FSX1, TCLK0, TCLK1, HOLDA#, R/W#, XR/W#, XF0, XF1, IACK#, IOSTRB#, MSTRB#, STRB#, H1, H3, В, $U_{CC1} = U_{CC2} = 3{,}0$ В, $I_{OL} = 2{,}0$ мА	U_{OL}	–	0,4 1,0[1), 2)]	–60 ± 3 25 ± 10 85 ± 3
2 Выходное напряжение высокого уровня по выводам A0 – A23, D0 – D31, EMU3, EMU5, EMU6, XA0 – XA12, XD0 – XD31, CLKR0, CLKR1, CLKX0, CLKX1, DR0, DR1, DX0, DX1, FSR0, FSR1, FSX0, FSX1, TCLK0, TCLK1, HOLDA#, R/W#, XR/W#, XF0, XF1, IACK#, IOSTRB#, MSTRB#, STRB#, H1, H3, В, $U_{CC1} = U_{CC2} = 3{,}0$ В, $I_{OH} = -2{,}0$ мА	U_{OH}	2,4 2,0[1), 2)]	–	
3 Ток утечки высокого уровня на входах RDY#, HOLD#, XRDY#, RESET#, EMU4/SHZ#, MC/MP#, INT0# – INT3#, D0 – D31, XD0 – XD31, XF0, XF1, CLKR0, CLKR1, CLKX0, CLKX1, DR0, DR1, DX0, DX1, FSR0, FSR1, FSX0, FSX1, TCLK0, TCLK1, X2/CLKIN, CLKMD, EMU0 – EMU2, мкА, $U_{CC1} = U_{CC2} = 3{,}6$ В, $U_{IL} = 0$ В	I_{ILH}	–	180[2)]	
4 Динамический ток потребления, мА, $U_{CC1} = U_{CC2} = 3{,}6$ В, $f_{CI} = 50$ МГц	I_{OCC}	–	1 200[2)]	
5 Функциональный контроль $U_{CC1} = U_{CC2} = (3{,}0; 3{,}6)$ В, $f_{CI} = 50$ МГц	ФК	–	–	

_____

[1)] Во время воздействия факторов с характеристиками 7.$И_6$.

[2)] Во время и после воздействия факторов с характеристиками 7.$И_7$, 7.$С_4$, 7.$К_1$, 7.$К_4$.

13 Заключение

В настоящем руководстве КФДЛ.431299.033 приведены архитектура, функциональное построение, система команд и особенности применения цифровых процес-соров обработки сигналов (ПЦОС) с фиксированной и плавающей запятой 1867ВЦ11Ф.

Все значения электрических параметров ИС приведены в технических условиях АЕНВ.431280.141. Значения параметров, приведенные в руководстве, являются справочными.

Настоящее руководство может служить практическим пособием по применению ПЦОС с фиксированной и плавающей запятой 1867ВЦ11Ф для разработчиков систем на основе ИС.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Подпись	Дата
	измененных	замененных	новых	аннулированных				
-	-	-	все	-	281			17.04.17
1	1	-	-	-	-			19.04.17