Утвержден

25.00152-01 97 01-1-ЛУ

Операционная система МОС ВП

АРХИТЕКТУРА И СИСТЕМА МАШИННЫХ ИНСТРУКЦИЙ

Справочный материал

25.00152-01 97 01-1

Листов 293/4К

1987

Перв. примен.

25.00152-01

Литера О

АННОТАЦИЯ

Документ предназначен для системных программистов в качестве первоначального материала для анализа исключительных ситуаций, возникающих при выполнении инструкций, анализа дампов программ, разработки операционных систем для СМ 1700. Данный документ описывает общую структуру семейства 32-разрядных СМ ЭВМ, первой базовой моделью которого является СМ 1700. Данный документ не предназначен для целей обучения ассемблеру СМ 1700.

Документ содержит описание функционирования инструкций исполняющихся центральным процессором СМ 1700 и исключительных ситуаций, которые могут наступить при выполнении инструкции. Описание выполнено по следующей схеме:

- полное наименование инструкции;
- формат инструкции;
- коды условий, устанавливаемые инструкцией;
- возможные исключительные ситуации;
- коды операций;
- описание инструкции;
- примечания.

Документ состоит из двух частей. Первая часть документа содержит 4 раздела.

В разделе 1 описаны терминология, принятые соглашения и сокращения, используемые для описания инструкций.

В разделе 2 описаны типы данных, с которыми работают инструкции. В этом разделе описано также слово состояния процессора и исключительные ситуации, которые могут возникнуть при выполнении инструкции. В разделе 3 описаны режимы адресации операндов, используемые в инструкциях СМ 1700.

В разделе 4 приводится описание инструкций СМ 1700.

СОДЕРЖАНИЕ

Операционная система МОС ВП. Архитектура и система машинных инструкций. Справочный материал. 00152-01 97 01-2

1. Введение

1.1. Архитектура модели СМ 1700

Архитектура СМ 1700 является дальнейшим расширением архитектурных возможностей семейства СМ ЭВМ. В ранних 16-разрядных моделях СМ ЭВМ (типа СМ 4, СМ 1420) и в СМ 1700 много общих черт, включая байтовую адресацию, похожие структуры ввода-вывода и системы инструкций, те же форматы данных.

По сравнению с ранними моделями СМ ЭВМ в ЭВМ СМ 1700 обеспечивается гораздо большее пространство виртуальных адресов, дополнительные типы данных, инструкции, новые режимы адресации. Кроме того, предусмотрены более сложный механизм распределения и защиты памяти, аппаратные средства диспетчеризации и синхронизации.

При разработке СМ 1700 преследовались следующие основные цели:

- максимальная совместимость с ранними моделями СМ ЭВМ при значительном расширении виртуального адресного пространства и функциональных возможностей;
- эффективное использование памяти. Это достигается за счет использования широкого диапазона используемых типов данных и введения новых режимов адресации;
- систематизированный, стройный набор инструкций с ортогональностью операторов, типов данных и режимов адресации. Это облегчает использование набора инст-

рукций особенно для трансляторов языков высокого уровня;

- возможность расширения. Набор инструкций спроектирован с учётом возможности эффективного включения новых типов данных и операторов, которые могут быть использованы наряду с существующими;
- область применения. Архитектурные возможности позволяют использовать СМ 1700 во всех областях, где применяются выпускаемые в настоящий момент эвм.

Документ может быть использован для всех операционных систем на базе СМ 1700 в качестве справочного материала.

1.2. Терминология и соглашения

1.2.1. Представление чисел

Все числа, за исключением специально оговоренных случаев, считаются представленными в десятичном виде. В тех случаях, когда возможно неоднозначное понимание записи, недесятичные числа сопровождаются записью основания в скобках. Например, FF (шестнадцатеричное).

1.2.2. Непредсказуемые и неопределенные результаты

Результаты, которые определяются как непредсказуемые, могут изменяться в зависимости от момента времени, используемых инструкций. Функционирование программных средств

никогда не может зависеть от результатов, которые определяются как непредсказуемые. Операции, которые определяются как неопределенные, могут также изменяться в зависимости от момента времени, используемых инструкций.

Возможен широкий диапазон последствий в результате неопределенной операции: от отсутствия каких-либо последствий до останова работы операционной системы. Неопределенные операции не должны приводить к такому состоянию центрального процессора, из которого невозможен переход к нормальному выполнению инструкций. Отметим, что между понятиями результата и операции имеется различие. Программные средства, работающие в непривилегированном режиме, не могут вызвать неопределенные операции.

1.2.3. Диапазоны и поля

Определение диапазона изменений значений является включительным. Например, диапазон целых чисел от 0 до 4 включает в себя целые числа 0, 1, 2, 3 и 4. Поле определяется парой цифр, разделяемых двоеточием (:) и также является включительным. Например, разряды 7:3 определяют поле разрядов, включающее разряды 7, 6, 5, 4 и 3.

1.2.4. Поля, обозначенные MBZ

Поля, обозначенные как MBZ (должен быть нулем), никогда не должны заполняться программными средствами значениями, отличными от нуля. Если центральный процессор обнаружит

ненулевое значение в тех полях, которые определены как MBZ, и, если это поле доступно программным средствам, работающим в непривилегированном режиме, то возникает ошибка резервного операнда или произойдет прекращение операции.

Поля, обозначенные как MBZ и доступные непривилегированным программам, не могут быть проверены на нулевое значение. В этом случае ненулевое значение в таких полях может привести к неопределенной операции.

1.2.5. Резервные коды

Неиспользованные значения полей зарезервированы для использования в будущем. Во многих случаях эти значения могут быть предназначены для пользователей. Эти значения могут быть использованы для нестандартных применений. Значения, определенные как резервные, и поля, определенные как MBZ, предназначены для расширения архитектурных возможностей в будущем.

1.2.6. Обозначения на рисунках

В тех местах, где показаны форматы регистров или ячейки памяти, адресация увеличивается справа налево и сверху вниз.

1.2.7. Описание инструкций

Формат инструкций задается, используя определенные соглашения по описанию кода операции и спецификаций операн-

да.

Код операции, его спецификация, состоит из мнемонического наименования инструкции, используемых типов данных и количества операндов. Содержимое квадратных скобок определяет список возможных типов данных, один из которых должен быть указан. Последним символом в коде операции может быть цифра, которая определяет количество используемых операндов в данной инструкции.

Описание операнда, его спецификация, состоит из мнемонического имени, типа доступа и типа данных. В списках операнда содержимое квадратных скобок определяет все предполагаемые операнды.

Формат операнда

имя.<тип доступа><тип данных>

где имя - это символическое обозначение операнда в контексте инструкции;

тип доступа - это символ, указывающий тип доступа к операнду:

A - вычисляется действительный адрес указанного операнда;

B - ссылки на операнд нет. Спецификация операнда является смещением перехода. Размер спецификации перехода определяется с помощью описателя <тип данных>;

M - операнд модифицируется. Возможны операции чтения/записи;

R - только чтение;

V - вычисляется действительный адрес операнда. Если это адрес в памяти, он указывает действительный операнд инструкции. Если спецификация указывает на регистр "RN", тогда используется R[N] или R[N+1]'R[N], т.е. конкатенация содержимого двух регистров;

W - только запись.

Тип данных - это символ, указывающий тип данных операнда:

B - байт;

D - D_формат;

F - F_формат;

G - G_формат;

H - H_формат;

L - длинное слово;

O - октаслово;

Q - квадрослово;

V - битовое поле;

W - слово;

X - первый тип данных, указанный инструкцией;

Y - второй тип данных, указанный инструкцией;

* - последовательность длинных слов (ис-

пользуется только в указанных инструкциях).

При описании инструкций используются следующие мнемонические обозначения:

ADD

- слагаемое;

ADDR

- адрес;

ARGLIST

- список аргументов;

BASE

- базовый адрес;

CHAR

- символ;

CNT

- счетчик;

DIF

- разность;

DISPL

- смещение;

DIVD

- делимое;

DIVR

- делитель;

DST

- адес приемника;

ENTRY

- точка входа;

ESC

- потеря, переход;

FILL

- заполнитель;

FINDPOS

- находить позицию;

FRACT

- мантисса;

INDEX

- индекс;

INICRE

- начальный контроль по циклической избыточности;

INT

- целое;

LEN

- длина;

LIMIT

- лимит;

MASK

- маска;

MIN

- минимальный;

MULD

- множитель;

MULR

- множимое;

MULRX

- расширение множителя;

NUMARG

- число (количество) аргументов;

OPTION

- выбор (необязательный), дополнительный;

PARAM

- параметр;

POS

- позиция;

PRED

- предшественник;

PROCREG

- внутренний регистр процессора;

PROD

- произведение;

QUO

- частное;

REM

- остаток;

SELECTOR

- переключатель;

SIZE

- размер;

SRE

- источник;

STARTPOS

- начальная позиция;

STREAM

- поток;

STRLEN

- длина строки;

SUB

- вычитаемое;

SUM

- сумма;

TBL

- таблица.

2. Базовая архитектура

2.1. Адресация

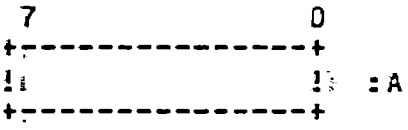
Основной адресуемой единицей в СМ 1700 является 8-разрядный байт. Длина виртуального адреса равна 32 разрядам. Таким образом, об'ем виртуального адресного пространства равен 2 в 32-ой степени (приблизительно 4300 мбайт). Виртуальные адреса, формируемые программой, преобразуются в адреса физической памяти с помощью механизма диспетчеризации памяти.

2.2. Типы данных

2.2.1. Байт

Байт представляет собой 8 непрерывно расположенных разрядов, начиная с адресуемой границы байта. Разряды нумеруются справа налево от 0 до 7.

Формат



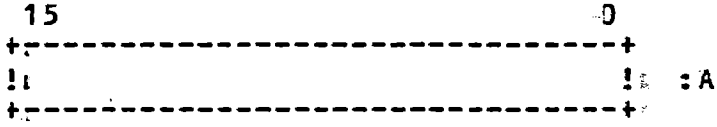
Байт определяется адресом А. При арифметической интерпретации байт представляет собой целое в дополнительном коде с разрядами, значение которых возрастает от 0-го до 6-го разряда и с 7-ым разрядом в качестве знакового.

Значение целого числа изменяется в диапазоне от -128 до 127. При выполнении операций сложения, вычитания, сравнения допускается интерпретация байта как числа без знака с разрядами, значения которых возрастают от 0-го до 7-го разряда. Значение целого числа без знака изменяется в диапазоне от 0 до 255.

2.2.2. Слово

Слово представляет собой два последовательно расположенных байта, начиная с произвольной границы байта. Разряды нумеруются справа налево от 0 до 15.

Формат

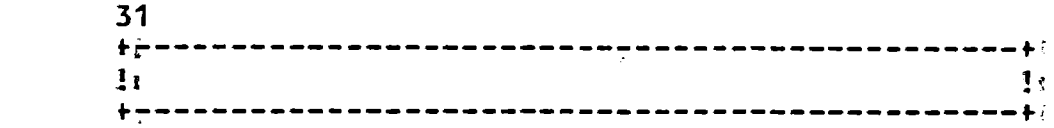


Слово определяется адресом A, то есть адресом младшего байта. При арифметической интерпретации слово представляет собой целое в дополнительном коде с разрядами, значение которых возрастает от 0-го разряда до 14-го разряда с 15-ым разрядом в качестве знакового. Значение целого числа изменяется в диапазоне от -32768 до 32767. При выполнении операций сложения, вычитания, сравнения допускается интерпретация слова как числа без знака с разрядами, значение которых возрастает от 0 до 65535.

2.2.3. Длинное слово

Длинное слово представляет собой четыре последовательно расположенных байта. Разряды нумеруются справа налево от 0 до 31.

Формат



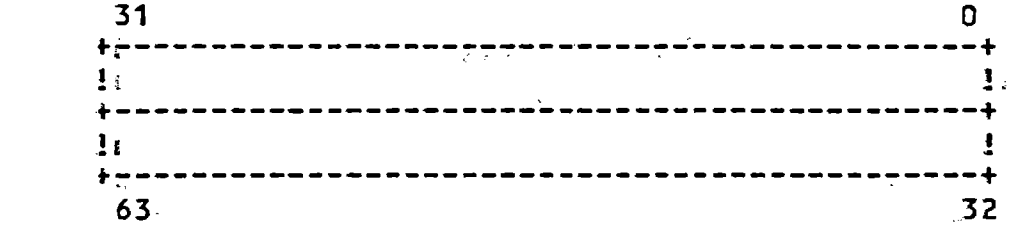
Длинное слово определяется адресом A, то есть адресом младшего байта. При арифметической интерпретации длинное слово представляет собой целое в дополнительном коде с разрядами, значение которых возрастает от 0-го до 30-го разряда, с 31-ым разрядом в качестве знакового. Значение целого числа изменяется в диапазоне от -2147483648 до 2147483647. При выполнении операций сложения, вычитания, сравнения допускается интерпретация длинного слова как числа без знака с разрядами, значение которых возрастает от 0-го до 31-го разряда. Значение целого числа без знака изменяется в диапазоне от 0 до 4294967295.

Отметим, что формат длинного слова отличается от формата, принятого в процессоре с плавающей запятой для ранних моделей см эвм, в котором значение разрядов возрастает от 16-го по 31-ый и от 0-го по 14-ый, и разряд 15-ый используется как знаковый.

2.2.4. Квадрослово

Квадрослово состоит из 8-ми последовательно расположенных байтов, начиная с произвольной границы байта. Разряды нумеруются справа налево от 0 до 63.

Формат

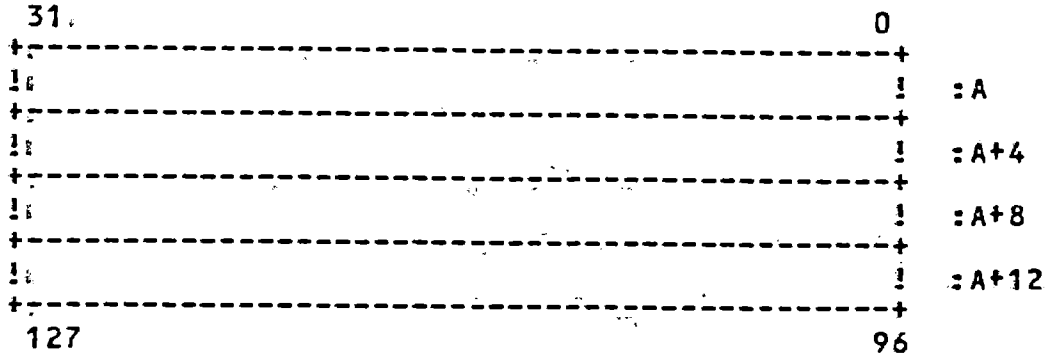


Квадрослово определяется адресом А, то есть адресом младшего байта. При арифметической интерпретации квадрослово представляет собой целое в дополнительном коде с разрядами, значение которых возрастает от 0-го до 62-го и с 63-им разрядом в качестве знакового. Значение целого числа изменяется в диапазоне от -2 в 63 степени до 2 в 63 степени минус 1. Этот формат может быть использован не во всех инструкциях СМ 1700.

2.2.5. Октаслово

Октаслово состоит из 16-ти последовательно расположенных байтов, начиная с произвольной границы байта. Разряды нумеруются справа налево от 0 до 127.

Формат

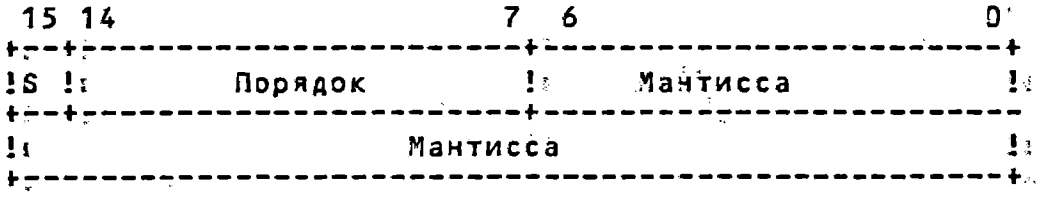


Октаслово определяется адресом А, то есть адресом младшего байта. При арифметической интерпретации октаслово представляет собой целое в дополнительном коде с разрядами, значение которых возрастает от 0-го до 126-го разряда и с 127-ым разрядом в качестве знакового. Значение целого числа изменяется в диапазоне от -2 в степени 127 до 2 в степени 127 минус 1. Этот формат может быть использован не во всех инструкциях СМ 1700.

2.2.6. Число с плавающей запятой F_формата

Число с плавающей запятой F_формата состоит из 4 последовательно расположенных байтов. Разряды нумеруются справа от 0 до 31.

Формат



Число с плавающей запятой F_формата определяется адресом А, то есть адресом младшего байта и отображает число с

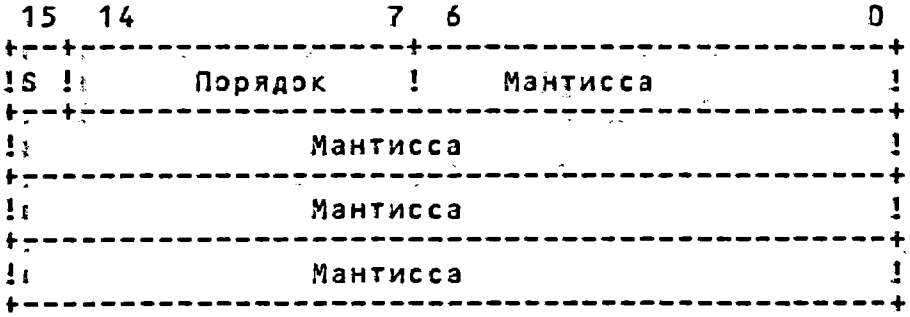
15-ым разрядом в качестве знакового, порядком в разрядах с 14 до 7 и нормализованной 24-разрядной мантиссой в разрядах с 6-го по 0-ой и с 31-го по 16-ый. Старший разряд мантиссы в слове не отображается. В мантиссе значение разрядов возрастает от 16-го до 31-го и от 0-го до 6-го. 8-ми разрядное поле порядка может содержать значение от 0 до 255. Число, равное нулю, отображается нулями в поле порядка и в знаковом разряде. Значение от 1 до 255 в поле порядка отображает двоичные порядки от -127 до 127. Комбинация нулевого порядка и единицы в знаковом разряде является резервной. При обработке командой такой комбинации возникает ошибка резервного операнда. Точность 7 десятичных цифр.

Значение числа с плавающей запятой F_формата изменяется в диапазоне приблизительно от 0.29*10**(-38) до 1.7*10**38.

2.2.7. Число с плавающей запятой D_формата

Число с плавающей запятой D_формата состоит из 8-ми последовательно расположенных байтов. Разряды нумеруются справа от 0 до 63.

Формат

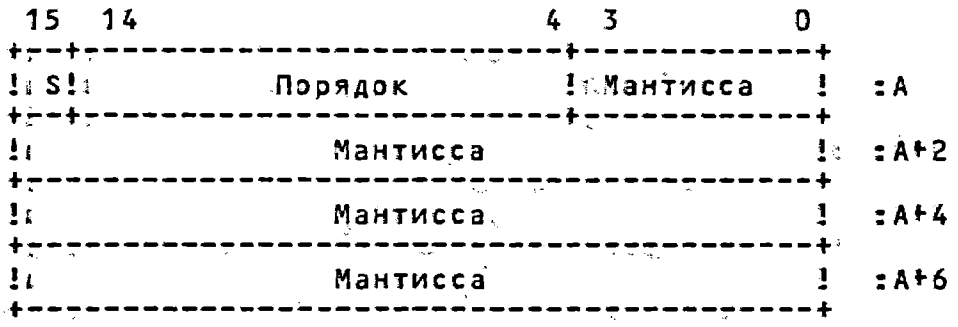


Число D_формата определяется адресом A, то есть адресом младшего байта. Этот формат аналогичен формату числа с плавающей запятой F_формата, за исключением дополнительных наименее значащих 32 разрядов мантиссы. В мантиссе разряды увеличивают свое значение от 48-го до 63-го, от 32-го до 47-го, от 16-го по 31-ый и от 0-го до 6-го разряда. Представление порядка то же самое, что и для числа с плавающей запятой F_формата. Точность 16 десятичных цифр.

2.2.8. Число с плавающей запятой G_формата

Число с плавающей запятой G_формата состоит из 8-ми последовательно расположенных байтов. Разряды нумеруются справа налево от 0 до 63.

Формат



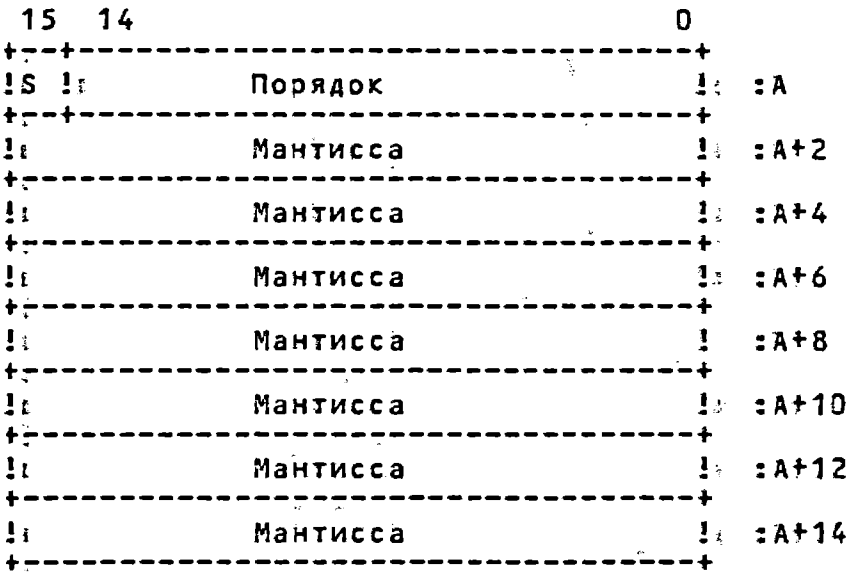
Число G_формата определяется адресом A, то есть адресом младшего байта. В этом формате представляется число со знаком в 15-ом разряде, порядком в разрядах с 14 по 4 и нормализованной 53-разрядной мантиссой в разрядах с 3-го по 0-ой и с 63-го по 16-ый. Старший разряд мантиссы не отображается. Внутри мантиссы значения разрядов возрастают от 48-го до 63-го, от 32-го до 47-го, от 16-го до 31-го и от

0-го по 3-ий. 11-ти разрядное поле порядка может содержать значения от 0 до 2047. Нулевое значение числа типа G_формата отображается нулями в поле порядка и нулем в знаковом разряде. Значения от 1 до 2047 в поле порядка соответствуют допустимым значениям двоичного порядка от -1023 до +1023. Комбинация нулевого порядка с единицей в знаковом разряде является резервной. При обработке командой такой комбинации возникает ошибка резервного операнда. Значение числа с плавающей запятой G_формата изменяется в диапазоне от 0.56*10**(-308) до 0.9*10**308. Точность - 15 десятичных цифр.

2.2.9. Число с плавающей запятой H_формата

Число с плавающей запятой H_формата состоит из 16-ти последовательно расположенных байтов. Разряды нумеруются справа налево от 0 до 127.

Формат



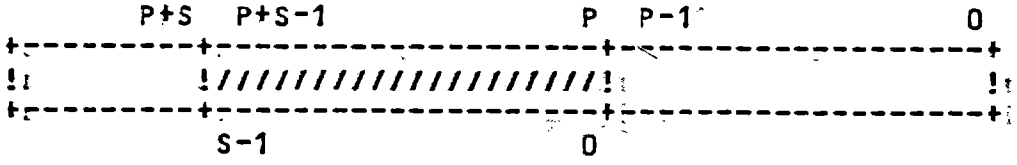
Число H_формата определяется адресом A, то есть адресом младшего байта. В этом формате представляется число со знаком в 15-ом разряде, порядком в разрядах с 14 по 3-ой и нормализованной 113-разрядной мантиссой в разрядах от 127-го по 16-ый. Старший разряд мантиссы не отображается. Внутри мантиссы значения разрядов возрастают от 112-го до 127-го, от 96-го до 111-го, от 80-го до 95-го, от 64-го до 79-го, от 48-го до 63-го, от 32-го до 47-го, от 16-го до 31-го; 15-разрядное поле порядка может содержать значения от 0 до 32767. Нулевое значение числа H_формата отображается нулевым значением поля порядка и нулем в знаковом разряде. Значения от 1 до 32767 соответствуют допустимым значениям двоичного порядка от -16383 до +16383. Комбинация нулевого порядка с единицей в знаковом разряде является резервной. При обработке командой такой комбинации возникает ошибка резервного операнда. Значение числа с плавающей запятой H_формата изменяется в диапазоне приблизительно от 0.84*10**(-4932) до 0.59*10**4932. Точность 33 десятичных цифр.

2.2.10. Битовое поле переменной длины

Битовое поле переменной длины состоит из некоторого числа (от 0 до 32) последовательно расположенных разрядов, начиная с произвольного места относительно границы байта. Поле определяется тремя атрибутами: адресом байта, разрядной позицией начала поля p относительно разряда 0 байта с адресом A и размером поля S. Спецификация битового поля

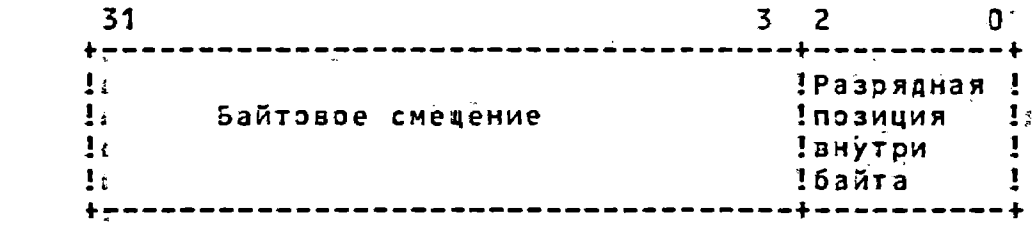
переменной длины выделена заштрихованной областью.

Формат



Для представления битовых полей переменной длины номер позиции может изменяться в диапазоне от (-2) в степени 31 до 2 в степени 31 минус 1 и может быть представлен в виде 29-разрядного байтового смещения со знаком и 3-разрядного поля позиции внутри байта.

Формат:



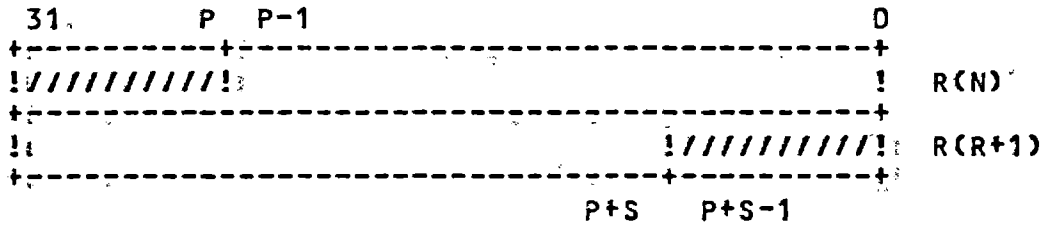
Байтовое смещение складывается с адресом А, формируя, таким образом, адрес байта, с которого начинается поле. В 3-х разрядном поле позиции внутри байта закодирована начальная позиция (от 0 до 7) поля внутри байта. В инструкциях см 1700 обеспечивается непосредственная интерпретация содержимого поля как целого числа со знаком или без знака. При знаковой интерпретации число представляется в дополнительном коде с разрядами, значение которых возрастает от 0 до S-2, а разряд S-1 является знаковым. При беззнаковой интерпретации значение разрядов возрастает от 0 до S-1. Поле нулевого размера имеет значение, равное нулю.

Битовые поля переменной длины могут занимать от 1 до 5

байтов. С точки зрения диспетчера памяти фактическое обращение производится только к минимальному числу байтов, необходимому для отображения поля.

При отображении битовых полей в регистрах позиция изменяется от 0 до 31. Операнд позиции определяет начальную позицию (от 0 до 31) поля в регистре. Битовое поле с переменным числом разрядов может содержаться в двух регистрах, если сумма значений позиции и размера превышает 32.

Формат

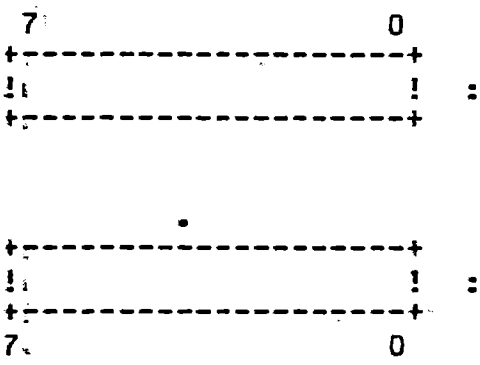


Подробно битовые поля переменной длины описаны в разделе 4.

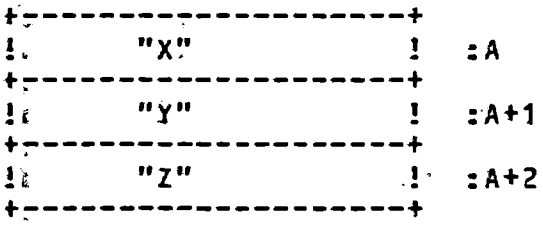
2.2.11. Символьная строка

Символьная строка представляет собой непрерывную последовательность байтов в памяти. Символьная строка определяется двумя атрибутами: адресом первого байта строки и длиной строки L в байтах.

Формат



Адрес строки определяет первый символ строки. Символьная строка "XYZ" в памяти отображается следующим образом:



Длина строки L меняется в диапазоне от 0 до 65535.

2.2.12. Числовая строка

Числовая строка представляет собой непрерывную последовательность байтов в памяти, которая представляет число как строку десятичных цифр. Строка определяется 2-мя атрибутами: адресом первого байта строки и длиной строки L в байтах.

Все байты в числовой строке, за исключением первого, должны содержать символы десятичных цифр (от 0 до 9) в коде КОИ-8. Представление цифр имеет вид, указанный в табл. 1.

Таблица 1

Цифра	Представление		Символ в коде КОИ-8
	Десятичное	Шестнадцатеричное	
0	48	30	0
1	49	31	1
2	50	32	2
3	51	33	3
4	52	34	4
5	53	35	5
6	54	36	6
7	55	37	7
8	56	38	8
9	57	39	9

В старшем адресуемом байте числовой строки одновременно кодируются наименее значащая цифра и знак числовой строки. В инструкциях обработки числовых строк используются несколько типов кодирования:

- при представлении чисел без знака младшая значащая цифра содержит символ цифры в коде КОИ-8;
- зонное кодирование;
- перфо-кодирование.

Допустимое отображение наименее значащей цифры и знака в каждом из последних двух форматов имеет вид, представленный в табл. 2.

Таблица 2

Цифра	Зонный формат		Перфо-формат	
	Представление			
	Десятичное	Шестнадцатеричное	Десятичное	Шестнадцатеричное
0	48	30	123	7B
1	49	31	65	41
2	50	32	66	42
3	51	33	67	43
4	52	34	68	44
5	53	35	69	45
6	54	36	70	46
7	55	37	71	47
8	56	38	72	48
9	57	39	73	49
-0	112	70	125	7D
-1	113	71	126	4A
-2	114	72	127	AB
-3	115	73	76	4C
-4	116	74	77	4D
-5	117	75	78	4E
-6	118	76	79	4F
-7	119	77	80	50

Продолжение табл. 2

Цифра	Зонный формат		Перфо-формат	
	Представление			
	Десятичное	Шестнадцатеричное	Десятичное	Шестнадцатеричное
-8	120	78	81	51
-9	121	79	82	52

Длина L числовой строки должна находиться в диапазоне от 0 до 31 (от 0 до 31 цифры). Значение строки с нулевой длиной идентично нулю.

Адрес строки определяет тот байт строки, который содержит первую значащую цифру.

Число "-123" представляется следующим образом:

Зонный формат

	7 4	3 0	
"1"	3	1	:A
"2"	3	2	:A+1
"-3"	7	3	:A+2

Перфо-формат

	7 4	3 0	
"1"	3	1	:A
"2"	3	2	:A+1
"-3"	4	C	:A+2

2.2.13. Числовая строка с выделенным ведущим знаком

Числовая строка с выделенным ведущим знаком представляет собой последовательность непрерывно расположенных бай-

тов в памяти, первый байт которой содержит знак числа. Числовая строка с выделенным знаком определяется двумя атрибутами: адресом первого байта (содержащего символ знака) и длиной L, которая определяет число цифр, а не число байтов. Число байтов в строке равно L+1.

Знак в строке запоминается в отдельном байте. Допустимые значения в этом байте указаны в табл. 3.

Таблица 3

Знак	Представление		Символ в коде КОИ-8
	Десятичное	Шестнадцатеричное	
+	43	2B	+
+	32	20	пробел
-	45	2D	-

Предпочтительным представлением для знака "+" является его код в КОИ-8. Все последующие байты содержат коды цифр в коде КОИ-8.

Длина L строки должна изменяться в диапазоне от 0 до 31 (от 0 до 31 цифры). Значение строки с нулевой длиной идентично нулю.

Адрес строки определяет тот байт строки, который содержит знак. Таким образом, число "+123" представляется следующим образом:

```
     7      4  3      0
    +-------+--------+
"+" !   2   !    8   !  :A
    +-------+--------+
"1" !   3   !    1   !  :A+1
    +-------+--------+
"2" !   3   !    2   !  :A+2
    +-------+--------+
"3" !   3   !    3   !  :A+3
    +-------+--------+
```

число "-123":

```
     7      4  3      0
    +-------+--------+
"-" !   2   !    D   !  :A
    +-------+--------+
"1" !   3   !    1   !  :A+1
    +-------+--------+
"2" !   3   !    2   !  :A+2
    +-------+--------+
"3" !   3   !    3   !  :A+3
    +-------+--------+
```

2.2.14. Упакованная десятичная строка

Упакованная десятичная строка представляет собой непрерывную последовательность байтов в памяти. Упакованная десятичная строка определяется двумя атрибутами: адресом первого байта строки длиной L, которая определяет число цифр, а не число байтов. Байты в упакованной десятичной строке делятся на два 4-разрядных поля (полубайта), каждое из которых должно содержать десятичную цифру, за исключением младшего полубайта (разряды 3:0) последнего (с наибольшим адресом) байта, который должен содержать знак. Отображение цифр и знака представлено в табл. 4.

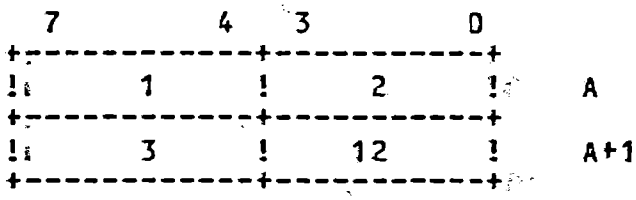
Таблица 4

Цифра или знак	Представление	
	Десятичное	Шестнадцатеричное
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
+	10, 12, 14 или 15	A, C, E или F
-	11 или 13	B или D

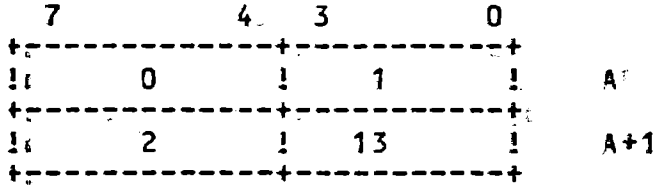
Предпочтительным представлением знака является 12 для "+" и 13 для "-". Длина L представляет собой число цифр в упакованной десятичной строке (не считая знака), которое может изменяться в диапазоне от 0 до 31. Если число цифр нечетное, то цифры и знак помещаются в L/2 (только целая часть)+1 байтах. Если число цифр четное, то требуется наличие дополнительной цифры "0" в старшем полубайте (разряды 7:4) первого байта строки. Длина строки также будет равна L/2+1.

Адрес строки определяет тот байт в строке, который

содержит значащую цифру в своем старшем полубайте. Таким образом, число "+123" имеет длину 3 и представляется следующим образом:



А число "-12" имеет длину 2 и представляется в виде:



2.3. Состояние процессора

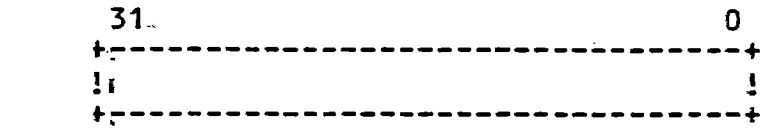
Состояние процессора представляет собой ту часть состояния процесса, которая во время его выполнения запоминается в регистрах процессора. Состояние процессора, описываемое в этом подразделе, доступно непривилегированному программному обеспечению. Некоторые дополнительные элементы состояния процессора описываются в разделах 2, 3.

К непривилегированным элементам состояния процессора относятся шестнадцать 32-разрядных регистров общего назначения, которые обозначаются RN, где N изменяется в диапазоне от 0 до 15, а также 16-ти разрядное слово состояния процессора (PSW). В данном документе принято, что если имеется неоднозначность обозначений регистров (например, N является арифметическим выражением), то для обозначения регистров используются также R[N]. Регистры общего назначения исполь-

зуются для временного хранения данных, а также в качестве аккумуляторов, индексных регистров и регистров базы.

Разряды в регистре нумеруются справа налево от 0 до 31.

Формат:



Некоторые из регистров имеют специальное назначение:

- R15 является счетчиком инструкций (PC). Счетчик инструкций содержит адрес следующего байта инструкции программы;
- R14 является указателем стека (SP). Указатель стека содержит адрес верхушки процессорного стека;
- R13 является указателем текущего кадра (FP). В соответствии с правилом вызова процедур создается структура данных в стеке, которая называется кадром стека. Указатель кадра стека содержит базовый адрес этой структуры данных;
- R12 является указателем списка аргументов (AP). При выполнении вызова процедуры используется структура данных, которая называется списком аргументов. Регистр ap содержит базовый адрес этой структуры данных.

Все эти регистры используются как указатели базового адреса. Специальное назначение этих регистров, в общем случае, не исключает их использование для других целей. Однако, как это будет показано в разделе 3, счетчик инструкций

(PC), не может быть использован в качестве аккумулятора, индексного регистра и для временного запоминания данных.

При запоминании в регистре байта, слова, длинного слова или числа с плавающей запятой F_формата нумерация разрядов в этом регистре соответствует нумерации в памяти. Следовательно, байт в регистре запоминается в разрядах 7:0, слово - в разрядах 15:0, длинное слово или число F_формата - в разрядах 31:0. При записи байта или слова в регистр записываются только разряды 7:0 или 15:0 соответственно, остальные разряды в регистре остаются без изменения. При чтении байта или слова из регистра считываются только разряды 7:0 или 15:0 соответственно, остальные разряды игнорируются.

Для запоминания в регистре R[N] квадрослова или числа с плавающей запятой D_формата или G_формата эти элементы фактически запоминаются в связке из двух последовательных регистров R[N] и R[N+1]. Из-за ограничений, накладываемых на использование счетчика инструкций PC, при обращении к связке PC-R0 результат будет непредсказуемым. Разряды 31:0 элемента данных запоминаются в разрядах 31:0 регистра R[N], а разряды 63:32 элемента данных запоминаются в разрядах 31:0 регистра R[N+1].

При запоминании в регистре R[N] октаслова или числа с плавающей запятой H_формата эти элементы фактически запоминаются в связке из четырех последовательных регистров R[N], R[N+1], R[N+2] и R[N+3]. Из-за ограничений, накладываемых на использование счетчика инструкций PC, при обращении к

связке, в которую входят PC и RO, результат будет непредсказуемым.

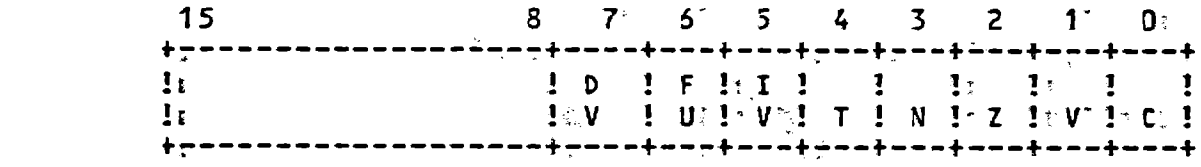
Разряды 31:0 элемента данных запоминаются в разрядах 31:0 регистра R[N], разряды 63:32 элемента данных запоминаются в разрядах 31:0 регистра R[N+1], разряды 95:64 элемента данных запоминаются в разрядах 31:0 регистра R[N+2], разряды 127:96 элемента данных запоминаются в разрядах 31:0 регистра R[N+3]. В регистрах можно задавать и битовые поля переменной длины, учитывая одно ограничение: начальная разрядная позиция p должна находиться в диапазоне от 0 до 31. Также, как и для квадрослова из числа D_формата или G_формата, пара регистров R[N] и R[N+1] интерпретируется как 64-разрядный регистр с разрядами 31:0 в регистре R[N] и с разрядами 63:32 в регистре R[N+1].

Инструкции СМ 1700 не могут обрабатывать данные, представленные в виде одного из типов символьной строки и расположенных в регистре. Таким образом, не имеется архитектурных требований к представлению символьных строк в регистрах.

2.4. Слово состояния процессора - PSW

Слово состояния процессора содержит коды условий, которые отображают информацию о результатах выполнения предыдущей инструкции, а также разряды разрешения прерывания в исключительных ситуациях, которые управляют действиями процессора при возникновении таких ситуаций (см. документ [1]).

Формат слова состояния процессора



Состояние кодов условий является непредсказуемым, если они устанавливаются по непредсказуемым результатам. Выполненные инструкции вызова процедуры (см. раздел 4) устанавливают разряды IV и DV, сбрасывают разряд FU и оставляют без изменения разряд T.

2.4.1. Разряд C

Если установлен в 1 разряд C (перенос), то это означает, что в результате выполнения последней инструкции, которая влияет на состояние разряда C, имел место перенос или заимствование из старшего разряда результата. Если разряд сброшен, то это означает, что не было ни переноса, ни заимствования.

2.4.2. Разряд V

Если установлен в 1 разряд V (переполнение), это означает, что в результате выполнения последней инструкции, которая влияет на состояние разряда V, было получено значение, которое выходит за пределы диапазона допустимого для данного операнда. Разряд V устанавливается, если имела место ошибка преобразования. Если разряд V сброшен, это означает, что не было ни переполнения, ни ошибки преобразова-

ния.

2.4.3. Разряд Z

Если установлен в 1 разряд Z (нуль), то это означает, что в результате выполнения последней инструкции, которая влияет на состояние разряда Z, был получен нулевой результат. Если разряд Z сброшен, то результат был ненулевым.

2.4.4. Разряд N

Если установлен в 1 разряд N, то это означает, что в результате выполнения последней инструкции, которая влияет на состояние разряда N, был получен отрицательный результат. Если этот разряд сброшен, то результат был положительным (или нулевым).

2.4.5. Разряд T

Если перед выполением инструкции установлен в 1 разряд T (слежение), то устанавливается разряд TP в длинном слове состояния процессора. Если в конце инструкции установлен разряд TP, то перед выполнением следующей инструкции происходит внутреннее прерывание. Подробная информация приводится в документе [1].

2.4.6. Разряд IV

Если при выполнении инструкции произошло целочисленное переполнение или ошибка преобразования и если установлен в 1 разряд IV (целочисленное переполнение), то происходит внутреннее прерывание по целочисленному переполнению. Если разряд IV сброшен, то прерывания не происходит (однако, при этом разряд V останется установленным).

2.4.7. Разряд FU

Если в результате выполнения инструкций с плавающей запятой получилось слишком малое число, которое не может быть представлено в результате, и если установлен в 1 разряд FU, то происходит ошибка переполнения по нижней границе диапазона. Если разряд FU сброшен, то эта ошибка не происходит.

2.4.8. Разряд DV

Если в результате выполения инструкции (которая работает с числовой или упакованной числовой строкой), происходит десятичное переполнение или ошибка преобразования и, если установлен в 1 разряд DV, то происходит внутреннее прерывание по десятичному переполнению. Если разряд DV сброшен, то это прерывание не происходит, хотя разряд V остается установленным.

2.5. Исключительные ситуации, по которым постоянно разрешено прерывание

Работа процессора при возникновении некоторых исключительных ситуаций не управляется разрядами в слове состояния процессора. В этих исключительных ситуациях всегда происходит внутреннее прерывание или ошибка.

2.5.1. Деление на нуль

Внутреннее прерывание при делении на нуль возникает в результате выполнения инструкции, работающей с целыми или десятичными числами и имеющей в качестве операнда нулевой делитель. При выполнении инструкции деления числа с плавающей запятой на нулевой делитель возникает ошибка.

2.5.2. Переполнение при работе с числами с плавающей запятой

Ситуация переполнения при работе с числами с плавающей запятой возникает после выполнения инструкций с плавающей запятой, которая формирует результат, выходящий за верхний предел диапазона представления числа с плавающей запятой.

2.6. Формат инструкции

В СМ 1700 используется формат инструкции переменной длины. В инструкциях определяется собственно операция и от 0 до 6 операндов. Спецификатор операции определяет код опе-

рации. В зависимости от инструкции код операции состоит из 1 или 2 байтов. Спецификатор операнда определяет режим адресации, который используется для выборки операнда. Спецификатор операнда может состоять из 1 или 2 байтов. Спецификатор операнда может быть дополнен расширением спецификатора, адресом или непосредственными данными. Формат инструкции имеет вид:

- код операции;
- спецификатор 1-го операнда;
- расширение спецификатора, адрес или непосредственные данные (если необходимо);
- спецификатор 2-го операнда;

- спецификатор N-ого операнда;
- расширение спецификатора, адрес или непосредственные данные (если необходимо).

Полное описание режимов адресации (см. раздел 3). Описание инструкций (раздел 4).

2.7. Разделение процедур и данных

Архитектура СМ 1700 обеспечивает (и предусматривает специальный механизм) разделение процедуры (инструкций) и записываемых данных. Процедуры не могут использовать сформированные данные, которые должны выполняться как инструкции, если не выполнена команда REI (см. документ [1]) или

переключение контекста (см. документ [1]). Если процедура сформировала данные, предназначенные для выполнения в качестве инструкций, и если не встретилась команда REI или переключатель контекста, то результат выполнения будет непредсказуемым.

2.8. Структура ввода-вывода

В целом структура ввода-вывода СМ 1700 близка к структуре ввода-вывода для 16-разрядных моделей СМ ЭВМ. Контроллер устройства ввода-вывода определяется набором регистров. Регистрам присвоены адреса в физическом адресном пространстве. Команды, посылаемые процессором в контроллеры ввода-вывода, записываются в эти регистры. Контроллеры записывают в эти регистры свое состояние, а процессор последовательно считывает их. Так как регистры имеют адреса, такие же как и ячейки памяти, то запись и чтение этих регистров можно производить с помощью обычных инструкций. Таким образом, отпадает необходимость в специальных инструкциях ввода-вывода. Доступом к регистрам контроллеров внешних устройств управляет обычный механизм диспетчеризации памяти.

2.9. Структура прерываний

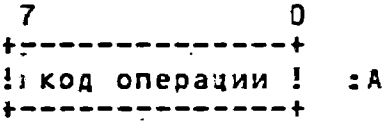
В процессоре СМ 1700 предусмотрена 32-уровневая приоритетная структура прерываний, которая детально описана в документе [1].

3. Форматы инструкций и режимы адресации.

3.1. Форматы кода операции

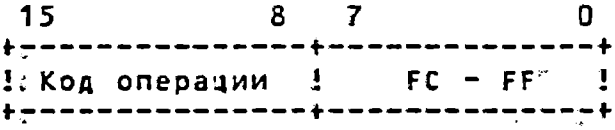
Инструкция определяется адресом байта A, в котором содержится код операции.

Формат:



Код операции может занимать 2 байта. Длина зависит от содержимого байта по адресу A. Длина кода операции составляет 2 байта, когда значение первого байта находится в пределах от FC (шестнадцатеричное) по FF (шестнадцатеричное).

Формат



3.2. Спецификаторы операнда

Каждая инструкция содержит некоторую последовательность типов спецификаторов операнда. Тип операнда состоит из двух компонентов: типа доступа и типа данных.

К типам доступа относятся:

1) чтение - указанный операнд предназначен только для чтения;

2) запись - указанный операнд предназначен только для записи;

3) модификация - указанный операнд считывается и может быть модифицирован записью результата по этому же адресу. Этот тип не может быть использован при блокировке памяти;

4) адрес - фактическим операндом инструкции является адрес указанного операнда, представленный в виде длинного слова. К указанному операнду нет непосредственного доступа, хотя инструкции могут последовательно использовать адрес для доступа к этому операнду;

5) базовый адрес битового поля переменной длины - то же самое, что и обычный адресный тип доступа, за исключением регистрового режима. В регистровом режиме поле, которое находится в регистре N, определяется спецификатором операнда (или регистром N+1, связанным с регистром N). Этот тип доступа является специальным случаем адресного типа доступа;

6) переход - к операнду доступа не производится. Спецификатор операнда представляет собой смещение для перехода.

Типы доступа с 1 по 5 соответствуют общим режимам адресации и описаны в подразделе 3.4. Тип 6 относится к режиму адресации перехода и описан в подразделе 3.6.

К типам данных относятся:

1) байт;

2) слово;

3) длинное слово и число с плавающей запятой F_формата, которые эквивалентны с точки зрения режима адресации;

4) квадрослово и число с плавающей запятой D_формата или G_формата, которые также эквивалентны с точки зрения режима адресации;

5) октаслово или число с плавающей запятой H_формата, которые также эквивалентны с точки зрения режима адресации.

Для доступа типа адреса и перехода, при которых нет непосредственного обращения к операнду, тип данных имеет следующий смысл:

1) адрес - длина операнда; используется при вычислении адреса в режимах автоувеличения, автоуменьшения и индексных режимах;

2) переход - размер смещения перехода.

3.3. Обозначения

При описании режимов адресации используются следующие обозначения:

+

- сложение;

-

- вычитание;

*

- умножение;

<-

- пересылка;

=

- равенство;

- конкатенация;

RN или R[N]

- содержимое регистра N;

PC или SP

- содержимое регистров 15 или 14 соответственно;

Примечание. При формальном описании режимов адресации обозначения, например, RN или PC всегда означают содержимое регистра RN или регистра 15. Если не возникает неоднозначности, то обозначения, например, RN или PC в тексте часто используются в качестве названий регистра RN или регистра 15.

(X)

- содержимое ячейки памяти с адресом X;

( )

- арифметические скобки, которые используются для индикации предшествования;

SEXT (X)

- знак операнда X расширяется до требуемого размера операнда;

ZEXT (X)

- операнд X дополняется ведущими нулями до требуемого размера операнда;

OA

- адрес операнда;

!

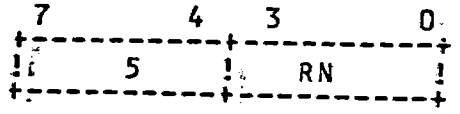
- разделитель комментария.

Описание каждого режима адресации включает определение адреса операнда и самого операнда. Для спецификаторов операндов с адресным типом доступа адрес операнда является фактическим операндом инструкции. Для других типов доступа специфицируемый операнд является в действительности операндом инструкции. В описание операции переходов включается определение адреса перехода.

3.4. Форматы общих режимов адресации

3.4.1. Регистровый режим

Формат спецификатора операнда



Расширение спецификатора не используется.

В регистровом режиме адресации операндом является содержимое регистра N (или содержимое регистров N и N+1 для некоторых других типов операндов).

Операнд = RN ! если один регистр

или

операнд = R[N+1]'RN ! если два регистра

или

операнд = R[N+3]'R[N+2]'R[N+1]'RN

! если четыре регистра

Так как регистры не имеют адресов в памяти, то адрес операнда не определяется и режим не может быть использован

для спецификации операндов с адресным типом доступа (исключение составляет случай использования базового адреса в инструкциях, работающих с полями с переменным числом разрядов (см. раздел 4). В противном случае возникает ошибка неверного режима адресации.

Счетчик инструкций может быть использован в регистровом режиме адресации. Если производится чтение счетчика инструкций, то результат будет непредсказуемым. Сходным образом, указатель стека не может быть использован в регистровом режиме адресации для операнда, состоящего из двух связанных регистров. В этом случае результат также будет непредсказуемым.

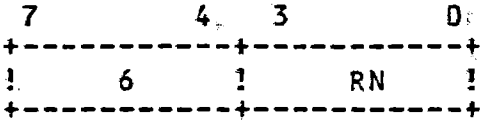
Если счетчик инструкций используется для записи операнда адресного типа в регистровом режиме с использованием двух связанных регистров, то содержимое регистра R0 будет непредсказуемым. Если регистры R12, R13, SP или PC используются в регистровом режиме для адресации операнда, состоящего из четырех связанных регистров, то результат будет непредсказуемым. Если счетчик инструкций используется в регистровом режиме для операнда с типом доступа "запись", состоящего из четырех связанных регистров, то содержимое R0, R1 и R2 будет непредсказуемым. Таким же образом, если регистр R13 используется в регистровом режиме для операнда с типом доступа "запись", состоящего из четырех связанных регистров, то содержимое R0 будет непредсказуемым, и если SP используется в регистровом режиме для операнда с типом доступа "запись", сотоящего из четырех связанных регистров,

то содержимое R0 и R1 будет непредсказуемым.

Обозначение регистрового режима на ассемблере: RN.

3.4.2. Косвенно-регистровый режим

Формат спецификатора операнда



Расширение спецификатора не используется.

В этом режиме адресом операнда является содержимое регистра N.

OA = RN

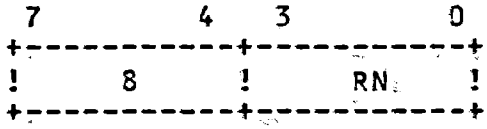
Операнд = (OA)

Счетчик инструкций в этом режиме не может быть использован. В противном случае адрес операнда и место, куда будет записан операнд (если он имеет тип модификации или записи), будут непредсказуемы.

Обозначение косвенно-регистрового режима на ассемблере: (RN).

3.4.3. Режим с автоувеличением

Формат спецификатора операнда



Расширение спецификатора не используется. Если в качестве регистра используется счетчик инструкций, то за

спецификатором следуют непосредственно данные, и режим называется непосредственным.

В этом режиме адресом операнда является содержимое регистра RN. После того, как вычислен адрес операнда, к содержимому регистра прибавляется размер операнда в байтах (1 - для байта; 2 - для слова; 4 - для длинного слова и числа с плавающей запятой F_формата; 8 - для квадрослова и числа с плавающей запятой G_формата и D_формата; 16 - для октаслова и числа с плавающей запятой H_формата):

OA = RN

RN <- RN + размер

Операнд = (OA)

При непосредственном режиме нельзя использовать операнды с доступом типа модификации или записи. Если непосредственный режим используется для операнда с доступом типа модификации, то считываемые данные будут непредсказуемыми. Если непосредственный режим используется для операнда с доступом типа модификации или записи, то адрес, по которому производится запись, будет непредсказуемым.

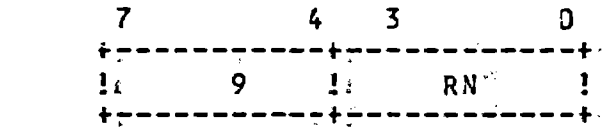
Обозначение режима с автоувеличением на ассемблере: (RN)+.

Обозначение для непосредственного режима: I^#константа

где константа - непосредственные данные, которые следуют за спецификатором.

3.4.4. Косвенный режим с автоувеличением

Формат спецификатора операнда



Расширение спецификатора не используется. Если в качестве регистра используется счетчик инструкций, то за спецификатором следует адрес в виде длинного слова и режим называется абсолютным режимом.

В косвенном режиме с автоувеличением, адрес операнда является содержимым длинного слова, адрес которого находится в регистре RN. После получения адреса к содержимому регистра N добавляется 4 (размер в байтах длинного слова адреса) и результат помещается в регистр.

ОА = (RN)

RN <- RN+4

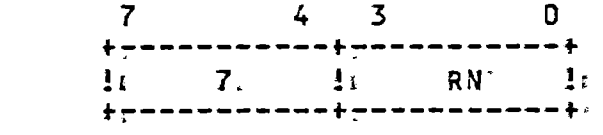
операнд = (ОА)

Обозначение косвенного режима с автоувеличением на ассемблере @(RN)+.

Обозначение для абсолютного режима: @#адрес.

3.4.5. Режим с автоуменьшением

Формат спецификатора операнда



Расширение спецификатора не используется.

В этом режиме из содержимого регистра вычитается размер операнда в байтах (1 - для байта; 2 - для слова; 4 - для длинного слова и числа с плавающей запятой F_формата; 8 - для квадрослова и чисел G_формата или D_формата; 16 - для октаслова и чисел H_формата) и модифицированное таким образом содержимое регистра является адресом операнда.

RN <- RN-размер

ОА = RN

Операнд = (ОА)

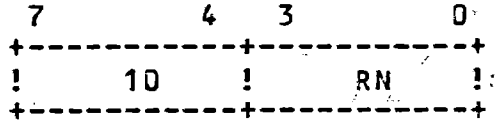
В режиме с автоувеличением нельзя использовать счетчик инструкций. В противном случае адрес операнда (или место, куда операнд записывается, если операнд имеет доступ типа модификации или записи) будет непредсказуемым и следующая выполняемая инструкция или следующий операнд будут непредсказуемыми.

Обозначение на ассемблере режима с автоуменьшением: -(RN).

3.4.6. Режим со смещением

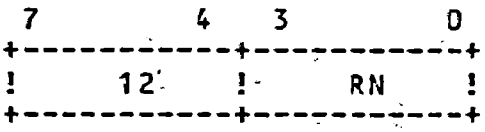
В режиме со смещением возможны 3 формата спецификатора операнда:

Формат 1



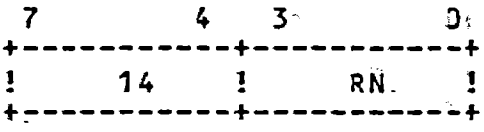
Расширение спецификатора представляет собой байт смещения со знаком, который следует за спецификатором. Этот вариант называется режимом со смещением в виде байта.

Формат 2



Расширение спецификатора представляет собой слово смещения со знаком, которое следует за спецификатором. Этот вариант называется режимом со смещением в виде слова.

Формат 3



Расширение спецификатора представляет собой длинное слово смещения со знаком, которое следует за спецификатором. Этот вариант называется режимом со смещением в виде длинного слова.

В этом режиме смещение (после расширения знакового разряда до 32-х разрядов, если смещение представляется в виде байта или слова) складывается с содержимым регистра, полученная сумма является адресом операнда:

ОА = RN + SEXT (смещение) если смещение в виде байта ; или слова

или

RN + смещение если смещение в виде L-слова

Операнд = (ОА)

Если в качестве регистра используется счетчик инструкций PC, то смещение складывается с модифицированным значе-

нием PC. Это значение является адресом байта, который непосредственно следует за расширением спецификатора.

Обозначение на ассемблере для режима смещения в виде байта, слова и длинного слова:

B^D(RN)

W^D(RN)

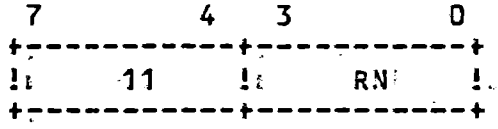
L^D(RN)

где D - смещение.

3.4.7. Косвенный режим со смещением

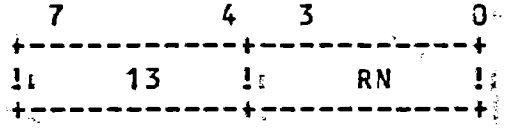
В косвенном режиме со смещением возможны три формата спецификатора операнда:

Формат 1



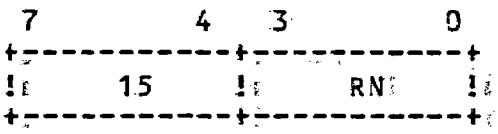
Расширение спецификатора представляет собой байт смещения со знаком, который следует за спецификатором операнда. Этот вариант называется косвенным режимом со смещением в виде байта.

Формат 2



Расширение спецификатора представляет собой слово смещения со знаком, которое следует за спецификатором операнда. Этот вариант называется косвенным режимом со смещением в виде слова.

Формат 3



Расширение спецификатора представляет собой длинное слово смещения со знаком, которое следует за спецификатором операнда. Этот вариант называется режимом со смещением в виде длинного слова.

В этом режиме адресации смещение (после расширения знака до 32-х разрядов, если смещение представляется в виде байта или слова) складывается с содержимым регистра; полученная сумма является адресом длинного слова, в котором содержится адрес операнда:

Oa = (RN + SEXT (смещение)) если смещение в виде байта или слова

или

(RN + смещение) если смещение в виде длинного слова

Операнд = (OA)

Если в качестве регистра используется счетчик инструкций PC, то смещение складывается с модифицированным значением PC. Это значение является адресом байта, который непосредственно следует за расширением спецификатора.

Соответственно обозначение на ассемблере для косвенного режима со смещением в виде байта, слова и длинного слова:

@B^D(RN)

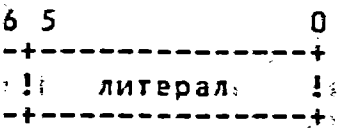
@W^D(RN)

@L^D(RN)

где D - смещение.

3.4.8. Литеральный режим.

Формат спецификатора операнда



Расширитель спецификатора не используется.

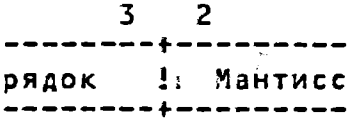
Для данных типа байта, слова, длинного слова, квадрослова и октаслова операнд представляет собой 6-разрядное поле операнда с расширением нулями в старших разрядах:

Операнд = ZEXT (литерал)

Таким образом, для этих типов данных в литеральном режиме могут быть представлены значения от 0 до 63.

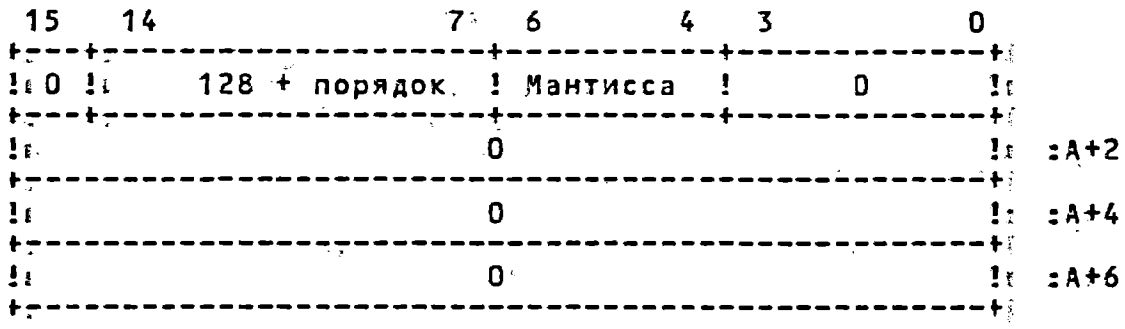
При использовании плавающих данных форматов F, G, D и H 6-разрядное поле литерала состоит из двух трехразрядных полей:

Формат



Поля порядка и мантиссы используются для формирования плавающих чисел типа F и D следующим образом:

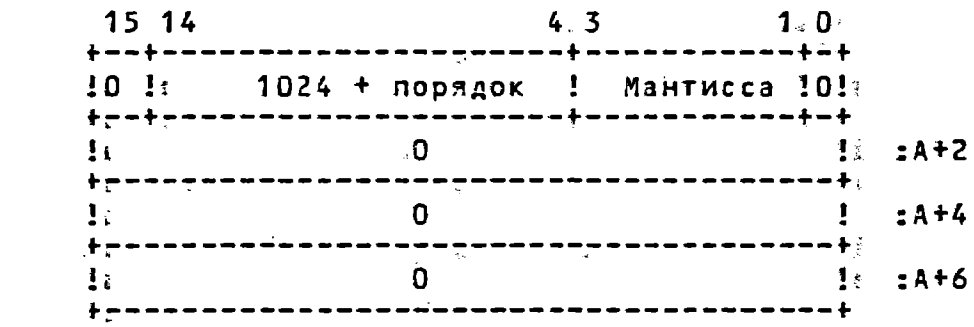
Формат



В случае плавающих чисел формата F разряды 63:32 отсутствуют в операнде.

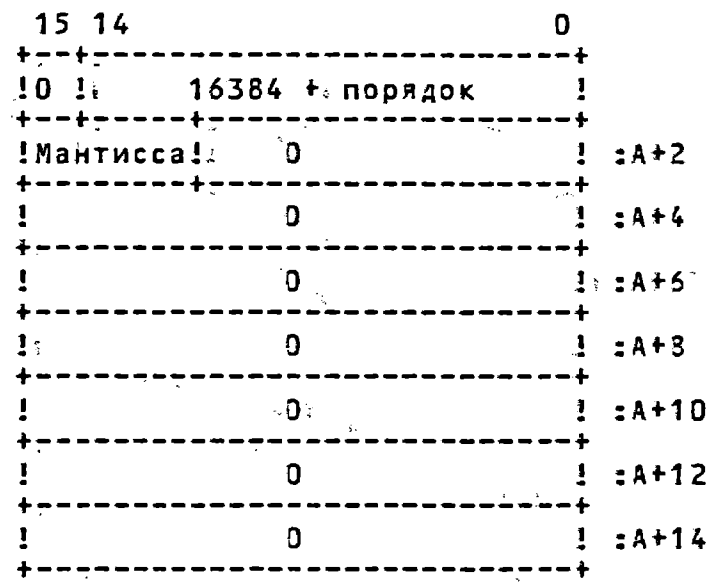
Для формирования числа формата G поля порядка и мантиссы используются следующим образом:

Формат



Для формирования числа формата H поля порядка и мантиссы используются следующим образом:

Формат



Допустимый диапазон представлений значений литералов приведен в табл. 5.

Таблица 5

1) пс	Дробная часть							
	0	1	2	3	4	5	6	7
0	1/2	9/16	5/8	11/16	3/4	13/16	7/8	15/16
1	1	1 1/8	1 1/4	1 3/8	1 1/2	1 5/8	1 3/4	1 7/8
2	2	2 1/4	2 1/2	2 3/4	3	3 1/4	3 1/2	3 3/4
3	4	4 1/2	5	5 1/2	6	6 1/2	7	7 1/2
4	8	9	10	11	12	13	14	15
5	16	18	20	22	24	26	28	30
6	32	36	40	44	48	52	56	60
7	64	72	80	88	96	104	112	120

1) пс. - показатель степени

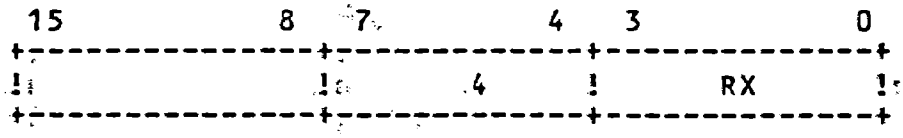
Так как в литеральном режиме нет адреса операнда, то он не может использоваться для операндов адресного типа. Литеральный режим не может использоваться также для операндов с доступом типа модификации и записи. В этих случаях возникает ошибка неверной адресации (см. документ [1]).

Литеральный режим является весьма эффективным способом задания целочисленных констанат в диапазоне от 0 до 63 и констант в формате плавающей запятой. Литеральные значения, превышающие допустимый диапазон, могут быть представлены с помощью режима, использующего счетчик инструкций (непосредственный режим).

Обозначение литерального режима на ассемблере: S^#литерал

3.4.9. Индексный режим

Формат спецификатора операнда



Разряды 15:8 содержат второй спецификатор операнда, который называется базовым спецификатором операнда для всех режимов адресации, кроме регистрового, литерального и индексного. Определение во втором спецификаторе регистрового, литерального или индексного режима адресации приводит к ошибке неверного режима адресации. Если базовый спецификатор требует расширения, то это расширение следует непос-

редственно за спецификатором. На базовый спецификатор операнда распространяются те же ограничения, как и при обычном его использовании. Если использование спецификатора является неверным (т.е. вызывает ошибку или приводит к непредсказуемым результатам при некоторых условиях), то использование его в качестве базового спецификатора в индексном режиме при тех же условиях также является неверным.

Операнд, который определяется индексным режимом адресации, называется первичным. Обычно базовый спецификатор операнда используется для определения адреса операнда. Этот адрес называется базовым адресом операнда (BOA). Содержимое индексного регистра X умножается на размер первичного операнда в байтах (1 - для байта; 2 - для слова; 4 - для длинного слова и числа с плавающей запятой F_формата; 8 - для квадрослова и числа D_формата или G_формата; 16 - для октаслова и числа H_формата) и складывается с базовым адресом (BOA).

Полученная сумма является адресом первичного операнда:

OA = BOA + (размер * (RX))

операнд = (OA)

Если в базовом спецификаторе определен режим с автоувеличением или с автоуменьшением, то величина зависит от размера в байтах первичного операнда.

Индексный режим обеспечивает весьма общий и эффективный доступ к элементам массива. Базовый адрес массива определяется адресом, вычисленным в соответствии с базовым спецификатором. Содержимое индексного регистра используется

как логический индекс внутри массива. Логический индекс преобразуется в реальное смещение (в байтах) путем умножения содержимого индексного регистра на размер первичного операнда в байтах.

На использование индексного регистра накладываются некоторые ограничения. Счетчик инструкций не может использоваться в качестве индексного регистра. В противном случае возникает ошибка резервного режима адресации. Если в базовом спецификаторе операнда определен режим адресации, который приводит к модификации регистра (т.е. режимы с автоувеличением, с автоуменьшением и косвенный режим с автоувеличением), то этот регистр не может быть использован в качестве индексного. В противном случае адрес первичного операнда будет непредсказуемым.

Название режимов адресации образуется из названия режима адресации в базовом спецификаторе операнда с добавлением слова "индексный". Индексный регистр обозначается RX для того, чтобы отличить его от регистра RN, который определяется в базовом спецификаторе операнда.

1) косвенно-регистровый режим с индексацией - (RN)[RX];

2) индексный режим с автоувеличением - (RN)+[RX];

Или непосредственный индексный режим - I^#константа [RX], который распознается ассемблером. При этом адрес операнда не зависит от значения константы;

3) косвенный индексный режим с автоувеличением - @(RN)+[RX] или абсолютный индексный режим - @#адрес[RX];

4) индексный режим с автоуменьшением - -(RN)[RX];

5) индексный режим со смещением в виде байта, слова, длинного слова - B^D(RN)[RX], W^D(RN)[RX], L^D(RN)[RX];

6) косвенный индексный режим со смещением в виде байта, слова, длинного слова - @B^D(RN)[RX], @W^D(RN)[RX], @L^D(RN)[RX].

3.5. Сводная таблица общих режимов адресации

3.5.1. Адресация через регистры общего назначения

Табл. 6 и табл. 7 суммируют сведения относительно режимов адресации СМ 1700.

Формат

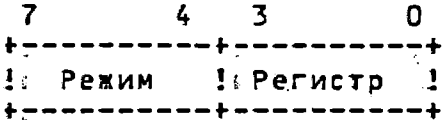


Таблица 6

Шест-надца-терич-ный	Ассемблер	R 1)	M 2)	W 3)	A 4)	V 5)	PC	SP	AP&FP	Индекс-ация
0-3	S^#литерал	Y 6)	F 7)	F	F	F	- 8)	-	-	F
4	I[RX]	Y 9)	Y	Y	Y	Y	F	Y	Y	F
5	RN	Y	Y	Y	F	Y	U 10)	UQ 11)	UO 12)	F
6	(RN)	Y	Y	Y	Y	Y	U	Y	Y	Y
7	-(RN)	Y	Y	Y	Y	Y	U	Y	Y	UX 13)
8	(RN)+	Y	Y	Y	Y	Y	P 14)	Y	Y	UX
9	@(RN)+	Y	Y	Y	Y	Y	P	Y	Y	UX
A	B^D(RN)	Y 15)	Y	Y	Y	Y	P	Y	Y	Y
B	@B^D(RN)	Y	Y	Y	Y	Y	P	Y	Y	Y
C	W^D(RN)	Y	Y	Y	Y	Y	P	Y	Y	Y
D	@W^D(RN)	Y	Y	Y	Y	Y	P	Y	Y	Y
E	L^D(RN)	Y	Y	Y	Y	Y	P	Y	Y	Y
F	@L^D(RN)	Y	Y	Y	Y	Y	P	Y	Y	Y

1) R - доступ типа чтения;;

2) M - доступ типа модификации;

3) W - доступ типа записи;

4) A - доступ типа адреса;

5) V - доступ типа поля;

6) Y - да, всегда правильный режим адресации;

7) F - ошибка резервного режима адресации;

8) - - логически невозможно;

9) I - любой индексный режим адресации;

10) U - непредсказуемо;

11) UQ - непредсказуемо для квадрослов и октаслов и чисел с плавающей запятой форматов D, G и H (также для полей переменной длины, если позиция + размер превышает 32);

12) UO - непредсказуемо для октаслов и чисел с плавающей запятой H_формата;

13) UX - непредсказуемо, если в качестве индексного и базового используется один и тот же регистр;

14) P - адресация через счетчик инструкций;

15) D - смещение.

3.5.2. Адресация через счетчик инструкций (R15)

Формат:

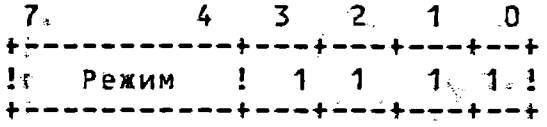


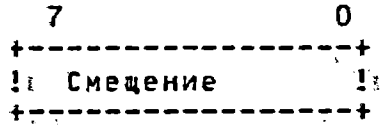
Таблица 7

Шестнадца-теричный	Ассемблер	R	M	W	A	V	PC	SP	Индек-сация
8	I^#константа	Y	U	U	Y	Y	-	-	Y
9	@#адрес	Y	Y	Y	Y	Y	-	-	Y
A	@B^адрес	Y	Y	Y	Y	Y	-	-	Y
B	@B^адрес	Y	Y	Y	Y	Y	-	-	Y
C	W^адрес	Y	Y	Y	Y	Y	-	-	Y
D	@W^адрес	Y	Y	Y	Y	Y	-	-	Y
E	L^адрес	Y	Y	Y	Y	Y	-	-	Y
F	@L^адрес	Y	Y	Y	Y	Y	-	-	Y

3.6. Форматы адресации режима переходов

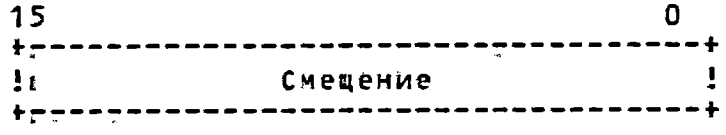
Существует 2 формата для спецификатора операндов.

Формат 1



спецификатор операнда представляет собой смещение со знаком в виде байта.

Формат 2



спецификатор операнда представляет собой смещение со знаком в виде слова.

При вычислении адреса перехода знак смещения, представленный в байте или слове, расширяется до 32 разрядов и складывается с модифицированным содержимым PC. Модифицированное содержимое PC представляет собой адрес байта, следующего непосредственно за спецификатором операнда. В результате формируется адрес перехода A.

A = PC + SEXT (смещение)

В обоих случаях адрес перехода на ассемблере обозначается A.

3.7. Условия обработки спецификатора операнда

Для каждой инструкции выполняются следующие три фазы:

1) каждый спецификатор в порядке своего появления в потоке команд интерпретируется следующим образом:

- если тип доступа - чтение, то вычисляется адрес операнда, этот операнд считывается и запоминается;
- если тип доступа - запись, то вычисляется адрес операнда и запоминается;
- если тип доступа - модификация, то вычисляется и запоминается адрес операнда, операнд считывается и запоминается;
- если тип доступа - адрес, то вычисляется и запоминается адрес;
- если тип доступа - адрес перехода, то запоминается спецификатор операнда.

2) производится операция, определенная в инструкции;

3) запись результата, используя сохраненный адрес (адреса), производится в той последовательности, в какой указаны спецификаторы операндов для данной инструкции.

Примечание. Инструкции для работы со строками (символьными, зонными и упакованными десятичными) представляют собой исключение в той части, что в фазе 2 и фазе 3. Промежуточные результаты запоминаются до завершения операции, определенной инструкцией. Инструкции для работы с битовыми полями переменной длины интерпретируют спецификаторы позиции, размера и базового адреса как составной спецификатор операнда (см. приложение документа [1]). Если во время выполнения фаз 1 или 2 одновременно возникнет несколько исключительных ситуаций, то порядок, в котором они обрабатываются, будет непредсказуемым. Такое может случиться, например, при выполнении инструкции с плавающей запятой, в которой операнд - приемник с доступом типа записи использует резервный режим адресации и, кроме того, в результате операции возникает ошибка переполнения.

Результатом этих условий является то, что:

1) операции с автоувеличением и с автоуменьшением производятся в процессе обработки спецификатора операнда и последующие спецификаторы операнда используют значения регистров, уже модифицированных предыдущими операциями.

2) в отличие от фазы 1 все входные операнды считываются и все адреса выходных операндов вычисляются до того, как запоминаются какие-либо результаты выполнения инструкции.

3) обработка операндов с типом доступа "модификация" не производится как непрерывная операция. Поэтому операнды с типом доступа "модификация" не могут использоваться для синхронизации (см. документ [1]).

4) если инструкция обращается к двум операндам с типом доступа "модификация" или запись по одному и тому же адресу, то значение первого операнда будет замещено значением второго.

4. Инструкции

4.1. Система инструкций

В этом разделе описываются те инструкции, которые используются в любом программном обеспечении и будут реализованы для всех моделей семейства 32-разрядных СМ ЭВМ. Другие инструкции, которые предназначены для работы со специализированной частью архитектуры СМ 1700 (например, диспетчеризация памяти, прерывания и исключительные ситуации, диспетчеризация процессора, внутренние регистры процессора) и которые обычно используются в привилегированном программном обеспечении, описаны в разделах, посвященных соответствующим особенностям архитектуры. Список инструкций приведен в приложении.

4.1.1. Описание инструкций

Система инструкций делится на 12 основных секций:

1) целочисленная арифметика и логика;

2) адресные;

3) битовые поля переменной длины;

4) управления;

5) вызов процедур;

6) инструкции различного назначения;

7) очереди;

8) плавающая арифметика;

9) символьные строки;

10) циклический контроль по избыточности;

11) десятичные строки;

12) редактирование.

Внутри каждой секции близкие по свойствам инструкции объединены в группы и описываются вместе. Описание группы инструкций состоит из следующих пунктов:

1) название группы;

2) формат каждой инструкции в группе. Здесь определяются название и тип каждого спецификатора операнда инструкции и порядок, в котором они появляются в памяти. Спецификаторы операндов слева направо располагаются по возрастающим адресам памяти;

3) действия, выполняемые инструкцией;

4) способ установки кодов условий;

5) исключительные ситуации, характерные для данной инструкции. Исключительные ситуации, которые встречаются во всех инструкциях (то есть неправильный или резервный режим адресации, прерывание по разряду слежения, нарушение диспечеризации памяти и так далее), не приводятся;

6) код операции, мнемоника и название каждой инструкции в группе. Код операции приводится в шестнадцатеричном виде;

7) описание инструкции;

8) дополнительные замечания по инструкции.

4.1.2. Обозначение спецификатора операнда

Спецификатор операнда описывается следующим образом:

<название>.<тип доступа><тип данных>

где <название> - определяет операнд в контексте инструкции. Название часто приводится в сокращенном виде;

<тип доступа> - буква, обозначающая тип доступа спецификатора операнда:

a

- вычисляется действительный адрес операнда. Адрес формируется в длинном слове, которое является фактическим операндом инструкции. Контекст вычисления адреса определяется типом данных, т.е. величиной, используемой при автоувеличении, автоуменьшении и индексации;

b

- нет ссылок на операнд. Спецификатор операнда представляет собой смещение перехода. Размер смещения перехода определяется типом данных;

m

- операнд считывается, может быть модифицирован и записывается. Отметим, что эта операция не является неделимой операцией с памятью. Отметим также, что если операнд фактически не модифицируется, то он может и не записываться обратно. Несмотря на это, операнды с таким типом доступа всегда проверяются на возможность как чтения, так и записи;

R

- операнд только читается;

V

- вычисляется исполнительный адрес операнда. Если исполнительный адрес - адрес памяти, то он сохраняется в длинном слове, которое является фактическим операндом инструкции. Контекст вычисления адреса определяется типом данных. Если исполнительным адресом является RN, то операндом является RN или R[N+1]'RN;

W

- операнд только записывается

<тип данных> - это символ, обозначающий тип данных операнда:

B

- байт;

D

- число с плавающей запятой D_формата;

F

- число с плавающей запятой F_формата;

G

- число с плавающей запятой G_формата;

H

- число с плавающей запятой H_формата;

L

- длинное слово;

O

- октаслово;

Q

- квадрослово;

W

- слово;

X

- тип первого данного определяется инструкцией;

Y

- тип второго данного определяется инструкцией.

4.1.3. Обозначения, используемые при описании операции

Действия, выполняемые каждой инструкцией, описываются в виде последовательности операторов управления и присвоения алгольного типа. Формально этот синтаксис не описывается. Предполагается, что пользователь с ним знаком. Используемые обозначения являются расширением обозначений, введенных в разделе 3.

+

- сложение;

-

- вычитание, знак минуса;

*

- умножение;

/

- деление (только частное);

**

- возведение в степень;

- конкатенация;

<-

- пересылка;

=

- присвоение;

RN или R[N]

- содержимое регистра RN;

PC, SP, FP или AP

- содержимое регистров R15, R14, R13, R12;

PSW

- содержимое слова состояния процессора;

PSL

- содержимое длинного слова состояния процессора;

(X)

- содержимое ячейки памяти с адресом X;

(X)+

- содержимое ячейки памяти с адресом X; X увеличивается на размер операнда, который находится по адресу X;

-(X)

- X уменьшается на размер операнда, который находится по адресу X; содержимое ячейки памяти по адресу X;

<X:Y>

- модификатор, который ограничивает диапазон от раз-

рядной позиции X до разрядной позиции Y включительно;

<X1, X2, ..., XN>

- модификатор, в котором перечислены разряды X1, X2, ... XN;

( ):

- арифметические скобки, которые используются для индикации предшествования;

AND

- логическое "И";

OR

- логическое "ИЛИ";

XOR

- исключающее "ИЛИ";

NOT

- инверсия;

LSS

- меньше чем (со знаком);

LSSU

- меньше чем (без знака);

LEQ

- меньше чем или равно (со знаком);

LEQU

- меньше чем или равно (без знака);

EQL

- равно (со знаком);

EQLU

- равно (без знака);

NEQ

- не равно (со знаком);

NEQU

- не равно (без знака);

GEQ

- больше чем или равно (со знаком);

GEQU

- больше чем или равно (без знака);

GTR

- больше чем (со знаком);

GTRU

- больше чем (без знака);

SEXT(X)

- знак X расширяется до размера, требуемого операнда;

ZEXT(X)

- X дополняется ведущими нулями до размера, требуемого операнда;

REM(X,Y)

- остаток, полученный при делении X на Y, при этом X/Y и REM(X,Y) имеют одинаковые знаки;

MINU (X,Y)

- минимум из значений X и Y (без знака);

MAXU (X,Y).

- максимум из значений X и Y (без знака).

Кроме того, нужно учитывать следующее:

- преобразуются только операнды или группы операндов, которые приводятся в левой части операторов присваивания, за исключением действий, вызываемых операциями автоувеличения, автоуменьшения и увеличения значения счетчика инструкций;
- в выражениях не используется никаких операторов предшествования, за исключением того, что пересылка (<-) имеет самый низкий приоритет. Отношения предшествования вводятся явным образом с помощью круглых скобок;
- все арифметические и логические операторы определяются с учетом контекста своих операндов. Например, операция "+", примененная к операндам с плавающей запятой, означает сложение таких чисел, а та же операция, примененная к байтовым операндам, означает целочисленное сложение байтов;
- операнды в инструкции вычисляются в соответствии с определением спецификаторов операндов (см. раздел 3). Порядок, в котором операнды появляются в описании инструкции, не имеет значения при вычислении;
- коды условий устанавливаются в соответствии с реально установленными результатами, а не в соответствии с "истинными" результатами (которые могут вычисляться для большей точности). Например, сумма двух положительных чисел может быть определена, вследствие переполнения, в виде отрицательного значения и в кодах условий будет зафиксирован этот результат,

несмотря на то, что "истинный" результат является положительным.

4.2. Целочисленные арифметические и логические инструкции

В этом подразделе описываются следующие инструкции:

- сложение выравненных слов ADAWI ADD.RW, SUM.MW;
- сложение двухоперандное ADD(B,W,L)2 ADD.RX, SUM.MX;
- сложение трехоперандное ADD(B,W,L)3 ADDL.RX, ADD2.RX, SUM.WX;
- сложение с переносом ADWC ADD.RL, SUM.ML;
- арифметический сдвиг ASH(L,Q) CNT.RB, SRC.RX, DST.WX;
- очистка разрядов двухоперандная BIC(B,W,L)2 MASK.RX, DST.MX;
- очистка разрядов трехоперандная BIC(B,W,L)3 MASK.RX, SRC.RX, DST.WX;
- установка разрядов двухоперандная BIS(B,W,L)2 MASK.RX, DST.MX;
- установка разрядов трехоперандная BIS(B,W,L)3 MASK.RX, SRC.RX, DST.WX;
- проверка разрядов BIT(B,W,L) MASK.RX, SRC.RX;
- очистка CLR(B,W,L,Q) DST.WX;
- сравнение CMP(B,W,L) SRC1.RX, SRC2.RX;
- преобразование CVT(B,W,L)(B,W,L) SRC.RX, DST.WY;
- автоуменьшение DEC(B,W,L) DIF.MX;
- деление двухоперандное DIV(B,W,L)2 DIVR,RX, QUO,MX;

- деление трехоперандное DIV(B,W,L)3 DIVR.RX, DIVD.RX, QUO.WX;
- расширенное деление EDIV DIVR.RG, DIVD.RG, QUO.WL, REM.WL;
- расширенное умножение EMUL MULR.RL, MULD.RL, ADD.RL, PROD.WG;
- автоувеличение INC(B,W,L) SUM.MX;
- пересылка с инверсией MCOM(B,W,L) SRC.RX, DST.WX;
- пересылка с отрицанием MNEG(B,W,L) SRC.RX, DST.WX;
- пересылка MOV(B,W,L,Q) SRC.RX, DST,WX;
- пересылка с лидирующими нулями MOVZ(BW,BL,WL) SRC.RX, DST,WY;
- умножение двухоперандное MUL(B,W,L)2 MULR.RX, PROD.MX;
- умножение трехоперандное MUL(B,W,L)3 MULR.RX, MULD.RX, PROD.WX;
- загрузка длинного слова в стек PUSHL SRC.RL, [-(SP).WL];
- циклический сдвиг длинного слова ROTL CNT.RB, SRC.GL, DST.WL;
- вычитание с переносом SBWC SUB.RL, DIF.ML;
- вычитание двухоперандное SUB(B,W,L)2 SUB.RX, DIF.MX;
- вычитание трехоперандное SUB(B,W,L)3 SUB.RX, MIN.RX, DIF.WX;
- проверка TST(B,W,L) SRC.RX;
- исключающее "ИЛИ" двухоперандное XOR(B,W,L)2 MASK.RX, DST.MX.

- исключающее "ИЛИ" трехоперандное XOR(B,W,L)3 MASK.RX, SRC.RX, DST.WX.

ADAWI сложение выровненных слов (с блокировкой памяти):

Формат:

Код операции ADD.RW, SUM.MW

Коды условий:

N <- SUM LSS 0;

Z <- SUM EQL 0;

V <- (целочисленное переполнение);

C <- (перенос из самого старшего значащего разряда);

Исключительная ситуация:

ошибка резервного операнда

целочисленное переполнение

Коды операций:

53 ADAWI сложение выровненных слов (с блокировкой памяти)

Описание.

Операнды складываются, и полученная сумма запоминается по адресу второго операнда. Операция предусматривает блокировку памяти для того, чтобы предотвратить аналогичные операции со стороны других процессоров в мультипроцессорной системе.

Второй операнд должен быть выровнен по границе слова, т.е. разряд 0 его адреса должен быть равен нулю. В противном случае возникает ошибка резервного операнда.

Примечания:

1. Целочисленное переполнение возникает, если слагаемые имели один знак, а результат - противоположный. При переполнении, в сумму заносятся младшие разряды истинного результата.

2. Если слагаемые операнды адресованы таким образом, что перекрывают друг друга, то результат и коды условий будут непредсказуемыми.

ADD сложение

Формат:

код операции ADD.RX, SUM.MX — 2 операнда

код операции ADD1.RX, ADD2.RX, SUM.WX — 3 операнда

Коды условий:

N <- SUM LSS 0;

Z <- SUM EQQ 0;

V <- (целочисленное переполнение);

C <- (перенос из самого старшего значащего разряда);

Исключительная ситуация:

целочисленное переполнение

Коды операций:

80	ADDB2	сложение байтов двухоперандное
81	ADDB3	сложение байтов трехоперандное
A0	ADDW2	сложение слов двухоперандное
A1	ADDW3	сложение слов трехоперандное
C0	ADDL2	сложение длинных слов двухоперандное

C1. ADDL3 сложение длинных слов трехоперандное

Описание.

При двухоперандном формате операнды складываются, а результат запоминается по адресу второго операнда. При трехоперандном формате два первых операнда складываются, а результат запоминается по адресу третьего операнда.

Примечание. Целочисленное переполнение возникает, если оба слагаемых имели один знак, а результат - противоположный. При переполнении в сумму заносятся младшие разряды истинного результата.

ADWC сложение с переносом

Формат:

код операции ADD.RL, SUM.ML

коды условий:

N <- SUM LSS 0;

Z <- SUM EQL 0;

V <- (целочисленное переполнение);

C <- (перенос из самого старшего значащего разряда);

Исключительная ситуация:

целочисленное переполнение

Коды операций:

D8 ADWC сложение с переносом.

Описание.

Значение суммы двух операндов складывается со значением кода условия C, а результат запоминается по адресу

второго операнда.

Примечания:

1. При переполнении во второй операнд заносятся младшие разряды истинного результата.

2. Обе операции сложения, предусмотренные алгоритмом, производятся одновременно.

ASH арифметический сдвиг

Формат:

код операции CNT.RB, SRC.RX, DST.WX

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- (целочисленное переполнение);

C <- 0;

Исключительная ситуация:

целочисленное переполнение

Коды операций:

78 ASHL арифметический сдвиг длинного слова

79 ASHQ арифметический сдвиг квадрослова

Описание.

Операнд-источник SRC арифметически сдвигается на число разрядов, определяемое операндом-счетчиком CNT, а результат запоминается по адресу операнда-приемника DST. Содержимое операнда SRC не изменяется. При положительном содержимом операнда-счетчика производится сдвиг влево. Младшие разряды при этом заполняются нулями.

При отрицательном содержимом операнда-счетчика производится сдвиг вправо. При этом происходит распространение старшего (знакового) разряда в старших разрядах результата. При нулевом значении операнда-счетчика содержимое операнда-источника без изменения заносится по адресу операнда-приемника.

Примечания:

1. При сдвиге влево целочисленное переполнение возникает, если какой-либо разряд, сдвинутый на позицию знакового разряда, отличается от знакового разряда операнда-источника.

2. Если CNT > 32 (ASHL) или CNT > 64 (ASHQ), то в операнд-приемник заносится нуль.

3. Если CNT < - 31 (ASHL) или CNT < - 63 (ASHQ), то во всех разрядах операнда-приемника записывается знаковый разряд операнда-источника.

BIC очистка разрядов

Формат:

код операции MASK.RX, DST.MX 2 операнда

код операции MASK.RX, SRC.RX, DST.WX 3 операнда

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует.

Коды операций:

8A	BICB2	очистка разрядов в байте двухоперандный
8B	BICB3	очистка разрядов в байте трехоперандный
AA	BICW2	очистка разрядов в слове двухоперандный
AB	BICW3	очистка разрядов в слове трехоперандный
CA	BICL2	очистка разрядов в длинном слове двухоперандный
CB	BICL3	очистка разрядов в длинном слове трехоперандный

Описание.

При двухоперандном формате содержимое операнда-приемника DST логически умножается на инвертированное содержимое операнда-маски MASK, а результат запоминается по адресу операнда-приемника. При трехоперандном формате содержимое операнда-источника SRC логически умножается на инвертированное содержимое операнда-маски, а результат запоминается по адресу операнда-приемника.

BIS	установка разрядов
---	---

Формат:

код операции MASK.RX, DST.MX	2 операнда
код операции MASK.RX, SRC.RX, DST.WX	3 операнда

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

88 BIS32 установка разрядов в байте
двухоперандная

89 BIS33 установка разрядов в байте
трехоперандная

A8 BISW2 установка разрядов в слове
двухоперандная

A9 BISW3 установка разрядов в слове
трехоперандная

C8 BISL2 установка разрядов в длинном слове
двухоперандная

C9 BISL3 установка разрядов в длинном слове
трехоперандная

Описание.

При двухоперандном формате содержимое операнда-маски логически складывается с содержимым операнда-приемника и по адресу операнда-приемника запоминается результат.

При трехоперандном формате содержимое операнда-маски логически складывается с содержимым операнда-источника, а результат запоминается по адресу операнда-приемника.

BIT проверка разрядов

Формат:

код операций MASK.RX, SRC.RX

Коды условий:

N <- TMP LSS 0

Z <- TMP EOL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

93 BITB проверка разрядов в байте

B3 BITW проверка разрядов в слове

D3 BITL проверка разрядов в длинном слове

Описание.

Содержимое операнда-маски MASK логически умножается на содержимое операнда-источника SRC. Содержимое обоих операндов остается без изменений. Результатом операции является соответствующая установка кодов условий.

CLR очистка

Формат:

код операций DST.WX

Коды условий:

N <- 0;

Z <- 1;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

94	CLRB	очистка байта
B4	CLRW	очистка слова
D4	CLRL	очистка длинного слова
7C	CLRQ	очистка квадрослова
7CFD	CLRQ	очистка октаслова

Описание.

По адресу операнда-приемника DST заносится нуль.

Примечание. Инструкция CLR DST эквивалентна MOV S^#0,DST но на один байт короче.

CMP сравнение

Формат:

код операции SRC1.RX, SRC2.RX

Коды условий:

N <- SRC1 LSS SRC2;

Z <- SRC1 EQL SRC2;

V <- 0;

C <- SRCL LSSU SRC2;

Исключительная ситуация:

отсутствует

Коды операции:

91. CMPB сравнение байтов

B1. CMPW сравнение слов

D1. CMPL сравнение длинных слов

Описание.

Первый операнд-источник SRC1 сравнивается со вторым операндом-источником SRC2. Единственным результатом операции является соответствующая установка кодов условий.

CVT преобразование

Формат:

код операции SRC.RX, DST.WY

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- (целочисленное переполнение);

Исключительная ситуация:

целочисленное переполнение

Коды операций:

99	CVTBW	преобразование байта в слово
98	CVTBL	преобразование байта в длинное слово
33	CVTWB	преобразование слова в байт
32	CVTWL	преобразование слова в длинное слово
F6	CVTLB	преобразование длинного слова в байт
F7	CVTLW	преобразование длинного слова в слово

Описание.

Производится преобразование операнда-источника SRC в соответствии с типом данных операнда-приемника DST, в котором запоминается результат преобразования. При преобразовании из короткого типа в более длинный тип данных производится расширение знака. При преобразовании в более короткий тип данных производится усечение старших разрядов.

Примечание. Целочисленное переполнение возникает, если какой-либо из усекаемых разрядов операнда-источника не равен знаковому разряду операнда-приемника.

DEC автоуменьшение

Формат:

код операции DIF.MX

Коды условий:

N <- DIF LSS 0;

Z <- DIF EQL 0;

V <- (целочисленное переполнение);

C <- (заимствование из старшего разряда);

Исключительная ситуация:

целочисленное переполнение

Коды операций:

97	DECB	автоуменьшение байта
B7	DECW	автоуменьшение слова
D7	DECL	автоуменьшение длинного слова

Описание.

Из содержимого операнда-приемника DIF вычитается единица и результат заносится по адресу операнда-приемника.

Примечания:

1. Целочисленное переполнение возникает, если производится вычитание из наибольшего отрицательного числа. При переполнении в операнд-приемник заносится наибольшее положительное число.

2. Инструкция DEC эквивалентна SUB S^#1,DIF, но на один байт короче.

DIV деление

Формат:

код операции DIVR.RX, QUO.MX 2 операнда

код операции DIDR.RX, DIVD.RX, QUO.WX 3 операнда

Операция:

```
QUO <- QUO / DIVR;  !2 операнда
QUO <- DIVD / DIVR; !3 операнда
```

Коды условий:

```
N <- QUO LSS 0;
Z <- QUO EQL 0;
V <- (целочисленное переполнение) или
     (деление на ноль);
C <- 0;
```

Исключительная ситуация:

целочисленное переполнение

деление на нуль

Коды операций:

86	DIVB2	деление байтов двухоперандное
87	DIVB3	деление байтов трехоперандное
A6	DIVW2	деление слов двухоперандное
A7	DIVW3	деление слов трехоперандное
C6	DIVL2	деление длинных слов двухоперандное
C7	DIVL3	деление длинных слов трехоперандное

Описание.

При двухоперандном формате содержимое операнда-приемника QUD делится на содержимое операнда-делителя DIVR. Результат помещается в QUD. При трехоперандном формате содержимое операнда-делимого DIVD делится на содержимое операнда-делителя DIVR. Результат запоминается по адресу операнда-приемника QUD.

Примечания:

1. Деление производится таким образом, что остаток (если он не равен нулю) имеет тот же знак, что и делимое, т.е. результат усекается в сторону нуля.

2. Целочисленное переполнение возникает тогда и только тогда, когда наибольшее отрицательное целое число делится на -1. При переполнении содержимое операндов устанавливается так, как это описано в примечании 3.

3. Если содержимое операнда-делителя равно нулю, то при двухоперандном формате содержимое операнда-частного остается без изменения, а при трехоперандном формате по адресу операнда-частного заносится содержимое операнда-делимого.

EDIV расширеное деление

Формат:

код операции DIVR.R1, DIVD.RQ, QUO.W1, REM.W1

Коды условий:

N <- QUO LSS 0;

Z <- QUO EQL 0;

V <- (целочисленное переполнение) или
(деление на нуль);

C <- 0;

Исключительная ситуация:

целочисленное переполнение

деление на нуль

Коды операций:

7B EDIV расширенное деление

Описание.

Содержимое операнда-делимого DIVD делится на содержимое операнда-делителя DIVR. Частное и остаток заносятся по адресам операнда-частного QUO и операнда-остатка REM соответственно.

Примечания:

1. Деление производится таким образом, что содержимое операнда-остатка (если оно не равно нулю) имеет тот же знак, что и содержимое операнда-делимого.

2. При переполнении содержимое операндов устанавливается так, как это описано в п.3.

3. Если содержимое операнда-делителя равно нулю, то по адресу операнда-частного заносятся разряды 31:0 операнда-делимого, а по адресу операнда-остатка заносится нуль.

EMUL расширенное умножение

Формат:

код операций MULR.R1, MULD.R1, ADD.R1, PROD.WQ

Коды условий:

N <- PROD LSS 0;

Z <- PROD EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует

Код операций:

7A EMUL расширенное умножение

Описание.

Содержимое операнда-множимого MULR умножается на содержимое операнда-множителя MULD. Результат умножения имеет двойную длину. Знак содержимого операнда-слагаемого ADD расширяется до двойной длины, после чего это содержимое прибавляется к результату. Окончательный результат запоминается по адресу операнда-результата PROD.

INC автоувеличение

Формат:

код операции: SUM.MX

Коды условий:

N: <- SUM LSS: 0;

Z <- SUM EQL 0;

V <- (целочисленное переполнение);

C <- (передается из самого старшего разряда);

Исключительная ситуация:

целочисленное переполнение

Коды операций:

96	INCB	автоувеличение байта
B6	INCW	автоувеличение слова
D6	INCL	автоувеличение длинного слова

Описание.

К содержимому операнда прибавляется единица и по адресу операнда запоминается результат.

Примечания:

1. Арифметическое переполнение возникает, если инкрементируется наибольшее положительное число. При переполнении заносится наибольшее отрицательное число.

2. Инструкция INC SUM эквивалентна ADD S^#1,SUM, но в памяти занимает на 1 байт меньше.

MCOM пересылка с инверсией

Формат:

код операции: SRC.RX, DST.WX

Коды условий:

N: <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операции:

92 MCOMB пересылка байта с инверсией

B2 MCOMW пересылка слова с инверсией

D2 MCOML пересылка длинного слова с инверсией

Описание.

По адресу операнда-приемника DST заносится инвертированное содержимое операнда-источника SRC.

Примечание. Целочисленное переполнение возникает, если операнд-источник содержит наибольшее отрицательное целое число (которое не имеет положительного эквивалента). При переполнении по адресу операнда-приемника заносится содержимое операнда-источника.

MNEG пересылка с отрицанием

Формат:

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- (целочисленное переполнение);

C <- DST NEQ 0;

Исключительная ситуация:

целочисленное переполнение

Коды операций:

BE	MNEGB	пересылка байта с отрицанием
AE	MNEGW	пересылка слова с отрицанием
CE	MNEGL	пересылка длинного слова с отрицанием

Описание.

По адресу операнда-приемника DST заносится отрицательное содержимое операнда-источника SRC.

Примечание. Целочисленное переполнение возникает, если операнд-источник содержит наибольшее отрицательное целое число (которое не имеет положительного эквивалента). При переполнении по адресу операнда-приемника заносится содержимое операнда-источника.

MOV пересылка

Формат:

код операции SRC.RX, DST.WX

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

90	MOVB	пересылка байта
B0	MOVW	пересылка слова
D0	MOVL	пересылка длинного слова
7D	MOVQ	пересылка квадрослова
7DFD	MOVO	пересылка октаслова

Описание.

По адресу операнда-приемника DST заносится содержимое операнда-источника SRC.

MOVZ пересылка с лидирующими нулями

Формат:

код операции SRC.RX, DST.WY

Коды условий:

N <- 0;

Z <- DST EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

9B	MOVZBW	пересылка байта в слово с лидирующими нулями
9A	MOVZBL	пересылка байта в длинное слово с

лидирующими нулями

3C MOVZWL пересылка слова в длинное слово с лидирующими нулями

Описание.

При пересылке байта в слово (MOVZBW) в разряды 7:0 операнда-приемника DST заносится содержимое операнда-источника SRC, а разряды 15:8 заполняются нулями. При пересылке байта в длинное слово (MOVZBL) в разряды 7:0 операнда-приемника заносится содержимое операнда-источника, а разряды 31:8 заполняются нулями. При пересылке слова в длинное слово (MOVZWL) в разряды 15:0 операнда-приемника заносится содержимое операнда-источника, а разряды 31:16 заполняются нулями.

MUL умножение

Формат:

код операции MULR.RX, PROD.MX 2 операнда

код операции MULR.RX, MULD.RX, PROD.WX 3 операнда

Коды условий:

N <- PROD LSS 0;

Z <- PROD EQL 0;

V <- (целочисленное переполнение);

C <- 0;

Исключительная ситуация:

целочисленное переполнение

Коды операций:

84 MULB2 умножение байтов двухоперандное

85 MUL33 умножение байтов трехоперандное

A4 MULW2 умножение слов двухоперандное

A5 MULW3 умножение слов трехоперандное

C4 MULL2 умножение длинных слов двухоперандное

C5 MULL3 умножение длинных слов трехоперандное

Описание.

При двухоперандном формате содержимое операнда-произведения PROD умножается на содержимое операнда-множителя MULR и младшая часть результата, имеющего двойную длину, заносится по адресу операнда-произведения. При трехоперандном формате содержимое операнда-множимого MULR умножается на содержимое операнда-множителя MULD и младшая часть результата, имеющего двойную длину, заносится по адресу операнда-произведения PROD.

Примечание. Целочисленное переполнение возникает, если старшая часть результата, имеющего двойную длину, не совпадает с расширением знака младшей части.

PUSHL занесение в стек длинного слова

Формат:

код операции SRC.R1

Коды условий:

N <- SRC LSS 0;

Z <- SRC EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

DD PUSHL занесение в стек длинного слова

Описание.

Длинное слово, которое содержится по адресу операнда-источника, заносится в стек.

Примечание. Эта инструкция эквивалентна MOVL SRC,-(SP), но занимает в памяти на 1 байт меньше.

ROTL циклический сдвиг длинного слова

Формат:

код операции CNT.RB, SRC.R1, DST.W1

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

9C ROTL циклический сдвиг длинного слова

Описание.

Содержимое операнда-источника SRC циклически сдвигается на число разрядов, которое определяется содержимым операнда-счетчика CNT, а результат засылается по адресу операнда-приемника DST. Содержимое операнда-источника не

изменяется. При положительном содержимом операнда-счетчика производится сдвиг влево, а при отрицательном - сдвиг вправо. При нулевом содержимом операнда-счетчика по адресу операнда-приемника заносится содержимое операнда-источника.

SBWC вычитание с переносом

Формат:

код операции SUB.R1, DIF.M1

Коды условий:

N <- DIF LSS 0;

Z <- DIF EQL 0;

V <- (целочисленное переполнение);

C <- (заимствование в самый старший разряд);

Исключительная ситуация:

целочисленное переполнение

Коды операций:

D9. SBWC вычитание с переносом

Описание.

Содержимое операнда-вычитаемого SUB и разряда кода условия с вычитаются из содержимого операнда DIF и результат запоминается по адресу операнда DIF.

Примечания:

1. При переполнении по адресу операнда DIF заносятся младшие разряды истинного результата.

2. Оба вычитания в инструкции производятся одновременно.

SUB вычитание

Формат:

код операции SUB.RX, DIF.MX 2 операнда

код операции SUB.RX, MIN.RX, DIF.WX 3 операнда

Коды условий:

N <- DIF LSS 0;

Z <- DIF EQL 0;

V <- (целочисленное переполнение);

C <- (заимствование из самого старшего разряда);

Исключительная ситуация:

целочисленное переполнение

Коды операций:

82	SUBB2	вычитание для байтов двух
83	SUBB3	и трехоперандное
A2	SUBW2	вычитание для слов двух и
A3	SUBW3	трехоперандное
C2	SUBL2	вычитание для длинных слов двух
C3	SUBL3	и трехоперандное

Описание.

При двухоперандном формате содержимое операнда-вычитаемого SUB вычитается из содержимого операнда-уменьшаемого DIF и результат запоминается по адресу операнда-уменьшаемого. При трехоперандном формате содержимое операнда-вычитаемого SUB вычитается из содержимого операнда-уменьшаемого MIN, а результат заносится по адресу операнда-разности DIF.

Примечание. Целочисленное переполнение возникает в том случае, если входные операнды имеют противоположные знаки, а знак результата совпадает со знаком вычитаемого. При переполнении по адресу операнда-разности заносятся младшие разряды истинного результата.

TST проверка

Формат:

код операции SRC.RX

Коды условий:

N <- SRC LSS 0;

Z <- SRC EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует

Коды операций:

95 TSTB

B5 TSTW тестирование байта,

D5 TSTL слова и длинного слова

Описание.

Коды условий устанавливаются в соответствии с содержимым операнда-источника.

Примечание. Инструкция TST SRC эквивалентна CMP SRC,S^#0, но занимает в памяти на 1 байт меньше.

4.3. Адресные инструкции

В данном подразделе описываются следующие инструкции:

- пересылка адреса MOVA[B,W,L=F,Q=D=G,O=H] SRC.AX, DST.W1;
- загрузка адреса в стек PUSHA[B,W,L=F,Q=D=G,O=H] SRC.AX, [-(SP).W1]

MOVA пересылка адреса

Формат:

код операции SRC.AX, DST.W1

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

9E	MOVAB	пересылка адреса байта
3E	MOVAW	пересылка адреса слова
DE	MOVAL,	пересылка адреса длинного слова
	MOVAF	пересылка адреса числа F_формата
7E	MOVAQ,	пересылка адреса квадрослова
	MOVAD,	пересылка адреса числа D_формата
	MOVAG	пересылка адреса числа G_формата
7EFD	MOVAH	пересылка адреса числа H_формата
	MOVAO	пересылка октаслова

Описание.

Адрес операнда-источника SRC пересылается по адресу операнда-приемника DST. Контекст, по которому производится вычисление операнда-источника, определяется типом данных, используемым в инструкции. Обращение к операнду, адрес которого заносится в операнд-приемник, не производится.

Примечание. Операнд-источник имеет адресный тип доступа, что приводит к пересылке адреса операнда.

PUSHA: загрузка в стек адреса

Формат:

код операции SRC.AX

Коды условий:

N <- SRC LSS 0

Z <- SRC EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

9F	PUSHAB	загрузка в стек адреса байта
3F	PUSHAW	загрузка в стек адреса слова
DF	PUSHAL	загрузка в стек адреса длинного слова
	PUSHAF	загрузка в стек адреса числа F_формата
7F	PUSHAQ	загрузка в стек адреса квадрослова
	PUSHAD	загрузка в стек адреса числа D_формата
	PUSHAG	загрузка в стек адреса числа G_формата

7FFD PUSHAH загрузка в стек адреса числа H_формата

PUSHAO загрузка в стек адреса октаслова

Описание.

Содержимое операнда-источника SRC засылается в стек. Контекст, по которому производится вычисление адреса операнда-источника, определяется типом данных, используемым в инструкции. Обращение к операнду, адрес которого засылается в стек, не производится.

Примечания:

1. Инструкция PUSHA SRC эквивалентна MOVA SRC, -(SP), но занимает в памяти на 1 байт меньше.

2. Операнд-источник имеет адресный тип доступа, что приводит к засылке в стек адреса операнда.

4.4. Инструкции для битовых полей переменной длины

Битовое поле переменной длины определяется тремя операндами:

1) операнд, указывающий местоположение длинного слова;

2) операнд размера поля, представляющий собой байт. Значение этого операнда должно изменяться в диапазоне от 0 до 32. В противном случае возникает ошибка резервного операнда;

3) операнд, содержащий базовый адрес, относительно которого определяется позиция поля. Этот адрес формируется операндом, имеющим адресный тип доступа. Однако, в отличие от других случаев использования спецификаторов операнда

адресного типа, в спецификаторе операнда можно использовать регистровый режим. В этом случае поле будет расположено в регистре, который определен в спецификаторе операнда (или в двух связанных регистрах). Если поле расположено в регистре и размер не равен нулю, то операнд позиции должен содержать значение в диапазоне от 0 до 31. В противном случае возникает ошибка резервного операнда.

В данном подразделе описываются следующие инструкции:

- сравнение поля CMPV POS.RL, SIZE.RB, BASE.VB, [FIELD.RV], SRC.RL;
- сравнение поля с расширением ведущими нулями CMPZV POS.RL, SIZE.RB, BASE.VB, [FIELD.RV], SRC.RL;
- скопировать поле EXTV POS.RL, SIZE.RB, BASE.VB, [FIELD.RV], DST.WL;
- скопировать поле с расширением лидирующими нулями EXTZV POS.RL, SIZE.RB, BASE.VB, [FIELD.RV], DST.WL;
- поиск первого (C-сброшенного, S-установленного) бита FF(S,C) STARTPOS.RL, SIZE.RB, BASE.VB, [FIELD.RV], FINDPOS.WL;
- запись поля INSV SRC.RL, SIZE.RB, BASE.VB, [FIELD.WV].

Дополнительные инструкции для работы с битовыми полями переменной длины описаны в подразделе 4.5:

- переход, если бит (C-сброшен, S-установлен) BB(S,C) POS.RL, BASE.VB, DISPL.BB, [FIELD.RV];
- переход, если бит (C-сброшен, S-установлен) и безусловно (C-сброшен, S-установлен) бит BB(S,C) (S,C)

POS.RL, BASE.VB,DISPL.BB, [FIELD.MV];

- переход, если бит (C-сброшен, S-установлен) и безусловно (C-сброшен, S-установлен) с блокировкой памяти BB(SS,CC)1 POS.RL, BASE.VB, DISPL.BB, [FIELD.MV].

CMP сравнение полей

Формат:

код операции POS.RL, SIZE.RB, BASE.VB, SRC.RL

Коды условий:

N <- TMP LSS SRC;

Z <- TMP EQL SRC;

V <- 0;

C <- TMP LSSU SRC;

Исключительная ситуация:

резервный операнд

Коды операций:

EC	CMPV	сравнение полей
ED	CMPZV	сравнение полей с расширением лидирующими нулями

Описание.

Содержимое поля, которое определяется операндами позиции POS, размера SIZE и базового адреса BASE пересылается для последующего сравнения во внутренний регистр процессора. При описании кодов условий он обозначен как TMP. Затем он сравнивается с содержимым операнда-источника. При выполнении инструкции CMPV содержимое операнда-источника сравнивается с содержимым поля, знак которого расширяется до

двойного слова. При выполнении инструкции CMPZV содержимое операнда-источника сравнивается с содержимым поля, дополненного лидирующими нулями. Единственным результатом выполнения инструкции является соответствующая установка кодов условий.

Примечания:

1. Ошибка резервного операнда возникает, если:

- размер больше 32;
- позиция больше 31, размер не равен 0 и поле расположено в регистрах.

2. При возникновении ошибки резервного операнда содержимое кодов условий является непредсказуемым.

EXT: чтение поля

Формат:

код операции POS.RL, SIZE.RB, BASE.VB, DST.WL

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- C;

Исключительная ситуация:

резервный операнд

Коды операций:

EE EXTV скопировать поле

EF EXTZV скопировать поле с расширением лидирующими нулями

Описание.

При выполнении инструкции EXTV содержимое поля бит, указанного операндами POS, SIZE, CASE расширяется влево, до величины длинного слова, значением крайне левого бита (знакового бита данного поля бит переменной длины). И затем засылается по адресу, определенному операндом-приемником DST.

При выполнении инструкции EXTZV по адресу приемника засылается определяемое операндами позиции, размера и базового адреса содержимое поля, дополненное лидирующими нулями. Если операнд размера равен нулю, то единственным результатом является занесение нуля по адресу приемника и соответствующая установка кодов условий.

Примечания:

1. Ошибка резервного операнда возникает, если:

- размер больше 32;
- значение позиции больше 31, размер не равен нулю и поле содержится в регистрах.

2. При возникновении ошибки резервного операнда содержимое операнда-приемника остается без изменения, а состояние кодов условий является непредсказуемым.

FF поиск первого (C-сброшенного, S-установленного) разряда

Формат:

код операции STARTPOS.RL, SIZE.RB, BASE.VB, FINDPOS.WL

Коды условий:

N <- 0;

Z <- (бит не найден);

V <- 0;

C <- 0;

Исключительная ситуация:

резервный операнд

Коды операций:

EB FFC поиск первого сброшенного разряда

EA FFS поиск первого установленного разряда

Описание.

Просматривается поле бит, определяемое операндами начальной позиции STARTPOS, размера SIZE и базового адреса BASE. Просмотр ведется слева направо, т.е. начиная с разряда 0 в сторону старшего разряда. При этом производится поиск первого разряда, состояние которого указано в инструкции. Если такой разряд найден, то его позиция заносится по адресу операнда найденной позиции FINDPOS, а разряд кода условия Z становится равным нулю. Если такой разряд не найден, то по адресу операнда FINDPOS заносится позиция (относительно базового адреса) разряда, который находится на один разряд левее позиции поля, а разряд кода условия Z устанавливается в единицу. Если операнд SIZE равен нулю, то по адресу операнда найденной позиции заносится содержимое начальной позиции, а разряд кода условия Z устанавливается в 1.

Примечания:

1. Ошибка резервного операнда возникает, если

- размер больше 32;
- значение начальной позиции больше 31, размер не равен нулю и поле содержится в регистрах.

2. При возникновении ошибок резервного операнда содержимое операнда найденной позиции остается без изменения, а состояние кодов условий является непредсказуемым.

INSV записать поле

Формат:

код операции SRC.RL, POS.RL, SIZE.RB, BASE.VB

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

резервный операнд

Коды операций:

FD INSV запись поля

Описание

Содержимое разрядов в диапозоне (SIZE-1):0 операнда-источника SRC заносится в поле, определяемое операндами позиции POS, размера SIZE и базового адреса BASE. Если размер операнда равен нулю, то единственным рузультатом выполнения инструкции является соответствующая установ-

ка кодов условий.

Примечания:

1. Ошибка резервного операнда возникает, если:

- размер больше 32;
- значение позиции больше 31, размер не равен нулю, поле содержится в регистрах.

2. При возникновении ошибки резерваного операнда содержимое поля остается без изменения, а состояние кодов условий является непредсказуемым.

4.5. Инструкции управления

В большинстве случаев использования архитектурных возможностей СМ 1700 наибольшая скорость выполнения достигается, если передача управления в этих инструкциях производится на границу длинного слова.

В этом подразделе описываются следующие инструкции:

- сложение, сравнение и переход ACB(B,W,L,F,D,G,H) LIMIT.RX,ADD.RX,INDEX.MX,DISPL.BW при сложении положительных чисел производится сравнение на "меньше или равно", а при сложении отрицательных чисел - на "больше или равно";
- прибавить единицу и перейти, если "меньше или равно" AOBLEQ LIMIT.RL, INDEX.ML, DISPL.BB;
- прибавить единицу и перейти, если "меньше" AOBLSS LIMIT.RL, INDEX.ML, DISPL.BB;
- переход по условию B(условие) DISPL.BB;

Условия перехода описаны в табл. 8.

Таблица 8

Условие	Название
LSS	"меньше"
LEQ	"меньше или равно"
EQL, EQLU	"равно" (знаковое, беззнаковое)
NEQ, NEQU	"не равно" (знаковое, беззнаковое)
GEQ	"больше или равно"
GTR	"больше"
LSSU, CS	"меньше" беззнаковое, установлен "C"
LEQU	"меньше или равно" беззнаковое
GEQU, CC	"больше или равно" беззнаковое сброшен "C"
GTRU	"больше" беззнаковое
VS	установлен "V"
VC	сброшен "V"

- переход, если бит (C-сброшен, S-установлен) BB(S,C) POS.RL, BASE.VB, DISPL.BB, (FIELD.RV);
- переход, если бит (C-сброшен, S-установлен), и безусловно (C-сбросить, S-установить) бит BB(S,C)(S,C) POS.RL, BASE.VB, DISPL.BB, (FIELD.MV);
- переход, если бит (C-сброшен, S-установлен) и безусловно (C-сбросить, S-установить) бит с блокировкой памяти BB(SS,CC)I POS.RL, BASE.VD, DISPL.BB, (FIELD.MV);

- переход, если младший бит (C-сброшен, S-установлен) BLB(S,C) SRC.RL, DISPL.BB;
- переход со смещением (в байтах, словах) BR(B,W) DISPL.BX;
- переход к подпрограмме со смещением (в байтах в словах) BSB(B,W) DISPL.BX, [-(SP).WL];
- выбор CASE(B,W,L) SELECTOR.RX, BASE.RX, LIMIT.RX, DISPL.BW-LIST;
- универсальный переход JMP DST.AB;
- универсальный переход к подпрограмме JSB DST.AB, [-(SP).WL];
- возврат из подпрограммы RSB [(SP)+.RL];
- вычитание единицы и переход по "больше или равно" SOBGEQ INDEX.ML, DISPL.BB;
- вычитание единицы и переход по "Больше" SOBGTR INDEX.ML, DISPL.BB.

ACB сложение, сравнение и переход

Формат:

код операции LIMIT.RX, ADD.RX, INDEX.MX, DISPL.BW

Коды условий:

N <- INDEX LSS 0;

Z <- INDEX EQL 0;

V <- целочисленное или плавающее переполнение;

C <- C;

Исключительная ситуация:

целочисленное переполнение

переполнение для чисел в формате с плавающей запятой по верхней границе диапазона (переполнение)

переполнение для чисел в формате с плавающей запятой по нижней границе диапазона (потеря значимости)

резервный операнд

Коды операций:

9D	ACBB	сложение байтов, сравнение и переход
3D	ACBW	сложение слов, сравнение и переход
F1	ACBL	сложение длинных слов, сравнение и
4F	ACBF	сложение чисел F_формата, сравнение и переход
6F	ACBD	сложение чисел D_формата, сложение и переход
4FFD	ACBG	сложение чисел G_формата, сложение и переход
6FFD	ACBH	сложение чисел H_формата, сложение и переход

Описание.

Содержимое операнда-слагаемого ADD складывается с содержимым операнда-индекса, а результат засылается по адресу операнда-индекса INDEX. Содержимое операнда-индекса сравнивается с содержимым операнда-границы LIMIT. Если содержимое операнда-слагаемого положительно (или равно нулю) и результат сравнения "меньше или равно" или если содержимое операнда-слагаемого отрицательно и результат сравнения "больше или равно", то к содержимому PC прибавляется смещение перехода с расширением знака DISPL, и

результат заносится в PC.

Примечания:

1. С помощью инструкции ACB эффективно реализуются операторы FOR или DO в языках высокого уровня, так как результат сравнения индекса и границы зависит от знака слагаемого.

2. При целочисленном переполнении по адресу операнда-индекса заносятся младшие разряды истинного результата. Сравнение и вычисление адреса перехода производится обычным способом с учетом модифицированного значения операнда-индекса.

3. При переполнении чисел с плавающей запятой по нижней границе диапазона, если разряд FU слова состояния процессора сброшен, то по адресу операнда-индекса заносится нуль, и сравнение и вычисление адреса перехода производится обычным образом. Если разряд FU установлен, то возникает ошибка, а содержимое операнда-индекса остается без изменения.

4. При переполнении чисел с плавающей запятой по верхней границе диапапазона возникает ошибка, а содержимое операнда-индекса остается без изменения.

5. При возникновении ошибки резервного операнда содержимое операнда-индекса остается без изменения, а состояние кодов условий является непредсказуемым.

6. Во всех случаях, кроме примечания 5, состояние разряда "C" остается без изменения.

AOBLEQ прибавить единицу и перейти, если "меньше или равно"

Формат:

код операции LIMIT.RL, INDEX.ML, DISPL.BB

Коды условий:

N <- INDEX LSS 0;

Z <- INDEX EQL 0;

V <- (целочисленное переполнение);

C <- C;

Исключительная ситуация:

целочисленное переполнение

Коды операций:

F3 AOBLEQ прибавить единицу и переход, если "меньше или равно"

Описание.

К содержимому операнда-индекса INDEX прибавляется единица и результат запоминается по адресу операнда-индекса. Содержимое операнда-индекса сравнивается с содержимым операнда-границы LIMIT. Если результат сравнения "меньше или равно", то к содержимому PC прибавляется смещение перехода с расширением знака DISPL и результат заносится в PC.

Примечания:

1. Целочисленное переполнение возникает, если перед сложением в операнде-индексе находится наибольшее положительное число. В этом случае по адресу операнда-индекса заносится наибольшее отрицательное число и происходит пере-

ход.

2. Состояние разряда C не изменяется.

AOBLSS прибавить единицу и перейти, если "меньше"

Формат:

код операции LIMIT.RL, INDEX.ML, DISPL.BB

Коды условий:

N <- INDEX LSS 0;

Z <- INDEX EQL 0;

V <- (целочисленное переполнение);

C <- C;

Исключительная ситуация:

целочисленное переполнение

Коды операций:

F2 AOBLSS прибавить единицу и переход, если "меньше"

Описание.

К содержимому операнда-индекса INDEX прибавляется единица и результат запоминается по адресу операнда-индекса. Содержимое операнда-индекса сравнивается с содержимым операнда-границы LIMIT. Если результат сравнения "меньше", то к содержимому PC прибавляется смещение перехода с расширением знака DISPL и результат заносится в PC.

Примечания:

1. Целочисленное переполнение возникает, если перед сложением в операнде-индексе находится наибольшее положительное число. При переполнении по адресу операнда-индекса заносится наибольшее отрицательное число и таким образом

происходит переход (за исключением случая, когда операнд-границы содержит наибольшее отрицательное целое число).

2. Состояние разряда C не изменяется.

B переход по условию

Формат:

код операции DISPL.BB

Коды условий:

N <- N

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций и условия:

14 (N OR Z) EQL 0	BGTR	знаковый переход по "Больше"
15 (N OR Z) EQL 1	BLEQ	знаковый переход по "меньше или равно"
12 Z EQL 0	BNEQ,	знаковый переход по "неравенству".
	BNEQU	беззнаковый переход по "неравенству".
13 Z EQL 1	BEQL,	знаковый переход по "равенству"
	BEQLU	беззнаковый переход по

			"равенству".
18	N EQL 0	BGEQ	знаковый переход по "Больше или равно"
19	N EQL 1	BLSS	знаковый переход по "меньше"
1A	(C OR Z) EQL 0	BGTRU	беззнаковый переход по "Больше"
1B	(C OR Z) EQL 1	BLEQU	беззнаковый переход по "меньше или равно"
1C	V EQL 0	BVC	переход по отсутствию переполнения
1D	V EQL 1	BVS	переход по наличию переполнения
1E	C EQL 0	BGEQU,	беззнаковый переход по больше или равно"
		BCC	переход по отсутствию переноса
1F	C EQL 1	BLSSU,	беззнаковый переход по "меньше".
		BCS	переход по наличию переноса

Описание.

Проверяется состояние кодов условий и если условие, указанное в инструкции, выполняется, то к счётчику команд прибавляется смещение перехода DISPL, знак которого предварительно расширяется.

Примечание. Инструкции условных переходов обеспечивают

значительную гибкость при организации ветвлений, но и требуют осторожности при выборе правильной инструкции перехода. Инструкции условных переходов можно разделить на три перекрывающих друг друга группы:

1) группа инструкций переходов по переполнению и переносу:

```
BVS     V EQL 1
BVC     V EQL 0
BCS     C EQL 1
BCC     C EQL 0
```

Эти инструкции используются для проверки наличия переполнения (когда внутренние прерывания по переполнению запрещены), для выполнения арифметических операций с многократной точностью и для других специальных целей;

2) группа инструкций беззнаковых переходов:

```
BLSSU   C EQL 1
BLEQU   (C OR Z) EQL 1
BEQLU   Z EQL 1
BNEQU   Z EQL 0
BGEQU   C EQL 0
BGTRU   (C OR Z) EQL 0
```

Эти инструкции следуют за арифметическими инструкциями и инструкциями для работы с полями бит переменой длины, в которых операнды интерпретируются как целые числа без знака, а также после адресных инструкций и инструкций для работы со строками;

3) группа инструкций знаковых переходов:

BLSS N EQL 1

BLEQ (N OR Z) EQL 1

BEQL Z EQL 1

BNEQ Z EQL 0

BGEQ N EQL 0

BGTR (N OR Z) EQL 0

Эти инструкции следуют за арифметическими инструкциями и инструкциями для работы с полями бит переменной длины, в которых операнды интерпретируются как целые числа со знаком, а также после инструкций для работы с числами с плавающей запятой и десятичными строками.

BB переход по состоянию разряда

Формат:

Код операции: POS.RL, BASE.VB, DISPL.BB

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

резервный операнд

Коды операций:

E0 BBS переход, если разряд установлен

E1 BBC переход, если разряд сброшен

Описание.

Проверяется состояние одноразрядного поля, которое определяется операндами позиции POS и базового адреса BASE. Если это состояние соответствует указанному в инструкции, то к счетчику инструкций прибавляется смещение перехода DISPL, знак которого предварительно расширяется.

Примечание. Если позиция больше 31 и операнд содержится в регистре, то возникает ошибка резервного операнда. При возникновении ошибки резервного операнда состояние кодов условий является непредсказуемым.

BB переход по состоянию разряда (и модификация разряда без блокировки памяти)

Формат:

код операции POS.RL, BASE.VB, DISPL.BB

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

резервный операнд

Коды операций:

E2 BBSS переход по установленному разряду и установить разряд

E3 BBCS переход по сброшенному разряду и установить разряд

E4 BBSC переход по установленному разряду и сбросить разряд

E5 BBCC переход по сброшенному разряду и сбросить разряд

Описание.

Проверяется состояние одноразрядного поля, которое определяется операндами позиции POS и базового адреса BASE. Если это состояние соответствует указанному в инструкции, то к счетчику инструкций прибавляется смещение перехода DISPL, знак которого предварительно расширяется. Проверяемый разряд устанавливается в указанное в инструкции новое состояние независимо от того, производится переход или нет.

Примечания:

1. Если позиция больше 31 и операнд содержится в регистре, то возникает ошибка резервного операнда.

2. При возникновении ошибки резервного операнда состояние кодов условий является непредсказуемым.

3. Модификация разряда производится без блокировки памяти.

BB переход по состоянию разряда

(с блокировкой памяти)

Формат:

код операции POS.RL, BASE.VB, DISPL.BB

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

резервный операнд

Коды операций:

E6 BBSSI переход по установленному разряду и установка разряда с блокировкой памяти

E7 BBCCI переход по сброшенному разряду и сброс разряда с блокировкой памяти

Описание.

Проверяется состояние одноразрядного поля, которое определяется операндами позиции POS и базового адреса BASE. Если это состояние соответствует указанному в инструкции, то к счетчику инструкций прибавляется смещение перехода DISPL, знак которого предварительно расширяется. Проверяемый разряд устанавливается в указанное в инструкции новое состояние независимо от того, производится переход или нет. Если этот разряд находится в оперативной памяти, то его чтение и установка нового состояния производится с блокировкой памяти. Во время этих операций никакие другие процессоры или устройства ввода-вывода не имеют к нему доступа.

Примечания:

1. Если позиция больше 31 и операнд содержится в

регистре, то возникает ошибка резервного операнда.

2. При возникновении ошибки резервного операнда состояние кодов условий является непредсказуемым.

3. За исключением блокировки памяти, инструкция BBSSI эквивалентна инструкции BBSS, а BBCCI эквивалентна BBCC.

4. Эта инструкция предназначена для организации взаимодействия с другими процессорами или устройствами. Например, для реализации "ожидания занятости" можно использовать инструкцию:

1¤: BBSSI BIT, BASE, 1¤

BLB переход по состоянию младшего разряда

Формат:

код операции SRC.RL, DISPL.BB

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

E8 BLBS переход по установленному младшему разряду

E9 BLBC переход по сброшенному младшему разряду

Описание.

Проверяется состояние младшего разряда (разряда 0) операнда-источника SRC. Если это состояние равно указанному

в инструкции состояния, то к счетчику инструкций прибавляется смещение перехода DISPL, знак которого предварительно расширяется.

BR безусловный переход

Формат:

код операции DISPL.BX

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

11 BRB безусловный переход со смещением в байте

31 BRW безусловный переход со смещением в слове

Описание.

К счетчику инструкций прибавляется смещение перехода DISPL, знак которого предварительно расширяется.

BSB переход к подпрограмме

Формат:

код операции DISPL.BX

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

10 BSBB перехрд к подпрограмме со смещением в байте

30 BSBW переход к подпрограмме со смещением в слове

Описание.

Содержимое счетчика инструкций заносится в стек в виде длинного слова. К счетчику инструкций прибавляется смещение перехода DISPL, знак которого предварительно расширяется.

CASE выбор

Формат:

Код операции: SELECTOR.RX, BASE.RX, LIMIT.RX,

DISPL[0].BW,...., DISPL[LIMIT].BW

Коды условий:

N <- TMP LSS LIMIT;

Z <- TMP EQL LIMIT;

V <- 0;

C <- TMP LSSU LIMIT;

Исключительная ситуация:

отсутствует

Коды операций:

8F CASEB выбор перехода со смещением в байте

AF CASEW выбор перехода со смещением в слове

CF CASEL выбор перехода со смещением в длинном

слове

Описание.

Содержимое операнда-базы BASE вычитается из содержимого операнда-селектора SELECTOR и заносится во внутренний регистр процессора TMP. Этот промежуточный операнд сравнивается с содержимым операнда-границы LIMIT. Если результат сравнения меньше или равен нулю (в беззнаковом диапазоне), то к счетчику инструкций прибавляется смещение перехода, выбираемое по значению промежуточного операнда от DISPL[0] до DISPL[LIMIT] в противном случае к счетчику инструкций дважды прибавляется содержимое операнда границы плюс единица. Таким образом производится переход непосредственно за массив смещений перехода, т.е. к следующей инструкции. Независимо от вида перехода коды условий устанавливаются в соответствии с результатом сравнения содержимого промежуточного операнда и операнда-границы.

Примечания:

1. После вычисления операндов счетчик инструкций указывает на DISPL[0], а не на следующую инструкцию. Смещение перехода определяется относительно адреса DISPL[0].

2. Содержимое операндов-селектора и базы может представлять собой целые числа как со знаком, так и без знака.

JMP — универсальный переход

Формат:

код операции DST.AB

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

17 JMP универсальный переход

В счетчик инструкций заносится содержимое операнда-приемника DST.

JSB универсальный переход к подпрограмме

Формат:

код операции DST.AB

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

16 JSB универсальный переход к подпрограмме

Описание.

Содержимое счетчика инструкций заносится в стек в виде длинного слова. Затем в счетчик инструкций заносится содер-

жимое операнда-приемника DST.

Примечание. Эта инструкция может быть использована для вызовов сопрограмм через стек. Такой вызов имеет вид:

JSB @(SP)+

RSB возврат из подпрограммы

Формат:

код операции

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

05 RSB возврат из подпрограммы

Описание.

В счетчик инструкций заносится содержимое длинного слова из стека.

Примечания:

1. Эта инструкция используется для возврата из подпрограмм, вызов которых производился инструкциями BSBB, BSBW и JSB.

2. Эта инструкция эквивалентна JMP @(SP)+, но занимает в оперативной памяти на один байт меньше.

SOBGEQ вычесть единицу и перейти, если "больше или равно"

Формат:

код операции INDEX.ML, DISPL.BB

Коды условий:

N <- INDEX LSS 0;

Z <- INDEX EQL 0;

V <- (целочисленное переполнение);

C <- C;

Исключительная ситуация:

целочисленное переполнение

Коды операций:

F4 SOBGEQ вычесть единицу и перейти, если "больше или равно".

Описание.

Из содержимого операнда-индекса INDEX вычитается единица. Если результат больше или равен нулю, то к содержимому счетчика инструкций прибавляется смещение перехода DISPL, знак которого предварительно расширяется.

Примечания:

1. Если до вычитания в операнде-индексе содержится наибольшее отрицательное число, то происходит целочисленное переполнение. При переполнении по адресу операнда-индекса заносится наибольшее положительное число. Поэтому, происходит переход.

2. Состояние разряда C не изменяется.

SOBGTR вычесть единицу и перейти, если "больше".

Формат:

код операции INDEX.ML, DISPL.BB

Коды условий:

N <- INDEX LSS 0;

Z <- INDEX EQL 0;

V <- (целочисленное переполнение);

C <- C;

Исключительная ситуация:

целочисленное переполнение

Коды операций:

F5 SOBGTR вычесть единицу и перейти, если "больше"

Описание.

Из содержимого операнда-индекса INDEX вычитается единица. Если результат больше нуля, то к содержимому счетчика инструкций прибавляется смещение перехода DISPL, знак которого предварительно расширяется.

Примечания:

1. Если до вычитания в операнде-индексе содержится наибольшее отрицательное число, то происходит целочисленное переполнение. При переполнении по адресу операнда-индекса заносится наибольшее положительное целое число. Поэтому происходит переход.

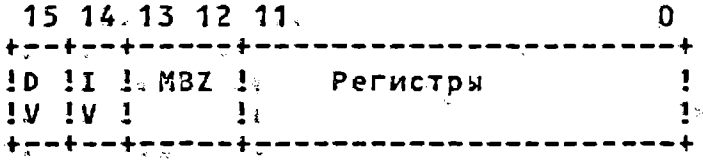
2. Состояние разряда C не изменяется.

4.6. Инструкции вызова процедуры

Инструкции вызова процедуры используются для реализации стандартного алгоритма вызова процедуры. Две инструкции используются для вызова, а третья - для возврата из процедуры. Инструкция CALLG производит вызов процедуры со списком аргументов, расположенных в произвольном месте. Инструкция CALLS производит вызов процедуры со списком аргументов, расположенным в стеке. После возврата из процедуры этот список автоматически удаляется из стека. В обеих инструкциях определяется начальный адрес вызываемой процедуры. Подразумевается, что по начальному адресу расположено слово, называемое входной маской, за которым следует собственно процедура. Процедура заканчивается выполнением инструкции RET.

Входная маска определяет используемые в процедуре регистры и разрешения внутренних прерываний по переполнению.

Формат маски:



При выполнении инструкциии CALL стек выравнивается по границе длинного слова, а разряды разрешения внутренних прерываний в слове состояния процессора устанавливаются в состояние, которое обеспечивает требуемую реакцию вызванной процедуры при возникновении внутренних прерываний. Разреше-

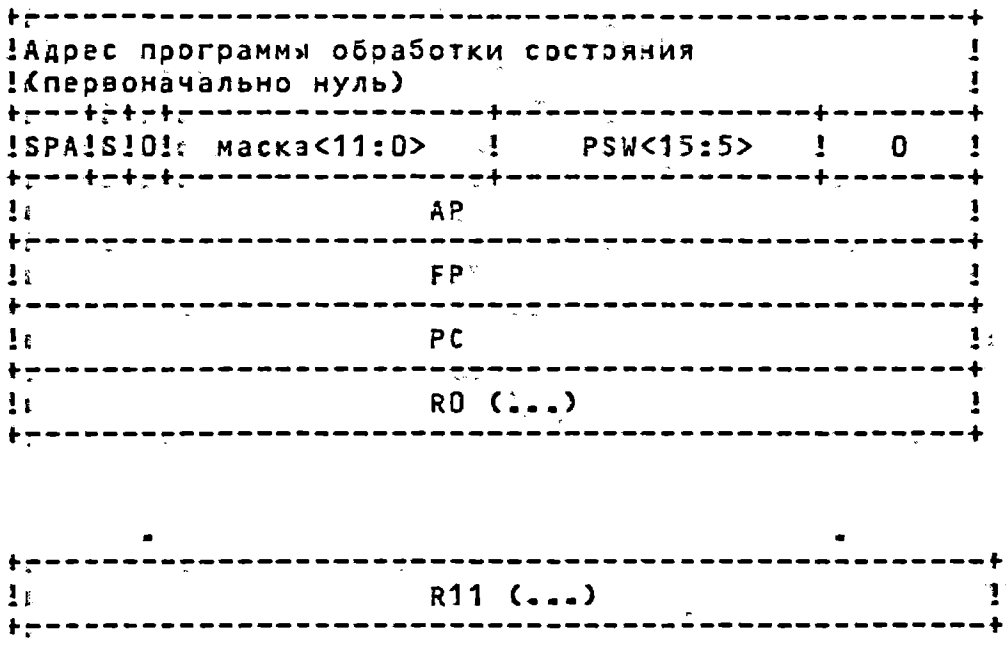
ния внутренних прерываний по целочисленному и десятичному переполнению устанавливаются в соответствии с разрядами 14 и 15 входной маски. Разряд разрешения внутреннего прерывания по переполнению снизу (потеря значимости) при операциях с числами с плавающей запятой сбрасывается в нуль. Содержимое регистров с R0 по R11, помеченных единицами во входной маске в разрядах с 0 по 11, запоминается в стеке и восстанавливается при выполнении инструкций возврата RET. Кроме того, содержимое регистров PC, SP, FP и AP всегда сохраняется при выполнении инструкций вызова и восстанавливается при выполнении инструкций возврата.

Все внешние процедуры вызова, генерируемые языками высокого уровня, и все межмодульные обращения к основным подсистемам операционной системы соответствуют общим правилам вызова процедуры. При стандартном вызове процедуры требуется, чтобы все используемые регистры в процедуре с R2 по R11 были указаны во входной маске. При стандартном вызове процедуры не обеспечивается сохранение содержимого регистров R0 и R1.

Для того, чтобы сохранить состояние процессора, при выполнении инструкции вызова в стеке формируется структура, которая называется кадром вызова или кадром стека. Такой кадр содержит значение регистров, слово состояния процессора, маску запоминания регистров и несколько управляющих разрядов. Кадр также содержит длинное слово, которое при вызове процедуры обнуляется. Это длинное слово используется для реализации управления.

После выполнения инструкции вызова регистр указателя кадра FP содержит адрес кадра вызова. Инструкция возврата использует содержимое указателя кадра при восстановлении состояния процессора.

Формат кадра вызова



Поле SPA определяет число байтов для выравнивания SP. Его значение от 0 до 3. Значение бита S зависит от вызова и равно:

S=1, если CALLS; S=0, если CALLG.

Коды условий и PSW <T> обнуляются в кадре вызова.

Содержимое разрядов PSW<3:0> в кадре при выполнении инструкции возврата устанавливается в соответствии с результатом последней инструкции в процедуре. Сходным образом содержимое PSW<4> кадра при выполнении инструкции возврата определяет состояние разряда слежения в слове состояния процессора.

В подразделе описываются следующие инструкции:

- вызов процедуры общего вида CALLG ARGLIST.AB, DST.AB, [-SP).W*];
- вызов процедуры стекового вида CALLS NUMARG.RL, DST.AB, [-(SP).W*];
- возврат из процедуры RET [(SP)+.R*];

CALLS вызов процедуры общего вида

Формат:

код операции ARGLIST.AB, DST.AB

Коды условий:

N <- 0;

Z <- 0;

V <- 0;

C <- 0;

Исключительная ситуация:

резервный операнд

Коды операций:

FA CALLG вызов процедуры общего вида

Описание.

Значение указателя стека запоминается во внутреннем регистре. Затем разряды 1:0 указателя стека обнуляются. Таким образом, происходит выравнивание стека по границе длинного слова. Содержимое входной маски процедуры последовательно опрашивается от разряда 0 до разряда 11 и содержимое регистров, номера которых соответствуют установленным в маске разрядам, заносится в стек в виде длинного слова. В

стек в виде длинных слов заносится также содержимое регистров PC, FP и AP. Коды условий обнуляются. В стек заносится длинное слово, которое состоит из сохраненных в разрядах 31:30 двух младших разрядов указателя стека, нулей в разрядах 29 и 28, младших 12 разрядов входной маски процедуры в разрядах 27:16 и слово состояния процессора с обнуленным разрядом слежения в разрядах 15:0. Затем в стек заносится длинное слово с нулевым содержимым. В регистр FP заносится содержимое указателя стека. В регистр AP заносится содержимое операнда списка аргументов ARGLIST. Разрешения внутренних прерываний в слове состояния процессора устанавливаются в указанное состояние. Разрешения внутренних прерываний по целочисленному и десятичному переполнению устанавливаются в соответствии с содержимым разрядов 14 и 15 входной маски. Бит разрешение внутреннего прерывания по переполнению формата чисел с плавающей запятой снизу обнуляется. Состояние разряда слежения T не изменяется. В счетчик инструкций PC заносится содержимое операнда-приемника DST, к которому прибавляется 2. В результате управление передается к байту, следующему непосредственно за входной маской.

Примечания:

1. Если содержимое разрядов 13:12 входной маски не равно нулю, то возникает ошибка резервного операнда.

2. При возникновении ошибки резервного операнда содержимое кодов условий является непредсказуемым.

3. Для вызова процедуры и условного управления при сохранении содержимого регистров используют определенные

соглашения. Например, регистры R0 и R1 всегда могут быть использованы для значений функций возврата и никогда не отражаются во входной маске. Все используемые в процедуре регистры с R2 по R11 должны быть помечены во входной маске.

CALLS вызов процедуры стекового вида

Формат:

код операции NUMERG.RL, DST.AB

Исключительная ситуация:

резервный операнд

Коды условий:

N <- 0;

Z <- 0;

V <- 0;

C <- 0;

Исключительная ситуация:

резервный операнд

Коды операций:

FB CALLS вызов процедуры стекового вида

Описание.

Содержимое операнда NUMERG заносится в стек в виде длинного слова (нулевой байт содержит число аргументов; старшие 24 разряда используются системным программным обеспечением). Значение указателя стека запоминается во внутреннем регистре. Затем разряды 1:0 указателя стека обнуляются. Таким образом, производится выравнивание стека по границе длинного слова. Содержимое входной маски процедуры

последовательно опрашивается от разряда 0 до разряда 11, и содержимое регистров, номера которых соответствуют установленным в маске разрядам, заносится в стек. В стек в виде длинных слов заносится содержимое регистров PC, FP и AP. Коды условий обнуляются. В стек заносится длинное слово, которое состоит из сохраненных в разрядах 31:30 двух младших разрядов указателя стека, единицы в разряде 29 и нуля в разряде 28, младших 12 разрядов входной маски процедуры в разрядах 27:16 и слова состояния процессора с обнуленным разрядом слежения в разрядах 15:0. Затем в стек заносится длинное слово с нулевым содержимым. В регистр FP заносится содержимое указателя стека. В регисто AP заносится содержимое указателя стека, которое было после занесения в стек числа аргументов. Разрешение внутренних прерываний в слове состояния процессора устанавливается в указанное состояние. Разрешение внутренних прерываний по целочисленному и десятичному переполнению устанавливается в соответствии с содержимым разрядов 14 и 15 входной маски. Разрешение внутреннего прерывания по переполнению для чисел с плавающей запятой снизу обнуляется. Состояние разряда T не изменяется. В счетчик инструкций PC заносится содержимое операнда-приемника DST, к которому прибавляется 2. В результате, управление передается к байту, следующему непосредственно за входной маской.

Примечания:

1. Если содержимое разрядов 13:12 входной маски не равно нулю, то возникает ошибка резервного операнда.

2. При возникновении ошибки резервного операнда содержимое кодов условий является непредсказуемым.

3. Обычно список аргументов заносится в стек в обратном порядке перед выполнением инструкции вызова процедуры. При возврате список аргументов автоматически удаляется из стека.

4. Стандартные средства вызова процедуры при сохранении содержимого регистров используют некоторые правила. Например, регистры R0 и R1 всегда могут быть использованы для возврата значений функций и никогда не помечаются во входной маске. Все используемые в процедуре регистры с R2 по R11 должны быть помечены во входной маске.

RET возврат из процедуры

Формат:

код операции

Коды условий:

N <- TMP1<3>;

Z <- TMP1<2>;

V <- TMP1<1>;

C <- TMP1<0>;

Исключительная ситуация:

резервный операнд

Коды операций:

04 RET возврат из процедуры

Описание.

В указатель стека SP заносится содержимое регистра FP,

к которому прибавляется 4. Длинное слово, содержащее величину выравнивания стека в разрядах 31:30, индикатор типа инструкции вызова в разряде 29, младшие 12 разрядов входной маски процедуры в разрядах 27:16 и сохраненное значение слова состояния процессора в разрядах 15:0 удаляются из стека и запоминается во внутреннем регистре TMP1. В регистры PC, FP и AP из стека заносится соответствующее содержимое длинных слов. Маска восстановления регистров формируется из разрядов 27:16 внутреннего регистра. Последовательно опрашивая разряды маски восстановления с 0 по 11, производится копирование содержимого из стека в регистры, номера которых соответствуют установленным в маске разрядам. К содержимому указателя стека добавляется значение разрядов 31:30 внутреннего регистра. Если разряд 29 во внутреннем регистре равен единице (это указывает на то, что вызов производится по инструкции CALLS), то длинное слово, содержащее число аргументов, удаляется из стека. Значение младшего байта (без знака) этого длинного слова умножается на 4 и прибавляется к содержимому указателя стека.

Примечания:

1. Ошибка резервного операнда возникает, если содержимое разрядов <15:8> промежуточного регистра не равны нулю.

2. При возникновении ошибки резервного операнда состояние кодов условий является непредсказуемым.

3. Состояние разряда 28 внутреннего регистра не влияет на выполнение инструкции.

4. При процедурах вызова и условного управления подра-

зумевается, что процедуры, которые возвращают значение функции или код состояния, делают это через регистр R0 или через регистры R0 и R1.

4.7. Инструкции различного назначения

В подразделе описываются следующие инструкции:

- сброс разрядов в слове состояния процессора BICPSW MASK.RW;
- установка разрядов в слове состояния процессора BISPSW MASK.RW;
- ловушка отладчика BPT [-(KSP).W];
- останов HALT [-(KSP).W];
- вычисление индекса INDEX SUBSCRIPT.RL, LOW.RL, HIGH.RL, SIZE.RL, INDEXIN.RL, INDEXOUT.WL;
- пересылка из длинного слова состояния процессора MOVPSL DST.WL;
- отсутствие операции NOP;
- извлечение общих регистров из стека POPR MASK.RW, [(SP)+.R*];
- загрузка общих регистров в стек PUSHR MASK.RW, [-(SP).W*];
- вызов специальной (микропрограммной) функции XFC (неопределенные операнды).

BICPSW сброс разрядов в слове состояния процессора

Формат:

код операции: MASK.RW

Коды условий:

N <- N AND (NOT MASK<3>);

Z <- Z AND (NOT MASK<2>);

V <- V AND (NOT MASK<1>);

C <- C AND (NOT MASK<0>);

Исключительная ситуация:

резервный операнд

Коды операций:

B9 BICPSW сброс разрядов в слове состояния процессора

Описание.

Содержимое слова состояния процессора логически умножается на инвертированное значение операнда-маски MASK. Результат заносится в слово состояния процессора.

Примечание. Ошибка резервного операнда возникает, если разряды <15:8> маски не равны нулю. При возникновении ошибки резераного операнда содержимое слова состояния процессора не изменяется.

BISPSW установка разрядов в слове состояния процессора

Формат:

код операции: MASK.RW

Коды условий:

N <- N .OR. MASK<3>);

Z <- Z OR MASK<2>);

V <- V OR MASK<1>);

C <- C OR MASK<0>);

Исключительная ситуация:

резервный операнд

Коды операций:

B8 BISPSW установка разрядов в слове состояния процессора

Описание.

Содержимое слова состояния процессора логически складывается с содержимым операнда-маски MASK, а результат заносится в слово состояния процессора.

Примечание. Ошибка резервного операнда возникает, если разряды <15:8> маски не равны нулю. При возникновении ошибки резервного операнда содержимое слова состояния процессора не изменяется.

BPT ловушка отладчика

Формат:

код операции

Коды условий:

N <- 0;

Z <- 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует

Коды операций:

03 BPT ловушка отладчика

Описание.

Подробное описание действия контрольного процессора, выполняемых по данной инструкции, приведены в документе [1].

HALT останов

Формат:

код операции

Коды условий:

```
N <- 0; !в случае возникновения ошибки привилегирован-
Z <- 0; !ной инструкции коды условий сбрасываются. Со-
V <- 0; !храненное в стеке длинное слово состояния
C <- 0; !процессора содержит состояние кодов условий
        !перед выполнением инструкции HALT

N <- N; !если процессор остановлен
Z <- Z;
V <- V;
C <- C;
```

Исключительная ситуация:

привилегированные инструкции

Коды операций:

00 HALT останов

Описание.

Подробное описание функционирования данной инструкции приведено в [1]. Если процесс выполняется в режиме ядра, то происходит останов поцессора. В противном случае возникает ошибка привилегированной инструкции.

Примечание. Код операции выбран равным нулю, чтобы фиксировать большинство случаев ошибочной передачи управления в область данных.

INDEX вычисление индекса

Формат:

код операции SUBSCRIPT.RL, LOW.RL, HIGH.RL,
SIZE.RL, INDEXIN.RL, INDEXOUT.WL

Коды условий:

N <- INDEXOUT LSS 0;

Z <- INDEXOUT EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

нарушение индексного диапазона

Коды операций:

0A INDEX вычисление индекса

Описание.

Содержимое операнда-индекса INDEXIN складывается с содержимым операнда SUBSCRIPT. Полученная сумма умножается

на содержимое операнда-размера SIZE и заносится по адресу операнда-выходного индекса INDEXOUT. Если содержимое операнда SUBSCRIPT меньше содержимого операнда LOW или больше содержимого операнда HIGH, то происходит внутреннее прерывание по нарушению индексного диапазона.

Примечания:

1. В результате выполнения этой инструкции не могут возникнуть никакие исключительные ситуации арифметического типа, кроме нарушения индексного диапазона. Таким образом, не предусмотрено никакой индикации в тех случаях, если в результате сложения или умножения происходит переполнение. Если переполнение возникает при сложении, то полученная сумма будет содержать 32 младших разряда истинного результата. Если переполнение возникает при умножении, то по адресу INDEXOUT заносятся 32 младших разряда истинного произведения содержимого суммы и содержимого операнда SUBSCRIPT. При нормальном использовании этой инструкции, переполнение не может наступить без предварительного нарушения границы индексного диапазона.

2. Эта инструкция полезна при вычислении индекса для массивов с данными фиксированной длины (целые или числа с плавающей запятой), а также для вычисления индекса для массивов полей переменной длины, символьных и десятичных строк. Наличие операнда INDEXIN обеспечивает последовательное использование инструкции для многоразмерных массивов. Для одноразмерных массивов битовых полей переменной длины он также позволяет введение постоянной части при вычислении

индекса, что не обеспечивается обычной адресной арифметикой.

Например, операторы языка ФОРТРАН

```
INTEGER*4 A(L1:U1, L2:U2), I, J
A(I,J) = 1
```

будут транслироваться в виде

```
INDEX J, #L2, #U2, #M1, #0, R0;
```

где M1=U1-L1+1

```
INDEX I, #L1, #U1, #1, R0, R0;
MOVL  #1, A-A[R0];
```

где a=((L2*M1)+L1)*4

MOVPSL пересылка длинного слова состояния процессора

Формат:

код операции: DST.WL

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

DC MOVPSL пересылка длинного слова состояния процессора

Описание.

Содержимое длинного слова состояния процессора пересылается по адресу операнда-приемника DST.

NOP отсутствие операции

Формат:

код операции

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

01 NOP отсутствие операции

Описание.

Не производится никакой операции.

POPR извлечение общих регистров из стека

Формат:

код операции MASK.RW

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

BA POPR изалечение общих регистров из стека

Описание.

Длинные слова, представляющие собой запомненное ранее в стеке содержимое регистров, извлекаются из стека и заносятся в регистры, номера которых соответствуют разрядам, помеченным единицами в операнде-маске MASK. Содержимое маски поразрядно опрашивается от разряда 0 до разряда 14. Разряд 15 игнорируется.

PUSHR загрузка общих регистров в стек

Формат:

код операции MASK.RW

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует.

Коды операций:

BB PUSHR загрузка общих регистров в стек

Описание.

Содержимое регистров, номера которых соответствуют помеченным единицами разрядам в операнде-маске MASK, зано-

сится в стек в виде длинных слов. Содержимое маски поразрядно опрашивается от разряда 14 до разряда 0. Разряд 15 игнорируется.

Примечание. Порядок занесения в стек определяется таким образом, что содержимое регистров с большими номерами запоминается в ячейках памяти с более старшими адресами. Такой порядок обеспечивает, например, правильность занесения в стек длинных чисел формата с плавающей запятой расположенных последовательно в регистрах.

XFC вызов специальной (микропрограммной) функции

Формат:

код операции

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

FC XFC вызов специальной (микропрограммной) функции

Описание.

Для того, чтобы понять, как функционирует эта инструкция, необходимо прочитать [1, раздел 2]. Эта инструкция предназначена для расширения основного набора инструкций за счет инструкций, определенных пользователем.

4.8. Инструкции для работы с очередями

Очередь представляет собой циклический двунаправленный список. Элемент очереди определяется своим адресом. Каждый элемент очереди связан с соседними элементами с помощью пары длинных слов. Первое длинное слово является прямой связью - оно определяет ячейку, в которой расположен последующий элемент. Второе двойное слово является обратной связью - оно определяет ячейку, в которой расположен предыдущий элемент. Предусмотрены два различных типа связи: абсолютная и относительная. При использовании абсолютной связи определяется абсолютный адрес элемента. При относительной связи определяется смещение относительно текущего элемента. Очереди различаются по типу используемой связи.

4.8.1. Абсолютные очереди

В абсолютных очередях используются абсолютные адреса в качестве связей. Элементы очередей связаны с помощью пары длинных слов.

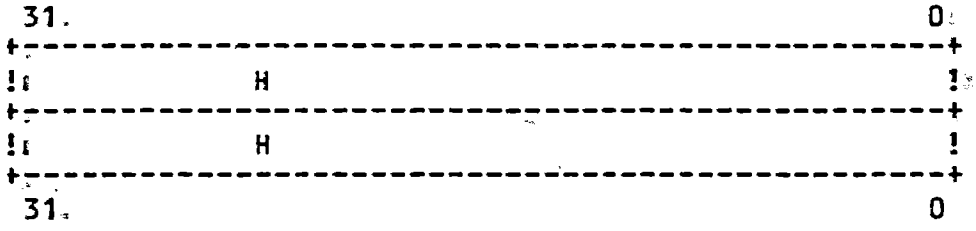
Первое (с младшим адресом) длинное слово представляет собой прямую связь: адрес последующего элемента очереди. Второе (со старшим адресом) длинное слово представляет собой обратную связь: адрес предыдущего элемента очереди. Очередь определяется своим заголовком, который структурно идентичен паре длинных слов связи. Прямая связь заголовка определяет адрес элемента, который называется началом очереди. Обратная связь заголовка определяет адрес элемента,

который называется концом очереди. Прямая связь конца очереди указывает на заголовок.

При работе с очередями определяются две основные операции занесения элементов и удаления элементов. В общем случае, операции занесение и удаление элементов могут производиться только в начале или конце очереди. Однако, с учетом определенных ограничений, эти операции могут производиться в произвольном месте очереди.

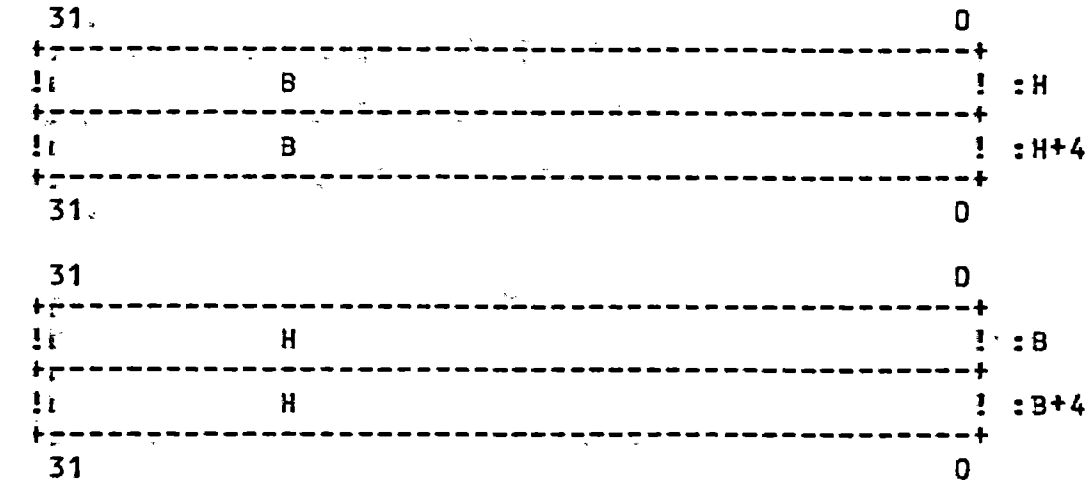
Приводятся примеры операций с очередями. Пустая очередь определяется своим заголовком по адресу "H" следующим образом:

Формат



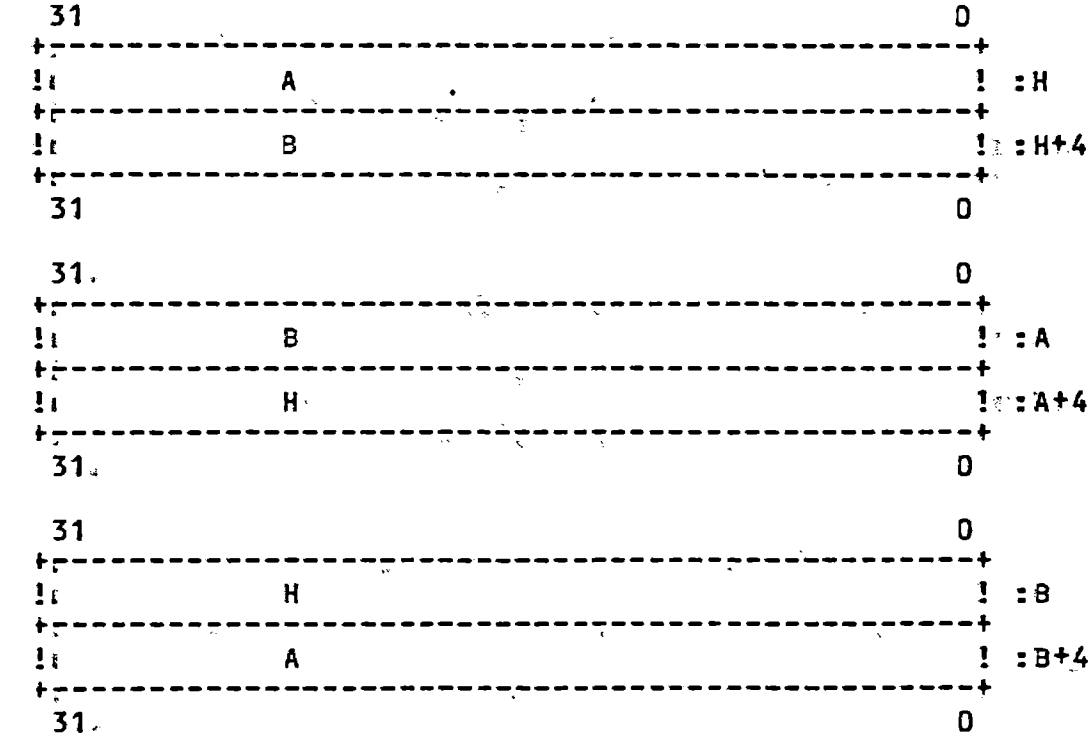
Если элемент по адресу "B" вносится в пустую очередь (либо в начало, либо в конец), то очередь будет выглядеть следующим образом:

Формат



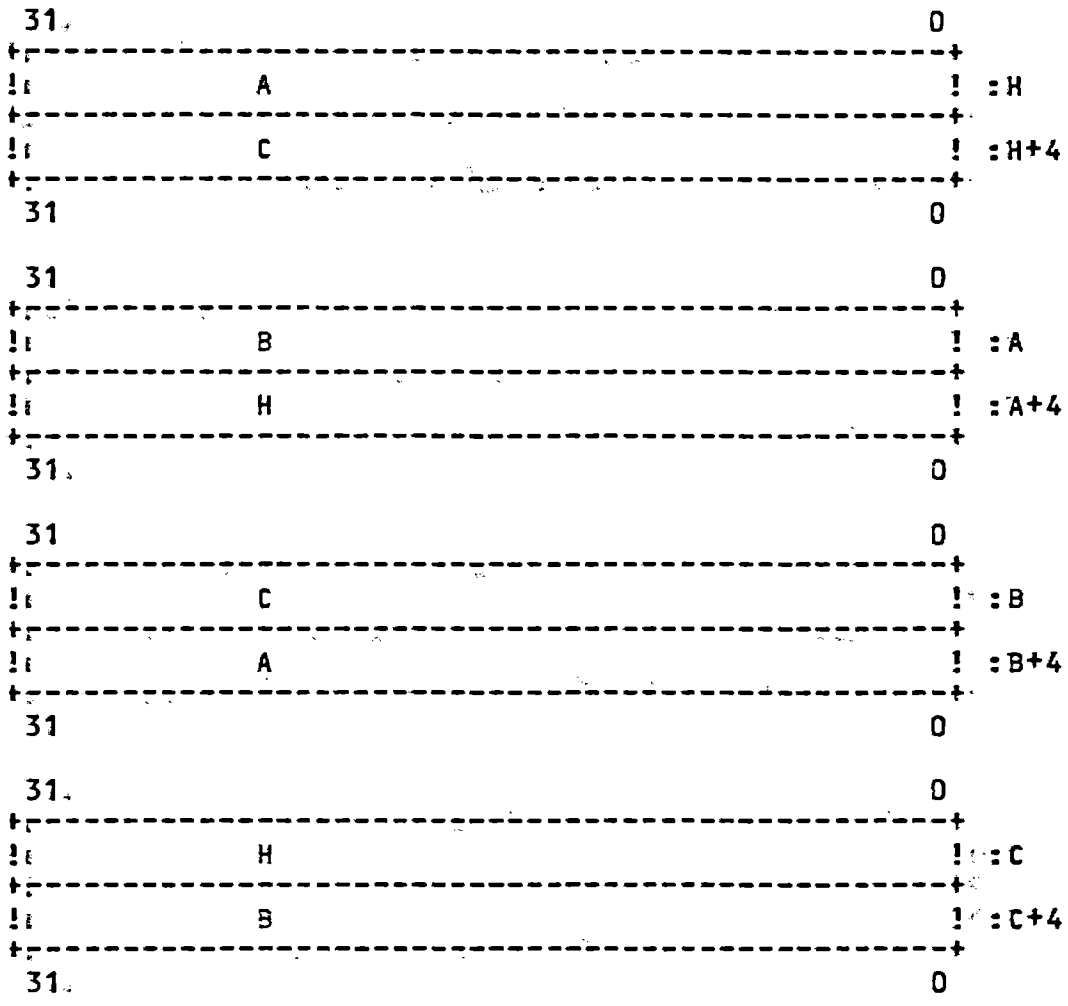
Если элемент по адресу "A" заносится в начало очереди, то очередь будет выглядеть следующим образом:

Формат.



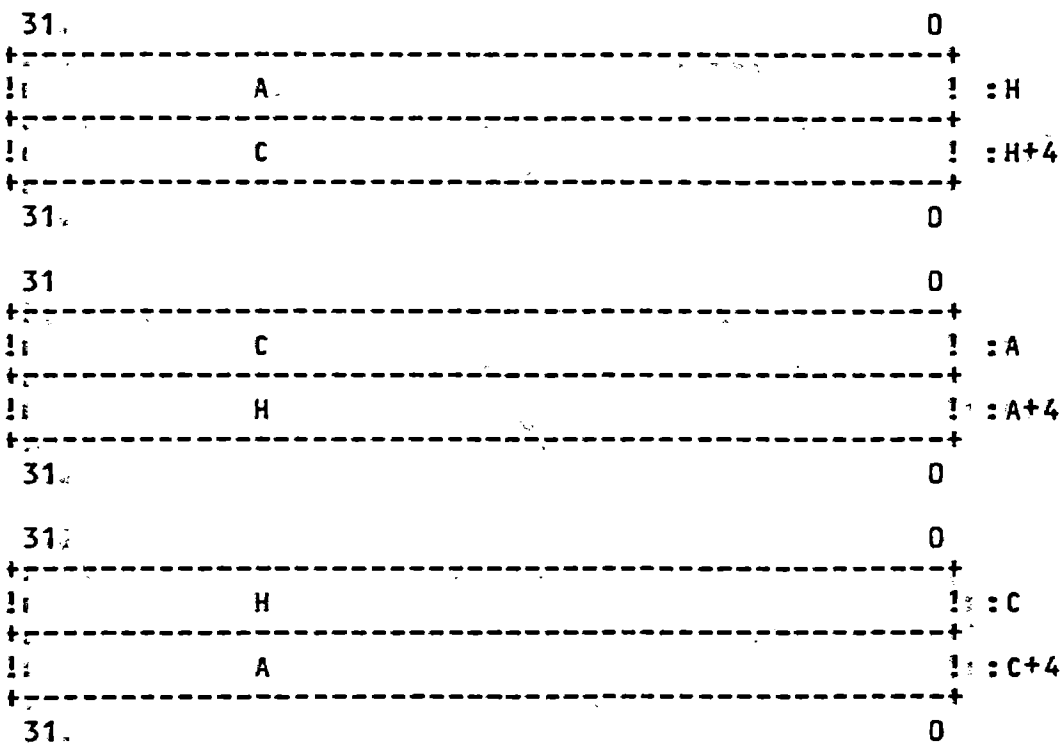
И если элемент по адресу "C" заносится в конец очереди, то очередь принимает следующий вид:

Формат



Если несколько процессов одновременно производят операции с очередью, то занесение и удаление элементов может производиться только в начало или конец очереди. В противном случае, операции можно производить в произвольном месте очереди. Если из очереди, которая показана на приведенном примере и содержит элементы A, B и C, удалить элемент по адресу "B", то очередь примет вид:

Формат



Причина ограничений заключается в том, что операции, которые производятся в начале или конце очереди, всегда будут верными, так как всегда сохраняется заголовок очереди. Операции в произвольном месте очереди зависят от конкретных имеющихся элементов и могут оказаться неверными, если другой процесс в это же время производит операции с этой очередью.

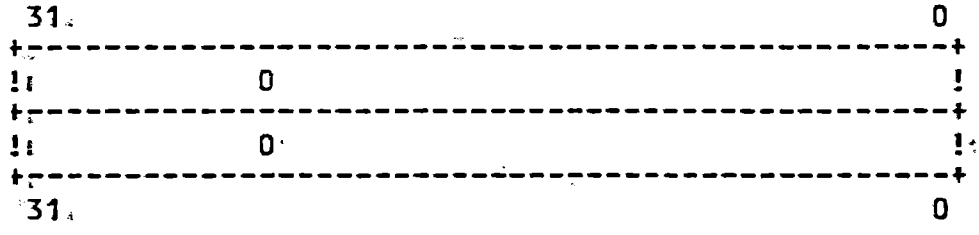
Для работы с абсолютными очередями предусмотрены две инструкции: INSQUE и REMQUE. Первая инструкция заносит в очередь элемент. Вторая инструкция удаляет из очереди элемент. Обе инструкции являются непрерываемыми инструкциями.

4.8.2. Относительные очереди

В относительных очередях в качестве связей используются смещения элементов относительно друг друга. Элементы очереди связаны с помощью пары длинных слов. Первое длинное слово (с младшим адресом) является прямой связью-смещением последующего элемента относительно текущего. Второе длинное слово (со старшим адресом) является обратной связью-смещением предыдущего элемента очереди относительно текущего. Очередь определяется своим заголовком, который также состоит из двух длинных слов связи.

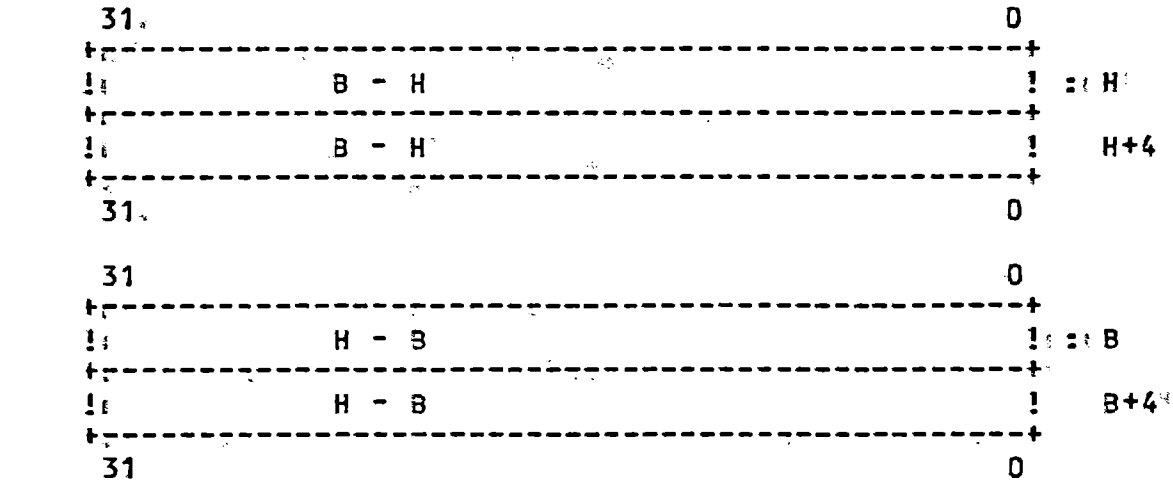
Примеры операций с очередями. Пустая очередь определяется своим заголовком по адресу "Н". Так как очередь пуста, то относительные связи должны быть равны нулю, как это показано:

Формат



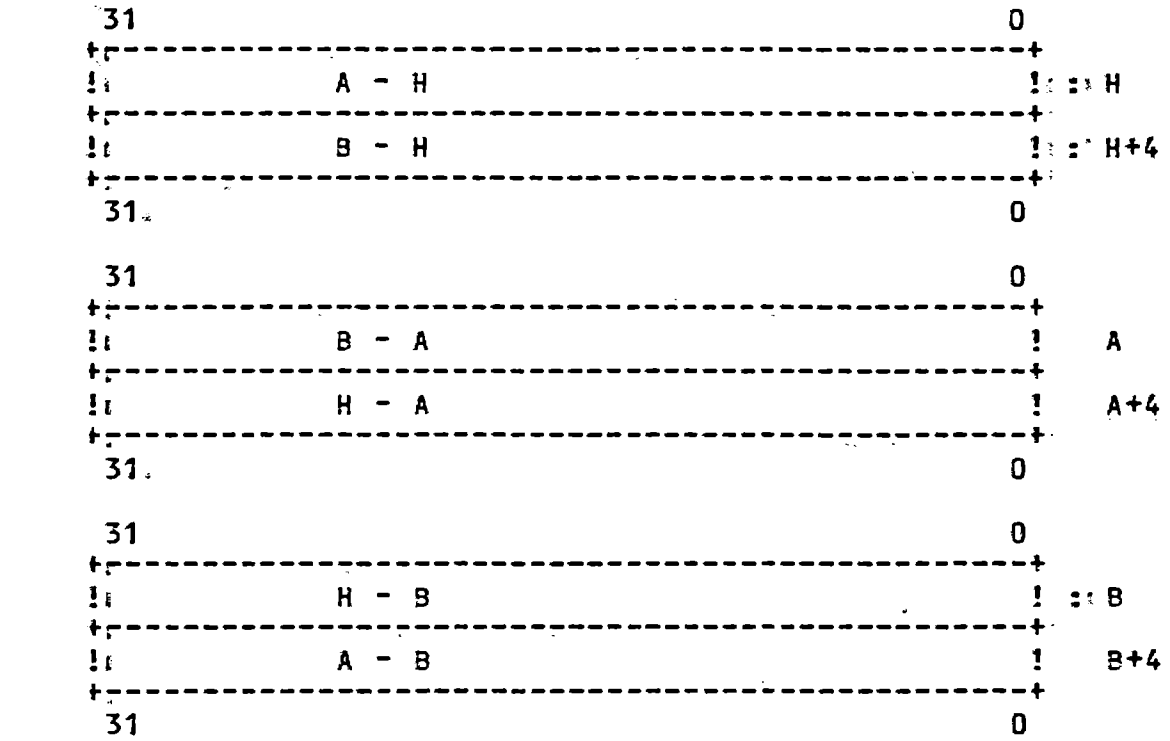
Если элемент по адресу "З" зачосится в пустую очередь (в начало или конец), то очередь примет вид:

Формат



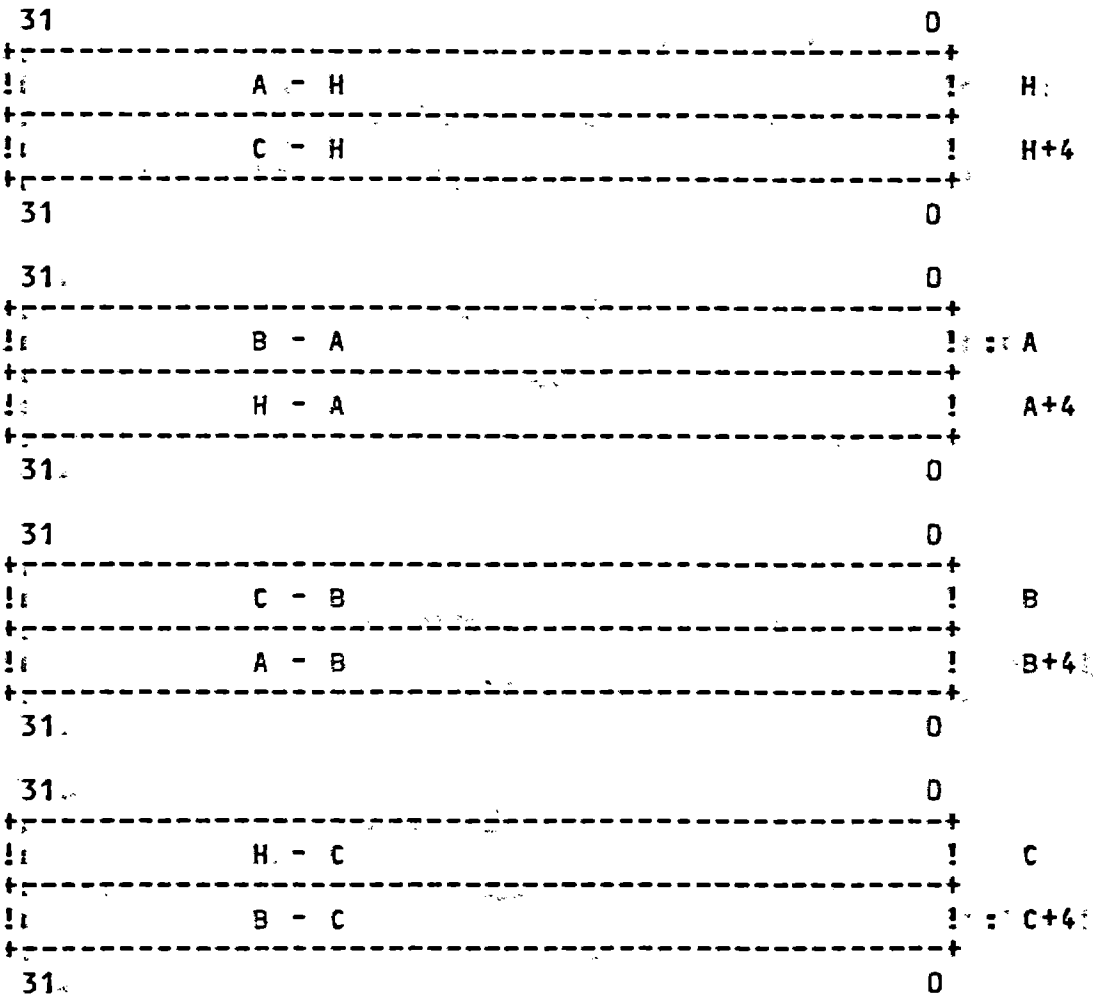
Если в начало очереди заносится элемент по адресу "A", то очередь примет вид:

Формат



И, наконец, если в конец очереди заносится элемент по адресу "C", то очередь примет вид:

Формат



Выполнение указанных операций в обратном порядке приведет к удалению элементов из начала и конца очереди.

Над относительными очередями могут быть произведены четыре операции: занесение элемента в начало очереди, занесение элемента в конец очереди; удаление элемента из начала очереди, удаление элемента из конца очереди.

Эти операции производятся с блокировкой памяти для того, чтобы обеспечить взаимодействие процессов в мультипроцессорных системах и доступ к общему списку без введения дополнительной синхронизации. Элементы очереди должны быть

выровнены по границе квадрослова. Для чтения заголовка очереди используется реализованный аппаратно механизм доступа к памяти с блокировкой. Разряд 0 заголовка очереди используется в качестве вторичной блокировки и устанавливается на то время, пока производится обращение к очереди. Если при исполнении инструкции для работы с очередями и с блокировкой памяти установлена вторичная блокировка, то действие инструкции заключается в установке кодов условий, которые фиксируют неудачу. При попытке получить доступ к очереди, если вторичная блокировка не установлена, такая инструкция устанавливает ее на время своей работы и сбрасывает ее по окончанию выполнения инструкции. Таким образом, блокируются операции других инструкций над этой же самой очередью.

4.8.3. Описание инструкций

В данном пункте описаны следующие инструкции:

- занесение элемента в начало очереди (с блокировкой памяти) INSQHI ENTRY.AB, HEADER.AG;
- занесение элемента в конец очереди (с блокировкой памяти) INSQTI ENTRY.AB, HEADER.AG;
- занесение элемента в очередь INSQUE ENTRY.AB, PRED.AB;
- удаление элемента из начала очереди (с блокоровкой памяти) REMQHI HEADER.AG, ADDR.WL;
- удаление элемента из конца очереди (с блокировкой памяти) REMQTI HEADER.AG, ADDR.WL;
- удаление элемента из очереди REMQUE ENTRY.AB, ADDR.WL.

INSQHI занесение элемента в начало очереди (с блокировкой памяти)

Формат:

код операции ENTRY.AB, HEADER.AQ

Коды условий:

если (занесение произошло), то:

N = 0

Z = (ENTRY) EQL (ENTRY + 4) ! первый элемент
! очереди

V = 0

C = 0

в противном случае

N = 0

Z = 0

V = 0

C = 1 ! нарушение вторичной блокировки

Исключительная ситуация:

резервный операнд

Коды операций:

5C INSQHI занесение элемента в начало очереди (с блокировкой памяти)

Описание.

Элемент, определяемый операндом-элементом ENTRY, заносится в очередь вслед за заголовком HEADER.

Если занесенный элемент был первым в очереди, то устанавливается разряд Z кодов условий, в противном случае - он

сбрасывается. Занесение сопровождается блокировкой памяти для того, чтобы предотвратить параллельные операции с той же очередью даже в мультипроцессорной системе. Перед тем, как выполнить любую часть операции, процессор проверяет возможность выполнения всей операции в целом. Таким образом обеспечивается сохранение целостности очереди при возникновении исключительных ситуаций при управлении памятью. Если установлена вторичная блокировка, то инструкция производит соответствующую установку кодов условий и на этом заканчивает свои операции.

Примечания:

1. Так как занесение является непрерываемой операцией, то процесс, работающий в режиме ядра, может использовать очередь параллельно с программами обработки прерываний.

2. Инструкции INSQHI, INSQTI, REMQHI и REMQTI реализованы таким образом, что взаимодействующие в мультипроцессорной системе программные процессы могут совместно использовать очереди без дополнительной синхронизации.

3. Для того, чтобы реализовать программную блокировку обращения к очереди, можно использовать следующую последовательность инструкций:

```
INSERT:     INSQHI              очередь пустая?
            BEQL  1¤            да
            BCS.  INSERT        снова сделать попытку
                                занесения
            CALL WAIT(...)      нет, ожидание
1¤:
```

4. В течение проведения проверки правильности доступа любой доступ, который не может быть завершен, приводит к исключительной ситуации управления памятью, даже если занесение в очередь еще не началось.

5. Если элемент или заголовок не выровнен по границе квадрослова, то есть разряды <2:0> адреса не равны нулю, или разряды <2:1> заголовка не равны нулю, то происходит ошибка резервного операнда. Кроме того, ошибка резервного операнда возникает, если заголовок HEADER равен элементу ENTRY. В этом случае очередь не изменяется.

INSQTI занесение элемента в конец очереди (с блокировкой памяти)

Формат:

код операции: ENTRY.AB, HEADER.AQ

Коды условий:

если (занесение произошло), то

N = 0

Z = (ENTRY) .EQL (ENTRY + 4) ;первый элемент ;очереди

V = 0

C = 0

в противном случае

N = 0

Z = 0

V = 0

C = 1. ;нарушение вторичной блокировки

Исключительная ситуация:

резервный операнд

Коды операций:

5D INSQTI занесение элемента в конец очереди (с блокировкой памяти)

Описание.

Элемент, определяемый операндом-элементом ENTRY, заносится в конец очереди HEADER. Если занесенный элемент был первым в очереди, то устанавливается разряд Z кодов условий, в противном случае - он сбрасывается. Занесение является непрерываемой операцией. Занесение сопровождается блокировкой памяти для того, чтобы предотвратить параллельные операции с той же очередью даже в мультипроцессорной системе. Перед тем, как выполнить любую часть операции, процессор проверяет возможность выполнения всей операции в целом. Таким образом обеспечивается сохранение целостности очереди при возникновении исключительных ситуаций управления памятью. Если установлена вторичная блокировка, то инструкция производит соответствующую установку кодов условий и на этом заканчивает свои операции.

Примечания:

1. Так как занесение является непрерываемой операцией, то процесс, работающий в режиме ядра, может использовать очередь параллельно с программами обработки прерываний.

2. Инструкции INSQHI, INSQTI, REMQHI и REMQTI реализованы таким образом, что взаимодействующие в мультипроцес-

сорной системе программные процессы могут совместно использовать очереди без дополнительной синхронизации.

3. Для того, чтобы реализовать программную блокировку обращения к очереди, можно использовать следующую последовательность инструкций:

```
INSERT: INSQHI  ...             ;очередь пуста?
        BEQL    1$              ;да
        BCS INSERT              ;снова сделать попытку
                                ;занесения
        CALL    WAIT            ;нет, ожидание
1$:
```

4. В течение проведения проверки правильности доступа любой доступ, который не может быть завершен, приводит к исключительной ситуации управления памятью, даже если занесение в очередь еще не началось.

5. Если элемент или заголовок не выравнен по границе учетверенного слова (т.е. разряды <2:0> адреса не равны нулю, или разряды <2:1> заголовка не равны нулю), то происходит ошибка резервного операнда. Кроме того, ошибка резервного операнда возникает, если заголовок HEADER равен элементу ENTRY. В этом случае очередь не изменяется.

INSQUE занесение элемента в очередь

Формат:

код операции ENTRY.AB, READ.AB

Коды условий:

N<-(ENTRY)LSS (ENTRY + 4)

Z<-(ENTRY) EQL (ENTRY + 4) ! первый элемент

! в очереди

V<-0

C<-(ENTRY) LSSU (ENTRY + 4)

Исключительная ситуация:

отсутствует

Коды операций:

0E INSQUE занесение элемента в очередь

Описание.

Элемент, определяемый операндом-элементом ENTRY, заносится в очередь вслед за элементом, определяемым операндом-предшествующим элементом READ. Если внесенный элемент является первым в очереди, то разряд Z кодов условий устанавливается, в противном случае - сбрасывается. Занесение является непрерываемой операцией. Перед тем, как выполнить любую часть операции, процессор проверяет возможность выполнения всей операции в целом. Таким образом обеспечивается сохранение целостности очереди при возникновении исключительных ситуаций управления памятью.

Примечания:

1. Путем соответствующего выбора предшествующего элемента можно выполнить три типа занесения в очередь:

1) занесение в начало очереди

INSQUE ENTRY,H ;H - заголовок очереди;

2) занесение в конец очереди

INSQUE ENTRY,@H + 4 ;H - заголовок очереди (отметим, что косвенная адресация используется только в этом случае);

3) занесение в произвольное место очереди

INSQUE ENTRY,P ;P - предшествующий элемент.

2. Так как занесение является непрерываемой операцией, то процессы, действующие в режиме ядра могут использовать очередь параллельно с программами обработки прерываний.

3. Инструкции INSQUE и REMQUE реализованы таким образом, что взаимодействующие программные процессы в однопроцессорной системе могут совместно использовать очереди без дополнительной синхронизации, если занесение и удаление элемента производится только в начало или конец очереди.

4. Для того, чтобы реализовать программную блокировку обращения к очереди, можно использовать следующую последовательность инструкций:

```
INSQUE  ...             ;очередь пуста?
BEQL    1¤              ;да
CALL    WAIT(...)       ;нет, ожидание
1¤:
```

5. В течение проведения проверки правильности доступа любой доступ, который не может быть завершен, приводит к исключительной ситуации управления памятью, даже если занесение в очередь еще не началось.

REMQHI удаление элемента из начала очереди (с блокировкой памяти)

Формат:

код операции: HEADER.AQ, ADDR.WL

Коды условий:

если (удаление произошло), то

N=0

Z=(HEADER) EQL 0! очередь пуста

V=TMP1 EQL 0 !отсутствует удаляемый элемент

C=0

в противном случае

N = 0

Z = 0

V = 1 ! ничего не удалено

C = 1 !нарушение вторичной блокировки

Исключительная ситуация:

резервный операнд

Коды операций:

5E REMQHI удаление элемента из начала очереди (с блокировкой памяти)

Описание.

Элемент очереди, следующий за заголовком HEADER, удаляется из очереди. Содержимое операнда-адреса ADDR заменяется адресом удаленного элемента. Если элемент не был удален из очереди (вследствие отсутствия элементов или

нарушения установки вторичной блокировки), то устанавливается разряд "V" кодов условий, в противном случае - он сбрасывается. Если блокировка произведена, и в конце выполнения инструкции очередь пуста, то устанавливается разряд Z кодов условий, в противном случае он сбрасывается. Удаление из очереди производится с блокировкой памяти для того, чтобы предотвратить параллельные операции над началом или концом той же очереди другим процессором в мультипроцессорной системе. Удаление из очереди является непрерываемой операцией. Перед тем, как выполнить любую часть операции, процессор производит проверку возможности выполнения всей операции в целом. Таким образом обеспечивается сохранение целостности очереди даже если возникает исключительная ситуация управления памятью. Если инструкция встречает вторичную блокировку, то производится установка кодов условий и операция заканчивается без изменения очереди.

Примечания:

1. Так как удаление из очереди является непрерываемой операцией, то процесс, работающий в режиме ядра, может использовать очередь совместно с программами обработки прерываний.

2. Инструкции INSQHI, INSQTI, REMQHI и REMQTI реализованы таким образом, что взаимодействующие в мультипроцессорной системе программные процессы могут совместно использовать очереди без дополнительной синхронизации.

3. Для того, чтобы реализовать программную блокировку обращения к очереди, можно использовать следующую последо-

вательность инструкций:

```
1¤:  REMQHI   ...        удален последний элемент?
     BEQL     2¤         да
     BCS      1¤         снова сделать попытку удаления
     CALL     ACTIVATE(...) ; активизировать другие
                        процессы, ожидающие доступа
                      ; к очереди.
```

4. Для того, чтобы произвести последовательное удаление элементов до момента, когда очередь станет пустой, можно использовать следующую последовательность:

```
1¤:  REMQHI              есть еще элементы в очереди?
     BVS      2¤         нет
```

Процесс удаления элемента.

```
     BR       1¤
      .
2¤:. BCS      1¤         сделать снова попытку удаления
                         элемента
```

Очередь пуста.

5. В течение проведения проверки правильности доступа любой доступ, который не может быть завершен, приводит к исключительной ситуации управления памятью, даже если удаление из очереди еще не начиналось.

6. Ошибка резервного операнда имеет место, если заголовок или (заголовок+(заголовок)) является адресом, который не выравнен по границе учетверенного слова (т.е. разряды

<2:0> адреса не равны нулю), или если разряды (заголовок) <2:1> не равны нулю. Ошибка резервного операнда также возникает, если операнд адреса заголовка равен адресу второго операнда. В этом случае очередь не изменяется.

REMQTI удаление элемента из конца очереди (с блокировкой памяти)

Формат:

код операции HEADER.AQ, ADDR.WL

Коды условий:

если (удаление произошло), то

N = 0

Z = (HEADER + 4) EQL 0 ! очередь пуста

V = TMP1 EQL 0 ! отсутствует удаляемый ! элемент

C = 0

в противном случае:

N = 0

Z = 0

V = 1 ! ничего не удалено

C = 1 ; вторичная блокировка не произошла

Исключительная ситуация:

резервный операнд

Коды операций:

5F REMQTI удаление элемента из конца очереди (с блокировкой памяти)

Описание.

Элемент очереди, предшествующий заголовку, удаляется из очереди. Содержимое операнда-адреса ADDR заменяется адресом удаленного элемента. Если элемент не был удален из очереди (вследствие отсутствия элементов или нарушения установки вторичной блокировки), то устанавливается разряд V кодов условий, в противном случае - он сбрасывается. Если блокировка произведена, и в конце выполнения инструкции очередь пуста, то устанавливается разряд Z кодов условий, в противном случае он сбрасывается. Удаление из очереди производится с блокировкой памяти для того, чтобы предотвратить параллельные операции над началом или концом той же очереди другим процессом даже в мультипроцессорной системе. Удаление из очереди является непрерываемой операцией. Перед тем, как выполнить любую часть операции, процессор производит проверку возможности выполнения всей операции в целом. Таким образом обеспечивается сохранение целостности очереди, даже если возникает исключительная ситуация управления памятью. Если инструкция встречает вторичную блокировку, то производится установка кодов условий и операция заканчивается без изменения очереди.

Примечания:

1. Так как процесс удаления элемента из очереди является непрерываемой операцией, то процесс, работающий в режиме ядра, может использовать очередь совместно с программами обработки прерываний.

2. Инструкции INSQHI, INSQTI, REMQHI и REMQTI реализо-

ваны таким образом, что взаимодействующие в мультипроцессорной системе программные процессы могут совместно использовать очереди без дополнительной синхронизации.

3. Для того, чтобы реализовать программную блокировку обращения к очереди, можно использовать следующую последовательность инструкций:

```
1¤:  REMQTI   ...          удален последний элемент?
     BEQL     2¤           да
     BCS      1¤           снова сделать попытку удаления
     CALL ACTIVATE(...)    активизировать другие процессы,
                            ожидающие доступа очереди
2¤:
```

4. Для того, чтобы произвести последовательное удаление элементов до момента, когда очередь станет пустой, можно использовать следующую последовательность:

```
1¤:  REMQTI   ...          есть еще элементы в очереди?
     BCS      2¤           нет
     ...
     Процесс удаления элемента

     BR       1¤

2¤:  BCS      1¤           сделать снова попытку удаления
                          ; элемента
     Очередь пуста
```

5. В течение проведения проверки правильности доступа любой доступ, который не может быть завершен, приводит к

исключительной ситуации управления памятью, даже если удаление из очереди не началось.

6. Ошибка резервного операнда имеет место, если заголовок, (заголовок+4) или (заголовок+(заголовок+4)+4) является адресом, который не выравнен по границе квадрослова (т.е. разряды <2:0> не равны 0), или если разряды (заголовок)<2:1> не равны нулю. Ошибка резервного операнда также возникает, если операнд адреса заголовка равен адресу второго операнда. В этом случае очередь не изменяется.

REMQUE — удаление элемента из очереди

Формат:

код операции ENTRY.AB, ADDR.WL

Коды условий:

```
N = (ENTRY) LSS (ENTRY + 4);
Z = (ENTRY) EQL (ENTRY + 4);    ! очередь пуста
V = (ENTRY) EQL (ENTRY + 4);    ! нет удаляемого
                                ! элемента
C = (ENTRY) LSSU (ENTRY + 4);
```

Исключительная ситуация

отсутствует

Коды операций:

OF REMQUE — удаление элемента из очереди

Описание.

Элемент, определяемый операндом-элементом ENTRY, удаляется из очереди. Содержимое операнда-адреса ADDR заменяется на адрес удаленного элемента. Если в очереди

отсутствует удаляемый элемент, то устанавливается разряд V кодов условий, в противном случае он сбрасывается. Если в конце выполнения этой инструкции очередь оказывается пустой, то устанавливается разряд Z кодов условий, в противном случае он сбрасывается. Удаление является непрерываемой операцией. Перед тем, как выполнить какую-либо часть операции, процессор проверяет возможность выполнения всей операции в целом. Таким образом, сохраняется целостность очереди при возникновении исключительной ситуации управления памятью.

Примечания:

1. Путем соответствующего выбора операнда-элемента можно производить три типа удаления из очереди:

1) удаление из начала очереди

   REMQUE @H, aDDR H - заголовок очереди

2) удаление из конца очереди

   REMQUE @H+4, aDDR H - заголовок очереди

3) удаление произвольного элемента очереди

   REMQUE ENTRY, aDDR

2. Так как удаление является непрерываемой операцией, то процесс, работающий в режиме ядра, может использовать очередь параллельно с программами обработки прерываний.

3. Инструкции INSQUE и REMQUE реализованы таким образом, что взаимодействующие программные процессы в однопроцессорной системе могут совместно использовать очередь без дополнительной синхронизации, если занесение и удаление элемента производится только в начало или конец очереди.

4. Для того, чтобы реализовать программную блокировку обращения к очереди, можно использовать следующую последовательность инструкций:

```
     REMQUE                ; очередь пуста?
     BEQL       1¤         ; да
     CALL       ACTIVATE(...)   нет, ожидание
1¤:
```

5. Для того, чтобы произвести удаление элементов до полной ликвидации очереди, можно использовать следующую последовательность инструкций:

```
1¤:  REMQUE                есть еще элементы?
     BVS        EMPTY      нет
      ...
     BR         1¤         ;
```

6. В течение проверки правильности доступа любой доступ, который не может быть завершен, приводит к исключительной ситуации управления памятью, даже если удаление из очереди еще не началось.

4.9. Инструкции для чисел с плавающей запятой

Инструкции с плавающей запятой работают с данными четырех типов, а именно со словами F_формата, D_формата, G_формата и H_формата.

Для обеспечения надежной работы с набором инструкций для чисел с плавающей запятой программы, использующие функции плавающей арифметики, должны проверять, не является ли

входной операнд (входные операнды) резервным. Простым способом проверки является выполнение инструкции пересылки или тестирования входных операндов в формате с плавающей запятой.

Для достижения большего быстродействия при работе с набором инструкций для чисел с плавающей запятой накладываются некоторые ограничения на сочетания режимов адресации, используемых в одной инструкции. К таким сочетаниям относится логически противоречивое использование данных одновременно в качестве операнда с плавающей запятой и адреса.

В частности, если в одной и той же инструкции содержимое регистра RN используется одновременно как часть входного операнда с плавающей запятой (то есть операнда .RF, .RD, .RG, .RH, .MF, .MD, .MG или .MH) и как адрес с режимом адресации, вызывающим модификацию RN, (то есть в режиме автоувеличения, автоуменьшения или в косвенном режиме с автоувеличением), значение операнда с плавающей запятой является непредсказуемым.

4.9.1. Форма представления для чисел с плавающей запятой

С математической точки зрения число с плавающей запятой может быть представлено (в виде знака, порядка и мантиссы) в форме:

(+ или -)(2**K)*F,

где K - целое (порядок);

F - неотрицательная дробная часть (или мантисса);

Числа K и F однозначно определяются условием

0,5 LEQ F LSS 1

В таком случае говорят, что мантисса F двоично нормализована. Число с плавающей запятой, равное нулю, представляется таким образом, что числу F должно быть присвоено значение 0, а значение K неопределено.

Форматы данных для инструкций плавающей арифметики см 1700 выведены из данного математического представления для чисел с плавающей запятой. Существуют четыре типа данных с плавающей запятой: два стандартных, которые использовались и в 16-ти разрядных СМ ЭВМ, это форматы F и D и два расширенных формата (форматы G и H). Данные с плавающей запятой одинарной точности (число с плавающей запятой F_формата) имеют разрядность 32. Данные двойной точности (число с плавающей запятой D_формата) представлены 64 разрядами. Данные двойной точности в расширенном формате (число с плавающей запятой G_формата), занимают 64 разряда. Данные учетверенной точности в расширенном формате (число с плавающей запятой H_формата) занимают 128 разрядов.

Используются следующие представления чисел с плавающей запятой:

1) ненулевые числа с плавающей запятой:

Старший разряд данных является знаковым: 0, если число положительное, и 1, если отрицательное.

Считается, что дробная часть (мантисса) нормализована,

то есть ее старший разряд должен быть равен 1. Этот разряд является "скрытым", то есть он не хранится в слове данных, а достраивается аппаратно перед выполнением арифметической операции. В форматах данных F и D под дробную часть отводится соответственно 23 и 55 разрядов, что вместе со скрытым разрядом дает в арифметических операциях значимость 24 и 56 разрядов. В данных расширенного формата (G и H) под дробную часть отводится 52 и 112 разрядов соответственно, что дает вместе со скрытым разрядом в арифметических операциях значимость 53 и 113 разрядов.

В форматах данных F и D для хранения порядка K отводится 8 разрядов. Таким образом, числами от 0 до 255 могут быть представлены порядки от -128 до +127. Значение порядка, равное 0 (действительное значение -128), зарезервировано число 0 плавающей арифметики. Таким образом, в форматах данных с плавающей запятой типа F и D порядка ограничены диапазоном от -127 до +127 включительно.

Для данных формата G под хранение порядка отведено 11 разрядов. В данных формата H отводится 15 разрядов под хранение порядка. Таким образом, диапазон его представления - от -1023 до +1023 включительно и от -16383 до +16383 включительно для чисел с плавающей запятой форматов G и H соответственно;

2) нуль плавающей арифметики:

Из-за скрытого разряда мантиссы нельзя отличить нуль от ненулевых величин с дробной частью, в точности равной 1/2. Поэтому для этой цели в см 1700 зарезервировано нуле-

вое поле знака и порядка. Любое положительное число с плавающей запятой, имеющее порядок 0, набором инструкций плавающей арифметики обрабатывается как точный нуль. В частности, операнд с плавающей запятой, имеющий все нулевые разряды, обрабатывается как нуль и именно этот формат вырабатывают все инструкции плавающей арифметики, результатом работы которых является нуль;

3) резервные операнды:

Резервным операндом называется любое двоичное число со знаковым разрядом 1 и нулевым порядком. В СМ 1700 все инструкции с плавающей запятой вырабатывают ошибку, если встречается резервный операнд. В результате работы инструкций с плавающей запятой резервный операнд никогда не обрабатывается.

4.9.2. Обзор набора инструкций

Для всех четырех типов данных с плавающей запятой СМ 1700 имеет стандартные арифметические операции сложения, вычитания, умножения и деления (ADD, SUB, MUL, DIV). Результаты этих операций всегда округляются (см. пункт 4.9.3). Кроме этого, имеются две составные операции EMOD и POLY, также применимые ко всем четырем типам данных с плавающей запятой. EMOD вырабатывает произведение двух операндов, а затем разделяет результат на целую и дробную части. POLY подсчитывает значение полинома по заданной степени, аргументу и указателю таблицы коэффициентов. Подробно выполнение EMOD и POLY рассмотрено в соответствующих описа-

ниях. При выполнении этих инструкций возникает ошибка округления, связанная с операциями над данными с плавающей запятой, а также в них возможно переполнение по верхней или нижней границе (переполнение сверху или снизу). Точность вычислений рассматривается в пункте 4.9.3, а исключительные ситуации описаны в документе [1].

В СМ 1700 также имеется полный набор инструкций преобразования из целочисленного типа данных (байт, слово или длинное слово) во все форматы с плавающей запятой (F, D, G и H), и наоборот. Также имеется набор инструкций преобразования из одного типа данных с плавающей запятой в другой, за исключением преобразований из формата D в G и наоборот. Многие из этих инструкций выполнятся точно в смысле, определенном в пункте 4.9.3. Однако, в некоторых из них может возникнуть ошибка округления, переполнение формата с плавающей запятой по верхней или нижней границе или они могут повлечь за собой целочисленное переполнение сверху. Подробности приведены при описании группы инструкций CVT.

Существует класс инструкций типа пересылок, которые всегда выполняются точно: MOV, NEG, CLR, CMP и TST. И, наконец, инструкция ACB (сложение, сравнение и переход), в результате которой возникают ошибки округления и переполнения.

Все инструкции с плавающей запятой в см 1700 вырабатывают ошибку по резервному операнду. Также они вырабатывают ошибку по переполнению формата с плавающей запятой по верхней или нижней границе или при делении на нуль и в этом

случае значения кодов условий непредсказуемы. Разряд FU в PSW доступен для разрешения или запрещения исключительных ситуаций по переполнению снизу. Если разряд FU сброшен, то по переполнению снизу исключительная ситуация не возникает, а результатом работы инструкции является нуль. Если разряд FU установлен, то по переполнению снизу вырабатывается ошибка. Дальнейшие подробности о действиях, выполняемых по любой из этих исключительных ситуаций, включены в описания инструкций и полностью описаны в документе [1].

4.9.3. Точность

Здесь приведены общие положения о точности выполнения набора инструкций с плавающей запятой СМ 1700. В описания конкретных инструкций могут быть включены дополнительные сведения о точности их выполнения.

Считается, что инструкция выполняется точно, если ее результат, расширенный вправо сколь угодно большой последовательностью нулей, идентичен результату вычисления над тем же операндом, выполненному с бесконечной точностью. Для всех арифметических операций, кроме DIV, нулевой операнд вызывает точное выполнение инструкции. То же утверждение справедливо для DIV, если нулевой операнд является делимым. Но если он является делителем, деление не определено, и инструкция вырабатывает ошибку.

Для ненулевых операндов с плавающей запятой мантисса двоично нормализована и занимает 24 или 56 разрядов при одинарной точности (формат F) или двойной точности (формат

D), соответственно или же 53 или 113 разрядов для расширенного формата двойной точности (формат G) или расширенного формата с учетверенной точностью (формат H) соответственно. Для ADD, SUB, MUL и DIV разряда переполнения слева и двух разрядов слежения справа необходимо и достаточно для того, чтобы была гарантирована выработка округленного результата, идентичного получаемому в соответствующей операции бесконечной точности, округленному до указанной длины в словах. Таким образом, при двух разрядах слежения максимальная ошибка округления находится в пределах 1/2 младшего разряда.

Результат арифметической операции является точным, если при усечении результата бесконечной точности до длины сохраняемых данных не теряются ненулевые разряды. Усечение означает, что запоминаются только 24 (формат F), 56 (формат D), 53 (формат G) или 113 (формат H) старших разрядов нормализованной мантиссы, остальные разряды отбрасываются. Первый отбрасываемый разряд называется "разрядом округления". Значение округленного результата связано с усечением следующим образом:

1) если разряд округления равен 1, то округленный результат есть увеличенный на 1 младшего разряда усеченный результат;

Если разряд округления равен 0, то округленный и усеченный результаты совпадают.

В процессоре СМ 1700 используется округление, дающее результаты, идентичные получаемым в следующем алгоритме.

прибавить 1 к разряду округления и выполнить распространение переноса, если он происходит. После округления может потребоваться повторная нормализация; если это происходит, новый разряд округления будет равен 0. Следующие утверждения устанавливают соответствие между усеченным, округленным и истинным (бесконечной точности) результатами:

1) если сохраняемый результат точен, то:

Округленная величина=усеченная величина=истинная величина;

2) если сохраняемый результат не точен, то его величина:

- всегда меньше истинного результата при усечении;
- всегда меньше истинного результата при округлении, если бит округления равен 0;
- больше истинного результата при округлении, если бит округления равен 1.

4.9.4. Описание инструкций

В данном пункте описаны следующие инструкции:

- двухоперандное сложение ADD(F,D,G,H)2 ADD.RX, SUM.MX;
- трехоперандное сложение ADD(F,D,G,H)3 ADD1.RX, ADD2.RX, SUM.WX;
- очистка CLR(L=F,Q=G,O=H) DST.WX;
- сравнение CMP(F,D,G,H) SRC1.RX, SRC2.RX;
- преобразование CVT(F,D,G,H)(B,W,L,D,G,H) SRC.RX, DST.WY CVT(B,W,L)(F,D,G,H) SRC.RX, DST.WY все комбина-

ции пар, за исключением пар FF,DD,GG,HH,DG AND GD;

- преобразование с округлением CVTR(F,D,G,H)L SRC.RX, DST.WL;
- деление двухоперандное DIV(F,D,G,H)2 DIVR.RX, QUO.MX;
- деление трехоперандное DIV(F,D,G,H)3 DIVR.RX, DIVD.RX, QUO.WX;
- расширенное умножение с выделением целой части EMOD(F,D) MULR.RX,MULRX.RB,MULD.RX,INT.WL,FRACT.WX EMOD(G,H) MULR.RX,MULRX.RW,MULD.RX,INT.WL,FRACT.WX;
- пересылка с отрицанием MNEG(F,D,G,H) SRC.RX, DST.WX
- пересылка MOV(F,D,G,H) SRC.RX, DST.WL;
- двухоперандное умножение MUL(F,D,G,H)2 MULR.RX, PROD.MX;
- трехоперандное умножение MUL(F,D,G,H)3 MULR.RX, MULD.RX, PROD.WX;
- вычисление полинома в формате F POLYF ARG.RF, DEGREE.RW, TBLADDR.AB, [R0-3.WL];
- вычисление полинома в формате D POLYD ARG.RD, DEGREE.RW, TBLADDR.AB, [R0-5.WL];
- вычисление полинома в формате G POLYG ARG.RG, DEGREE.RW, TBLADDR.AB, [R0-5.WL];
- вычисление полинома в формате H POLYH ARG.RH, DEGREE.RW, TBADDR.AB, [R0-5.WL, -16(SP):-1(SP).WB];
- двухоперандное вычитание SUB(F,D,G,H)2 SUB.RX, DIF.MX;
- трехоперандное вычитание SUB(F,D,G,H)2 SUB.RX, MIN.RX, DIF.WX;
- проверка TST(F,D,G,H) SRC.RX

В разделе инструкций управления описана следующая инструкция формата с плавающей запятой:

- сложение, сравнение, переход ACB(F,D,G,H): LIMIT.RX, ADD.RX, INDEX.MX, DISPL.BW

При положительном слагаемом при сравнении проверяется условие "меньше или равно", при отрицательном - "больше или равно".

ADD сложение

Формат:

код операции ADD.RX, SUM.MX 2 операнда

код операции ADD1.RX, ADD2.RX, SUM.WX 3 операнда

Коды условий:

N <- SUM LSS 0;

Z <- SUM EQL 0;

V <- (переполнение формата с плавающей запятой сверху);

C <- 0;

Исключительная ситуация:

переполнение с плавающей запятой сверху

переполнение с плавающей запятой снизу

резервный операнд

Коды операций:

40 ADDF2

41 ADDF3

60 ADDD2

61 ADDD3 сложение чисел с плавающей запятой с

40FD ADDG2 двумя или тремя операндами

41FD ADDG3

60FD ADDH2

61FD ADDH3

Описание.

В двухоперандном формате операнд-слагаемое ADD прибавляется к сумме SUM, а сумма заменяется округленным результатом. В трехоперандном формате первое слагаемое ADD1 прибавляется ко второму ADD2, а операнд-сумма SUM заменяется округленным результатом.

Примечания:

1. При ошибке по резервному операнду сумма остается неизменной, а коды условий непредсказуемы.

2. При переполнении формата с плавающей запятой снизу, если установлен разряд FU, вырабатывается ошибка. Только в том случае, если FU сброшен, результатом переполнения снизу является 0. При ошибке по переполнению снизу операнд-сумма остается неизменным. Если FU сброшен, то сумма заменяется нулем и исключительной ситуации не возникает.

3. При переполнении сверху вырабатывается ошибка, операнд-сумма не изменяется, а коды условий непредсказуемы.

CLR	очистка

Формат:

код операции DST.WX

Коды условий:

N <- 0;

Z <- 1;

V <- 0;

C <- C;

Исключительная ситуация:

отсутствует

Коды операций:

D4 CLRF

7C CLRD очистка чисел в формате с плавающей

CLRG запятой

7CFD CLRH

Описание.

Операнд-приемник DST замещается нулем.

Примечание. Инструкция CLRX DST эквивалентна MOVX #0, DST, но на 5 (для формата F), 9 (для D и G), 17 (формат H) байтов короче.

CMP сравнение

Формат:

код операции SRC1.RX, SRC2.RX

Коды условий:

N <- SRC1 LSS SRC2;

Z <- SRC1 EQL SRC2;

V <- 0;

C <- 0;

Исключительная ситуация:

резервный операнд

Коды операций:

5L	CMPF	
7L	CMPD	сравнение чисел в формате с плавающей
5LFD	CMPG	запятой
7LFD	CMPH	

Описание.

Операнд 1 (SRC1) сравнивается с операндом 2 (SRC2). Команда воздействует только на коды условий.

Примечание. При ошибке по резервному операнду коды условий непредсказуемы.

CVT — преобразование

Формат:

код операции SRC.RX, DST.WY

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- (источник не может быть представлен в виде, задаваемом приемником);

C <- 0;

Исключительная ситуация:

целочисленное переполнение сверху

переполнение сверху в формате с плавающей запятой

переполнение снизу в формате с плавающей запятой

резервный операнд

Код операции:

4C	CVTBF	преобразование байта
6C	CVTBD	в форматы с плавающей
4CFD	CVTBG	запятой F, D, G и H
6CFD	CVTBH	соответственно
4D	CVTWF	преобразование слова
6D	CVTWD	в форматы с плавающей
4DFD	CVTWG	запятой F, D, G и H
6DFD	CVTWH	соответственно
4E	CVTLF	преобразование длинного слова
6E	CVTLD	в формат с плавающей
4EFD	CVTLG	запятой F, D, G и H
6EFD	CVTLH	соответственно
48	CVTFB	преобразование
68	CVTDB	из формата с плавающей
48FD	CVTGB	запятой
68FD	CVTHB	в байт
49	CVTFW	преобразование из
69	CVTDW	формата с плавающей
49FD	CVTGW	запятой
69FD	CVTHW	в слово
4A	CVTFL	преобразование из

6A	CVTDL	формата с плавающей
4AFD	CVTGL	запятой в длинное
6AFD	CVTHL	слово
4B	CVTRFL	преобразование
6B	CVTRDL	числа с
4BFD	CVTRGL	плавающей запятой
6BFD	CVTRHL	в длинное слово (с округлением)
56	CVTFD	преобразование
99FD	CVTFG	из
98D	CVTFH	одного
76	CVTDF	формата
32FD	CVTDH	
33FD	CVTGF	с
56FD	CVTGH	плавающей
F6FD	CVTHF	запятой
F7FD	CVTHD	в
76FD	CVTHG	другой

Описание.

Операнд-источник SRC преобразуется в тип данных операнда-приемника DST, и операнд-приемник замещается результатом. Форма преобразования следующая:

CVTBF точное

CVTBD точное

VCTBG точное

CVTBH точное

CVTWF	точное
CVTWD	точное
CVTWG	точное
CVTWH	точное
CVTLF	округление
CVTLD	точное
CVTLG	точное
CVTLH	точное
CVTFB	усечение
CVTDB	усечение
CVTGB	усечение
CVTHB	усечение
CVTFW	усечение
CVTDW	усечение
CVTGW	усечение
CVTHW	усечение
CVTFL	усечение
CVTRFL	округление
CVTDL	усечение
CVTRDL	округление
CVTGL	усечение
CVTRGL	округление
CVTHL	усечение
CVTRHL	округление
CVTFD	точное
CVTFG	точное
CVTFH	точное

CVTDF округление

CVTDH точное

CVTGF округление

CVTGH точное

CVTHF округление

CVTND округление

CVTHG округление

Примечания:

1. Ошибка по переполнению сверху в формате с плавающей запятой может произойти только в инструкциях CVTHF, CVTDF, CVTHF, CVTHD и CVTHG; операнд-приемник остается без изменения, а коды условий непредсказуемы.

2. Только преобразования с операндом в формате с плавающей запятой в качестве источника могут привести к ошибке по резервному операнду. При ошибке по резервному операнду операнд-приемник остается без изменения, а коды условий непредсказуемы.

3. Только преобразования с целочисленным операндом-приемником могут вызвать целочисленное переполнение сверху. При целочисленном переполнении операнд-приемник замещается младшими разрядами истинного результата.

4. Переполнение снизу в формате с плавающей запятой может произойти только в инструкциях CVTGF, CVTHF, CVTHD и CVTHG. Если разряд FU установлен, вырабатывается ошибка. В результате переполнения снизу в формате с плавающей запятой нуль запоминается в качестве результата только в том случае, если разряд FU сброшен. При ошибке по переполнению

снизу операнд-приемник остается без изменения. Если разряд FU сброшен, операнд-приемник замещается нулем и исключительной ситуации не возникает.

DIV деление

Формат:

код операции DIVR.RX, QUO.MX 2 операнда

код операции DIVR.RX, DIVD.RX, QUO.WX 3 операнда

Коды условий:

N <- QUO LSS 0;

Z <- QUO EQL 0;

V <- (переполнение сверху с плавающей запятой или делитель равен нулю);

C <- 0;

Исключительная ситуация:

переполнение формата с плавающей запятой сверху

переполнение формата с плавающей запятой снизу

деление на нуль

резервный операнд

Коды операций:

46	DIVF2	
47	DIVF3	
66	DIVD2	деление чисел в формате с
67	DIVD3	плавающей запятой
46FD	DIVG2	двух и трехоперандное
47FD	DIVG3	
66FD	DIVH2	

67FD DIVH3

Описание.

В двухоперандном формате операнд-частное QUO делится на операнд-делитель DIVR и замещается округленным результатом. В трехоперандном формате операнд-делимое DIVD делится на операнд-делитель DIVR, а операнд-частное замещается округленным результатом QUO.

Примечания:

1. При ошибке по резервному операнду частное не изменяется, а коды условий непредсказуемы.

2. По переполнению формата с плавающей запятой снизу вырабатывается ошибка, если разряд FU установлен. Только в том случае, если FU сброшен, в качестве результата записывается нуль. При ошибке по переполнению снизу частное остается без изменения. Если FU сброшен, операнд-частное замещается нулем и исключительной ситуации не возникает.

3. По переполнению формата с плавающей запятой сверху вырабатывается ошибка, операнд-частное остается без изменения, а значения кодов условий непредсказуемы.

4. При делении на нуль операнд-частное и коды условий устанавливаются так же, как в пункте 3.

EMOD расширенное умножение с выделением целой части

Формат:

EMODF и EMODD:

код операции: MULR.RX, MULRX.R3, MULD.RX, INT.WL, FRACT.WX

EMODG и EMODH:

Код операции MULR.RX, MULRX.RW, MULD.RX, INT.WL, FRACT.WX

Код операций:

N <- FRACT LSS 0;

Z <- FRACT EQL 0;

V <- (целочисленное переполнение);

C <- 0;

Исключительная ситуация:

целочисленное переполнение

переполнение формата с плавающей запятой снизу

резервный операнд

Коды операций:

54	EMODF	
74	EMODD	расширенное умножение чисел в формате
54FD	EMODG	с плавающей запятой с выделением
74FD	EMODH	целой части

Описание.

К операнду-множителю MULR присоединяется операнд-расширение MULRX, который дает 8 (EMODD и EMODF), 11 (EMODG), или 15 (EMODH) дополнительных младших разрядов в мантиссе (FRACT). Младшие 5 или 1 разрядов 16-разрядного операнда расширения игнорируются инструкциями EMODG и EMODH соотвественно. Операнд-множимое MULD умножается на расширенный операнд-множитель. Умножение выполняется таким образом, что результат эквивалентен точному произведению, усеченному (перед нормализацией) до мантиссы в 32 разряда в

формате F, 64 разряда в форматах D и G и 128 в формате H с плавающей запятой. Если рассматривать результат операции как сумму, состоящую из дробной и целой части с одинаковым знаком, операнд-целая часть INT замещается целой частью результата, а операнд-дробь FRACT замещается округленной дробной частью результата.

Примечания:

1. При ошибке по резервному операнду операнды целой и дробной части остаются без изменения.

2. При переполнении формата с плавающей запятой снизу происходит ошибка, если установлен разряд FU. Целая и дробная части замещаются нулем вследствие переполнения снизу только в том случае, если FU сброшен. При ошибке по переполнению снизу целая и дробная части остаются без изменения. Если FU сброшен, целая и дробная части замещаются нулями, исключительной ситуации не возникает.

3. При целочисленном переполнении операнд целой части замещается младшими разрядами истинного результата.

4. Переполнение формата с плавающей запятой сверху обнаруживается по целочисленному переполнению, однако целочисленное переполнение возможно и при отсутствии переполнения формата с плавающей запятой.

5. Знаки дробной и целой части совпадают, за исключением того случая, когда возникает целочисленное переполнение сверху.

6. Так как дробная часть округляется после выделения целой части, возможно, что величина дробной части становит-

см. разной 1.

MNEG пересылка с отрицанием

Формат:

Код операции SRC.RX, DST.WX

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

резервный операнд

Коды операций:

52	MNEGF	
72	MNEGD	пересылка с отрицанием
52FD	MNEGG	в формате D, F, G, H
72FD	MNEGH	

Описание.

Операнд-приемник DST замещается операндом-источником SRC, взятым с противоположным знаком.

Примечание. По ошибке по резервному операнду операнд-приемник остается без изменения, а коды условий непредсказуемы.

MOV пересылка

Формат:

код операции SRC.RX, DST.WX

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

резервный операнд

Коды операций:

50	MOVE	
70	MOVD	пересылка в формате
50FD	MOVG	с плавающей запятой
70FG	MOVH	

Описание.

Операнд-источник SRC пересылается по адресу операнда-приемника DST.

Примечание. При ошибке по резервному операнду операнд-приемник не изменяется, а коды условий непредсказуемы.

MUL умножение

Формат:

код операции MULR.RX, PROD.WX 2 операнда

код операции MULR.RX, MULD.RX, PROD.WX 3 операнда

Коды условий:

N <- PROD LSS 0;

Z <- PROD EQL 0;

V <- (переполнение сверху формата с плавающей запятой);

C <- 0;

Исключительная ситуация:

переполнение сверху формата с плавающей запятой

переполнение снизу формата с плавающей запятой

резервный операнд

Коды операций:

44	MULF2	
45	MULF3	
64	MULD2	
65	MULD3	двух и трех операндное умножение в
44FD	MULG2	формате с плавающей запятой
45FD	MULG3	
64FD	MULH2	
65FD	MULH3	

Описание.

В двухоперандном формате операнд-произведение PROD умножается на множитель MULR и замещается округленным результатом в PROD. В трехоперандном формате содержимое операнда MULD умножается на содержимое операнда MULR, результат помещается по адресу операнда PROD.

Примечания:

1. При ошибке по резервному операнду

операнд-произведение остается без изменения, а коды условий непредсказуемы.

2. По переполнению с плавающей запятой снизу, если установлен разряд FU, вырабатывается ошибка. Нуль сохраняется в качестве результата при переполнении снизу только в том случае, если FU сброшен. При ошибке по переполнению снизу операнд-произведение остается без изменения. Если FU сброшен, операнд-произведение заменяется нулем и исключительной ситуации не возникает.

3. При переполнении с плавающей запятой сверху вырабатывается ошибка; операнд-произведение остается без изменения, а коды условий непредсказуемы.

POLY вычисление полинома

Формат:

код операции ARG.RX, DEGREE.RW, TBLADD.AB

Коды условий:

N <- R0 LSS 0;

Z <- R0 EQL 0;

V <- (переполнение формата с плавающей запятой);

C <- 0;

Исключительная ситуация:

переполнение сверху с плавающей запятой

переполнение снизу с плавающей запятой

резервный операнд

Коды операций:

55 POLYF

75	POLYD	вычисление полинома в форматах F, D, G
55FD	POLYG	и H с плавающей запятой
75FD	POLYH	

Описание.

Операнд адреса таблицы TBLADDR является указателем к таблице коэффициентов полинома. Операнд адреса таблицы указывает на коэффициент старшего члена полинома. Коэффициенты для младших членов располагаются в последовательно возрастающих адресах. Тип данных коэффициентов совпадает с типом данных операнда-аргумента ARG. Подсчет осуществляется методом горнера и содержимое регистра R0 (R1'R0 для POLYD и POLYG, R3'R2'R1'R0 для POLYH) замещается результатом. Результат вычисляется следующим образом:

если D=степень (DEGREC)

и X=аргумент (ARG)

результат= C[0] + X*(C[1] + X*(C[2] +...+ X*C[D]))

Операнд-степень - слово без знака, определяющее номер старшего коэффициента, участвующего в подсчете. Для POLYH необходимо резервирование 4-х длинных слов в стеке для хранения аргументов в случае прерывания инструкции.

Примечания:

1. После вычисления:

POLYF

R1=0

R0=результат

R2=0

R3=адрес таблицы+степень * 4+4

POLID и POLYG

R0=старшая часть результата

R1=младшая часть результата

R2=0

R3=адрес таблицы + степень * 8+8

R4=0

R5=0

POLYH:

R0=старшая часть результата

R1=вторая старшая часть результата

R2=вторая младшая часть результата

R3=младшая часть результата

R4=0

R5=адрес таблицы + степень * 16+16

2. При возникновении исключительной ситуации:

2.1. Если PSL<FPD>=0, вырабатывается ошибка, и все задействованные данные восстанавливаются до первоначального состояния.

2.2. Если PSL<FPD>=1, выполнение инструкции приостанавливается, а состояние запоминается в общих регистрах следующим образом:

POLYF.

R0 = TMP3 промежуточный результат после итерации, предшествующей переполнению сверху или снизу

R1 = аргумент

R2<7:0> = TMP1; число оставшихся итераций

R2<31:8> = специфическое использование

R3 = TMP2 указатель на данное в таблице, вызвавшее исключительную ситуацию

POLYD и POLYG

R1'R0 = TMP3 промежуточный результат после итерации, предшествующей переполнению сверху или снизу

R2<7:0> = TMP1 ; число оставшихся итераций

R2<31:8> = специфическое использование

R3 = TMP2 ; указывает на данное в таблице, ; вызвавшее исключительную ситуацию

R5'R4 = аргумент

POLYH

R3'R2'R1'R0=TMP3 промежуточный результат после итерации, предшествующей ; вызвавшей переполнение ; сверху или снизу

R4<7:0> = TMP1 число оставшихся итераций

R4<31:8> = специфическое применение

R5 = TMP2 ; указывает на данное в таб- ; лице, вызвавшее исключи- чительную ситуацию

При ошибке во время исполнения инструкции аргумент сохраняется в используемом стеке.

3. Если беззнаковый операнд-степень равен 0, а аргу-

мент не является резервным операндом, результат равен C[0].

4. Если операнд-степень больше 31, вырабатывается ошибка по резервному операнду.

5. При ошибке по резервному операнду:

5.1. Если PSL<FPD>=0, резервным операндом является либо степень (больше 31), либо аргумент, либо один из коэффициентов.

5.2. Если PSL<FPD>=1, резервный операнд является коэффициентом, а R3 (кроме POLYH) или R5 (для POLYH) указывает на значение, вызвавшее исключительную ситуацию.

5.3. Состояния сохраненных кодов условий и других регистров непредсказуемы.

6. При переполнении формата с плавающей запятой снизу (потеря значимости) после операции округления на любой итерации в цикле вычисления вырабатывается ошибка, если FU установлен. Если FU сброшен, временный результат заменяется нулем, и подсчет продолжается. В этом случае конечный результат может быть ненулевым, если переполнение снизу возникло перед последней итерацией.

7. При переполнении формата с плавающей запятой сверху после операции округления на любой итерации в цикле вычисления выполнение инструкции прекращается и вырабатывается ошибка.

8. Если аргумент равен 0, а один из коэффициентов в таблице является резервным операндом, факт, возникает ошибка или нет по резервному операнду, является непредсказуемым.

9. Аргумент всегда будет в стеке, если после установки FPD возникает прерывание или ошибка формата с плавающей запятой. Если четыре длинных слов в вершине стека перекрывают какой-либо из операндов-источников, результаты являются непредсказуемыми.

Пример.

Вычислить P(X)=C0+C1*X+C2*X**2,

где C0=1.0, C1=.5, C2=.25

```
        POLYF     X,#2,PTABLE



PTABLE: .FLOAT    0.25        C2
        .FLOAT    0.5         C1
        .FLOAT    1.0         C0
```

SUB вычитание

Формат:

код операции SUB.RX, DIF.MX 2 операнда

код операции SUB.RX, MIN.RX, DIF.WX 3 операнда

Коды условий:

N <- DIF LSS 0;

Z <- DIF EQL 0;

V <- (переполнение формата с плавающей запятой сверху);

C <- 0;

Исключительная ситуация:

переполнение формата с плавающей запятой сверху

переполнение формата с плавающей запятой снизу

резервный операнд

Коды операций:

42	SUBF2	
43	SUBF3	
62	SUBD2	
63	SUBD3	двух и трехоперандное вычитание чисел
42FD	SUBG2	формата F, D, G и H с плавающей запятой
43FD	SCBG3	
62FD	SUBH2	
63FD	SUBH3	

Описание.

В двухоперандном формате операнд-вычитаемое SUB вычитается из операнда-разности DIF и разность заменяется округленным результатом. В трехоперандном формате вычитаемое SUB отнимается от уменьшаемого MIN, а операнд-разность DIF заменяется округленным результатом.

Примечания:

1. При ошибке по резервному операнду операнд-разность остается без изменения, а коды условий непредсказуемы.

2. При переполнении формата с плавающей запятой снизу, если разряд FU установлен, возникает ошибка. Нуль сохраняется как результат переполнения с плавающей запятой только в том случае, если FU сброшен. При ошибке по переполнению снизу операнд-разность остается без изменения. Если

разряд FU сброшен, разность замещается нулем и исключительной ситуации не возникает.

3. При переполнении формата с плавающей запятой сверху вырабатывается ошибка, разность остается без изменения, а коды условий непредсказуемы.

TST. проверка

Формат:

код операции SRC.RX

Коды условий:

N <- SRC LSS 0;

Z <- SRC EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

резервный операнд

Коды операций:

53	TSTF	
73	TSTD	тестирование в форматах F, D, G и H
53FD	TSTG	с плавающей запятой
73FD	TSTH	

Описание.

Установка кодов условий в зависимости от значения операнда-источника SRC.

Примечания:

1. TSTX SRC эквивалентна CMPX SRC,#0, но на 5 (формат F), 9 (форматы D или G) или 17 (формат H) байтов короче.

2. При ошибке по резервному операнду коды условий непредсказуемы.

4.10. Инструкции, работающие с символьными строками

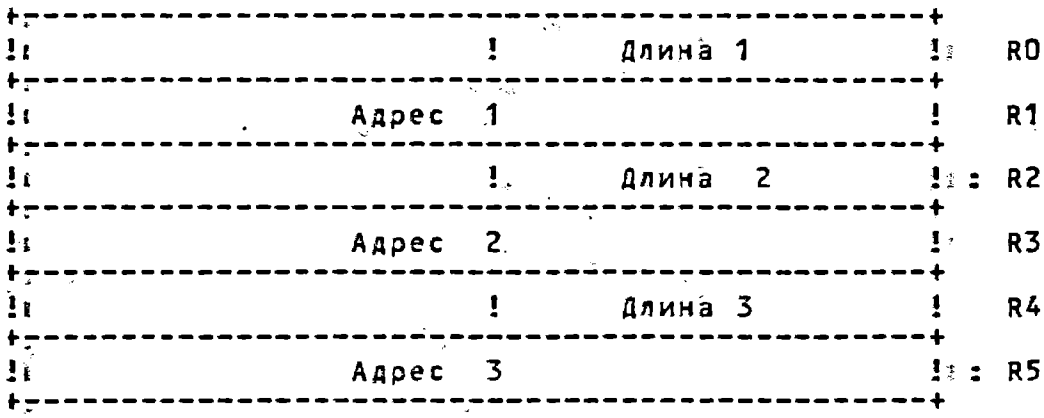
Символьная строка специфицируется двумя операндами:

1) словом без знака, определяющим длину символьной строки в байтах;

2) адресом младшего адресуемого байта в символьной строке. Этот операнд адресного типа доступа;

В каждой инструкции, работающей со строками символов, используются регистры общего назначения R0-R1, R0-R3 или R0-R5 для хранения управляющего блока, в котором содержатся модифицируемые адрес и состояние во время выполнения инструкции. По окончании эти регистры программно доступны и могут быть использованы как спецификаторы строки для последующей инструкции, выполняемой над непрерывной строкой символов. Во время исполнения инструкций проверяется условие ожидания прерывания, и в случае обнаружения прерывания управляющий блок корректируется, в PSL устанавливается разряд "первая часть выполнена", и выполнение инструкции прерывается. После прерывания выполнение инструкции возобновляется.

Формат управляющего блока



Поля длина 1, длина 2 (если требуется) и длина 3 (если требуется) содержат число байтов, которое осталось обработать в первом, втором и третьем операндах-строках соответственно. Поля адрес 1, адрес 2 (если требуется) и адрес 3 (если требуется) содержат адреса следующего обрабатываемого байта в первом, втором и третьем операндах-строках соответственно.

Когда длина строки равна нулю, то ошибки обращения к памяти возникнуть не могут, так как обращений к памяти не происходит.

В данном подразделе описываются следующие инструкции:

- сравнение двух строк (3 операнда) CMPC3 LEN.RW, SRC1ADDR.AB, SRC2ADDR.AB, [R0-3.WL];
- сравнение двух строк (пять операндов) CMPC5 SPCLLEN.RW, SRCLADDR.AB, FILL.RB, SRC2LEN.RW, SRC2ADDR.AB, [R0-3.WL];
- поиск символа LOCC CHAR.RB, LEN.RW, ADDR.AB, [R0-1.WL];
- поиск подстроки символов MATCHC LENL.RW, ADDRL.AB, LEN2.RW,ADDR2.AB, [R0-3.WL];
- пересылка символов (3 операнда) MOVC3 LEN.RW, SRCADDR.AB,

DSTADDR.AB, [R0-5.WL];

- пересылка символов (5 операндов) MOVC5 SRCLEN.RW, SRCADDR.AB, FILL.RB, DSDLEN.RW, DSTADDR.AB, [R0-5.WL];
- пересылка символов с перекодировкой MOVTC SRCLEN.RW, SRCADDR.AB, FILL.RB, TBLADDR.AB, DSTLEN.RW, DSTADDR.AB, [R0-5.WL];
- пересылка с перекодировкой до помеченного символа MOVTUC SRCLEN.RW, SRCADDR.AB, ESC.RB, TBLADDR.AB, DSTLEN.RW, DSTADDR.AB, [R0-5.WL];
- поиск несоответствия символов SCANC LEN.RW, ADDR.AB, TBLADDR.AB, MASK.RB, [R0-3.WL];
- пропуск символа в SKPC CHAR.RB, LEN.RW, ADDR.AB, [R0-1.WL];
- поиск несоответствия символов SPANC LEN.RW, ADDR.AB, TBLADDR.AB, MASK.RB, [R0-3.WL].

CMPC сравнение двух строк

Формат:

код операции LEN.RW, SRC1ADDR.AB, SRC2ADDR.AB
! 3 операнда

код операции SRC1LEN.RW, SRC1ADDR.AB, FILL.RB,
SRC2LEN.RW, SRC2ADDR.AB ! 5 операндов

отсутствует

29 CMPC3 сравнение двух строк (3 операнда)

2D CMPC5 сравнение двух строк (5 операндов)

Описание.

В трехоперандном формате сравниваются побайтно две строки, расположенные, соответственно, по адресам SRC1ADDR и SRC2ADDR и имеющих одинаковую длину LEN. Сравнение продолжается до обнаружения несовпадения или исчерпания всех байтов в строках. Коды условий устанавливаются по результату сравнения последних байтов. В пятиоперандном формате байты строки 1, специфицированной длиной SRC1LEN и адресом SRC1ADDR, сравниваются с байтами строки 2, специфицированной длиной SRC2LEN и адресом SRC2ADDR. Если одна строка длиннее другой, более короткая строка дополняется для сравнения до длины большей строки с помощью (в старших адресах) байтов, указанных операндом FILL. Сравнение продолжается до тех пор, пока не обнаружено несовпадение или не проверены все байты. Коды условий устанавливаются по результатам сравнения последних байтов. И для CMPC3 и для CMPC5 две нулевые строки считаются равными (то есть Z устанавливается, N, V, C сбрасываются).

Примечания:

1. После выполнения CMPC3:

R0=числу оставшихся байтов в строке 1 (включая байт, по которому сравнение прекратилось); R0=0 только если строки равны;

R1=адрес байта в строке 1, по которому сравнение прекратилось; если строки равны, то адрес байта, непосредственно следующего за строкой 1;

R2=R0;

R3=адрес байта в строке 2, по которому сравнение прекратилось. Если строки совпадают, то адрес байта, следующего за строкой 2.

2. После выполнения CMPC5:

R0=число байтов, оставшихся в строке 1 (включая байт, по которому сравнение прекратилось); R0 равно 0, только если строки 1 и 2 одной длины и равны, или строка 1 была исчерпана до того, как сравнение прекратилось;

R1=адрес байта в строке 1, по которому сравнение прекратилось; если сравнение не прекратилось до исчерпания строки 1, то адрес байта, следующего за строкой 1;

R2=число байтов, оставшихся в строке 2 (включая байт, по которому сравнение прекратилось); R2 равен нулю только если строки 2 и 1 имеют разную длину, или если строка 2 была исчерпана до того, как сравнение прекратилось;

R3=адрес байта в строке 2, по которому сравнение прекратилось; если сравнение не было прекращено до исчерпания строки 2, то адрес байта, следующего за строкой 2.

3. Если обе строки имеют нулевую длину, код Z устанавливается, N, V и C сбрасываются так же, как в случае совпадения двух строк.

LOCC поиск символа

Формат:

код операции CHAR.RB, LEN.RW, ADDR.AB

Коды условий:

N <- 0;

Z <- R0 EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует

Коды операций:

3a LOCC поиск символа

Описание.

Операнд-символ CHAR сравнивается с байтами в строке, специфицируемой операндами длины LEN и адреса ADDR. Сравнение производится до тех пор, пока не будет обнаружено совпадение, или не исчерпаются все байты в строке. Если обнаружено совпадение, разряд Z в кодах сбрасывается, в противном случае разряд Z устанавливается.

Примечания:

1. После выполнения:

R0 = числу байтов, оставшихся в строке (включая обнаруженный), если байт найден, в противном случае нуль.

R1 = адресу обнаруженного байта, если он был обнаружен, в противном случае адрес у следующего за строкой байта.

2. Если строка имеет нулевую длину, код условий Z устанавливается, как если бы ни один из байтов строки не совпал с символом.

MATCHC поиск подстроки символов

Формат:

код операции OBJLEN.RW, OBJADDR.AB, SRCLEN.RW,
SRCADDR.AB

Коды условий:

N <- 0;

Z <- R0 EQL 0; !подстрока найдена

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует

Коды операций:

39 MATCHC поиск подстроки

Описание.

В исходной строке, специфицированной двумя операндами: длиной SRCLEN и адресом SRCADDR ищется подстрока, совпадающая с об'ектной строкой, специфицированной длиной OBJLEN и адресом OBJADDR. Если найдена подстрока, то разряд Z устанавливается, в противном случае - он очищается.

Примечания:

1. После выполнения:

R0=0, если найдена подстрока, в противном случае числу байтов в об'ектной строке;

R1=, если произошло совпадение, то адресу байта, следующего за об'ектной строкой, то есть OBJADDR+OBJLEN; в противном случае - адресу об'ектной строки;

R2=, если произошло совпадение, то числу байтов, оставшихся в исходной строке, в противном случае нуль;

R3=, если произошло совпадение, то адресу байта, следующего за последним совпадающим; в противном случае - адресу байта, следующего за строкой-источником, то есть SRCADDR+SRCLEN.

Для об'ектных и исходных строк нулевой длины R1 и R3 содержат адреса об'ектной и исходной строк соответственно.

2. Если обе строки имеют нулевую длину или об'ектная строка имеет нулевую длину, разряд Z в кодах условий устанавливается, а регистры R0-R3 устанавливаются в такое состояние, как если бы подстрока была найдена.

3. Если исходная строка имеет нулевую длину, а об'ектная строка - ненулевую, то разряд Z сбрасывается, а регистры R0-R3 устанавливаются в такое состояние, как если бы подстрока была обнаружена.

MOVC — пересылка символов

Формат:

код операции LEN.RW, SRCADDR.AB, DSTADDR.AB — 3 операнда

код операции SRCLEN.RW, SRCADDR.AB, FILL.RB, DSTLEN.RW, DSTADDR.AB — 5 операндов

Коды условий:

N <- 0; !MOVC3

Z <- 1;

V <- 0;

C <- 0;

N <- SRCLEN LSS DSTLEN; !MOVC5

Z <- SRCLEN EQL DSTLEN;

V <- 0;

C <- SRCLEN LSSU DSTLEN;

Исключительная ситуация:

отсутствует

Коды операций:

28 MOVC3 пересылка символов (3 операнда)

2C MOVC5 пересылка символов (5 операндов)

Описание.

В трехоперандном формате инструкция копирует строку символов, указанную SRCADD в поле байт, начиная от DSTADDR, длина строки описывается параметром LEN. В пяти-операндном формате в строку-приемник, специфицированная длиной DSTLEN и адресом приемника DSTADDR, копируется содержимое строки-источника, специфицированная длиной источника SRCLEN и адресом источника SRCADDR. Если строка-приемник длиннее, чем строка-источник, старшие адресуемые байты приемника замещаются операндом-заполнителем FILL.

Если строка-приемник короче строки-источника, старшие адресуемые байты строки-источника не пересылаются. Выполнение инструкции такое, что перекрывание строки источника и приемника не влияет на результат.

Примечания:

1. После выполнения MOVC3:

R0=0

R1=адресу байта, следующего за строкой-источником

R2=0

R3=адресу байта, следующего за строкой-приемником

R4=0

R5=0

2. После выполнения MOVC5:

R0=числу оставшихся в строке-источнике байтов.

R0 не равно нулю только, если строка-источник длиннее строки-приемника.

R1=адресу байта, следующего за последним переданным байтом строке-источнике

R2=0

R3=адресу байта, следующего за строкой-приемником

R4=0

R5=0

3. Для копирования одного блока памяти в другой предпочтительнее MOVC3.

4. Для заполнения блока памяти символами-заполнителями предпочтительнее использовать MOVC5 со строкой-источником нулевой длины.

MOVTC. пересылка симаолов с перекодировкой

Формат:

код операции SRCLEN.RW, SRCADR.AB, FILL.RB,
TBLADDR.AB, DSTLEN.RW, DSTADDR.AB

Коды условий:

N <- SRCLEN LSS DSTLEN;

Z <- SRCLEN EQL DSTLEN;

V <- 0;

C <- SRCLEN LSSU DSTLEN;

Исключительная ситуация:

отсутствует

Коды операций:

2E MOVTC пересылка символов с перекодировкой

Описание.

Строка-источник, специфицированная операндами длины SRCLEN и адреса источника SRCADDR, перекодируется и заменяет строку-приемник, специфицированную длиной DSTLEN и адресом приемника DSTADDR. Перекодировка заключается в использовании каждого байта в строке-источнике в качестве индекса в таблицу из 256 байтов, нулевой вход в которую специфицирован операндом адреса таблицы TBLADDR. Выбранный из таблицы перекодировки байт заменяет символ в строке-приемнике. Если строка-приемник длиннее строки-источника, старшие адресуемые байты строки-приемника заменяются символом-заполнителем FILL. Если строка-приемник короче строки-источника, старшие байты строки-источника не перекодируются и не передаются. Выполнение инструкции такое, что перекрытие строк источника и приемника не влияет на результат. Если строка-приемник перекрывает таблицу трансляции, значение строки-приемника непредсказуемо.

Примечание. После выполнения:

R0=числу оставшихся неперекодируемых байтов в строке-источнике, R0 не равен нулю только в том случае, если строка-источник длиннее, чем строка-приемник;

R1=адресу байта, следующего за последним перекодированным байтом в строке-источнике;

R2=0;

R3=адрес таблицы перекодировки;

R4=0;

R5=адресу байта, следующего за строкой-приемником.

MOVTUC пересылка с перекодировкой до указанного символа

Формат:

код операции SRCLEN.RW, SRCADDR.AB, ESC.RB,
TBLADDR.AB, DSTLEN.RW, DSTADDR.AB

Коды условий:

N <- SRCLEN LSS DSTLEN;

Z <- SRCLEN EQL DSTLEN;

V <- (окончание через байт ESC);

C <- SRCLEN LSSU DSTLEN;

Исключительная ситуация:

отсутствует

Коды операций:

2F MOVTUC пересылка с перекодировкой до помеченного символа

Описание.

Строка-источник, специфицированная операндами длины

источника SRCLEN и адреса источника SRCADDR, перекодируется и замещает строку-приемник, специфицированную операндами длины DSTLEN адреса приемника DSTADDR. Перекодировка заключается в использовании каждого байта строки-источника в качестве индекса в таблицу из 256 байт, нулевой вход в которую задается операндом адреса таблицы TBLADDR. Выбранный таким образом из таблицы перекодировки байт замещает байт в строке-приемнике. Трансляция продолжается до тех пор, пока перекодируемый байт не совпадет с символом ESC или до тех пор, пока строка-источник, либо строка-приемник не будет исчерпана. Если перекодирование прекращено, так как встретился байт ESC, устанавливается разряд V в кодах условий. В противном случае он очищается. Если строка-приемник перекрывает таблицу, содержимое строки-приемника и регистров с R0 по R5 непредсказуемо. Если адреса источника и приемника совпадают, трансляция выполняется правильно.

Примечание. После выполнения:

R0=числу байтов, оставшихся в строке-источнике (включая байт, вызвавший выход). R0=0 только в том случае, если вся строка-источник была перекодирована и передана;

R1=адресу байта, на котором строка-источник исчерпалась, или который вызвал выход; или, если не было выхода, и не исчерпалась строка-приемник, это адрес байта, следующего за строкой-источником;

R2=0;

R3=адрес таблицы;

R4=число байтов, оставшихся в строке-приемнике;

R5=адрес байта в строке-приемнике, в который должен был быть послан после перекодировки байт, вызвавший выход, или в который должен был быть послан очередной байт, если бы строка-источник в этот момент кончилась; если же не было ни выхода, ни конца строки-источника, то адресу байта, следующего за строкой-приемником.

SCANC поиск несоответствия символов

Формат:

код операции LEN.RW, ADDR.AB, TBLABBR.AB, MASK.BR

Коды условий:

N <- 0;

Z <- R0 EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует

Коды операций:

2A SCANC поиск несоответствия символов

Описание.

Байты, входящие в строку, специфицированную операндами длины LEN и адреса ADDR, последовательно используются как индексы в таблицу из 256 байт, нулевой вход в которую специфицирован операндами адреса таблицы TBLADDR. Выбранный из

таблицы байт логически умножается на операнд-маску MASK. Операция продолжается до тех пор, пока не будет получен ненулевой результат умножения или пока все байты строки не будут исследованы. Если обнаружен ненулевой результат умножения, сбрасывается разряд Z в кодах условий, в противном случае он устанавливается.

Примечания:

1. После выполнения:

R0=числу оставшихся байтов в строке (включая байт, вызвавший результат умножения). R0=0 только в том случае, если не было ненулевого результата;

R1=адресу байта, на котором получен ненулевой результат умножения; или, если ненулевого результата не было, адресу байта, следующего за строкой;

R2=0;

R3=адресу таблицы.

2. Если строка имеет ненулевую длину, разряд Z в кодах условий устанавливается, как если бы была просмотрена вся строка.

SKPC пропуск символа

Формат:

код операции CHAR.RB, LEN.RW, ADDR.AB

Коды условий:

N <- 0;

Z <- R0 EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует

Коды операций:

3B SKPC. пропуск символа

Описание.

Операнд-символ CHAR сравнивается с байтами в строке, специфицированной операндами длины LEN и адреса ADDR. Сравнение продолжается до тех пор, пока не будет обнаружено несовпадение, или пока все байты в строке не будут проверены. Если обнаружено несовпадение, разряд Z кода условий сбрасывается, в противном случае он устанавливается.

Примечания:

1. После выполнения:

R0=числу байтов, оставшихся в строке, включая несовпадающий, если таковой обнаружен; в противном случае 0;

R1=адресу обнаруженного байта, в противном случае адресу байта, следующего сразу за строкой.

2. Если строка имеет ненулевую длину, разряд Z устанавливается, как если бы все байты в строке были равны сравниваемому символу.

SPANC поиск соответствия символов

Формат:

Код операции LEN.RW, ADDR.AB, TBLADDR.AB, MASK.RB

Коды условий:

N <- 0;

Z <- R0 EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует

Коды операций:

2B SPANC поиск соответствия символов

Описание.

Байты в строке, специфицированной операндами длины LEN и адреса строки ADDR, последовательно используются как индексы в таблице из 256 байтов, нулевой вход в которую специфицирован операндом адреса таблицы TBLADDR. Выбранный из таблицы байт логически умножается на операнд-маску MASK. Операция продолжается до тех пор, пока не будет получен нулевой результат умножения или пока все байты в строке не проверены. Если обнаружен нулевой результат, разряд Z сбрасывается; в противном случает он устанавливается.

Примечания:

1. После выполнения:

R0=числу байтов, оставшихся в строке (включая байт, который дал нулевой результат умножения). R0=0 только в том случае, если не было обнаружено нулевого результата умножения

R1=адресу байта, вызвавшего нулевой результат умножения; или, если нулевого результата не было, адрес байта,

следующего за строкой

R2=0

R3=адрес таблицы.

2. Если строка имеет нулевую длину, то разряд Z кодов условий устанавливается, как если бы было полное соответствие символов.

4.11. Инструкция циклического контроля

Инструкция циклического контроля предназначена для подсчета и проверки циклического полинома, имеющего длину (CRC) до 32 разрядов.

Циклический контроль - это метод обнаружения ошибок, основанный на делении потока данных на полиномы CRC. Поток данных представлен как стандартная для СМ 1700 строка в памяти. Обнаружение ошибок осуществляется путем подсчета CRC в источнике, а затем в приемнике и сравнение полиномов в обеих точках. Выбор полинома таков, чтобы минимизировать число необнаруженных ошибок в блоках указанной длины.

Операндами для инструкции CRC является дескриптор строки, таблица из 16-ти длинных слов и первоначальный CRC. Дескриптор строки - это стандартная для СМ 1700 пара операндов: длина строки в байтах (до 65535) и начальный адрес строки. Содержимое таблицы зависит от используемого полинома CRC. Оно может быть подсчитано по полиному, приведенному в примечаниях. Первоначальный полином используется для того, чтобы правильно начать проверку. Обычно он имеет значение 0 или -1, но будет другим, если поток данных предс-

тавлен последовательностью отдельных строк.

Инструкция CRC выполняется путем просмотра строки и включением каждого байта потока данных в вычисляемый CRC. Каждый новый байт включается в SRC путем выполнения логической операции XOR с младшими 8 битами SRC. Затем CRC сдвигается вправо на один разряд, а слева подставляется 0. Самый правый разряд CRC (теряемый при сдвиге) используется в качестве признака выполнять ли операцию XOR между SRC полинома с результирующим CRC. Если этот бит установлен, операция XOR между полиномом и результирующим SRC выполняется. Затем CRC снова сдвигается вправо и складывается с результатом в общей сложности до 8 раз. В действительности алгоритм может сдвигать и по 1, 2 или 4 разряда за раз, используя соответствующие данные в специально построенной таблице. Инструкция создает 32-разрядный CRC. Для более коротких полиномов результат должен быть выделен из 32-разрядного поля. Поток данных должен иметь длину, кратную восьми разрядам. Если нет, он должен быть выравнен справа в строку с ведущими нулевыми разрядами.

CRC циклический контроль

Формат:

Код операции TBL.AB, INICRC.RL, STLEN.RW, STREAM.AB

Коды условий:

N <- R0 LSS 0;

Z <- R0 EQL 0;

V <- 0;

C <- 0;

Исключительная ситуация:

отсутствует.

Коды операций:

0B CRC циклический контроль

Описание.

Вычисляется CRC для потока данных, описанного дескриптором строки. STRLEN - это длина строки в байтах и STREAM - это начальный адрес строки. Первоначальный CRC задается операндом INICRC и обычно равен 0 или -1, если только CRC не вычисляется в несколько шагов. Результат остается в R0. Если полином меньше, чем обычно - менее 32 разрядов, то результат должен быть выделен из полученного числа. Полином CRC вычисляется с помощью содержимого таблицы TBL, состоящей из 16 длинных слов.

Примечания:

1. Если длина потока данных не кратна 8 разрядам, он должен быть выравнен вправо ведущими нулями.

2. Если полином CRC имеет порядок меньше 32, то результат должен быть выделен из младших разрядов R0.

3. Следующий алгоритм можно использовать для расчета таблицы CRC по заданному полиному:

разряд<N> полинома <- [коэффициент при X **[порядок-1-N]]

Эта программа имеется в системной библиотеке под именем LIB$CRC_TABLE (POLY.RL, TABLE.AB). TABLE - операнд, задающий местоположение 64-байтной (16 длинных слов) таблицы, в которую должен быть записан результат.

```
      SUBROUTINE LIB$CRC_TABLE (POLY, TABLE)
      INTEGER*4 POLY, TABLE(0:15), TMP, X
      DO 190 INDEX = 0, 15
      TMP = INDEX
      DO 150 I = 1, 4
      X = TMP .AND. 1
      TMP = ISHFT(TMP,-1)          !логический сдвиг вправо на
                                   !один разряд
      IF (X .EQ. 1) TMP = TMP .XOR. POLY
  150 CONTINUE
      TABLE(INDEX) = TMP
  190 CONTINUE
      RETURN
      END
```

4. Описание некоторых часто используемых полиномов CRC.

CRC-16 (используется в сетевых протоколах)

полином: X^16+X^15+X^2+1

представление 120001 (восьмеричное)

начальное значение полинома: 0

результат: R0 <15:0>

CCITT (используется в протоколах ADCCP, HDLC, SDLC)

полином: X^16+X^12+X^5+1

представление: 102010 (восьмеричное)

начальное значение: -1<15:0>

результат: дополнение до 1 содержимого R0<15:0>

AUTODIN-II (32-разрядный SRC)

полином: X^32+X^26+X^23+X^22+X^16+X^12+X^11+X^10+
+X^8+X^7+X^5+X^4+X^2+X+1

представление: EDB88320 (шестнадцатеричное)

начальное значение: -1 < 31:0>

результат: дополнение до 1 содержимого R0 <31:0>

5. Если таблица сформирована неверно, эта инструкция дает непредсказуемый результат. Причем в правильно созданной таблице элемент [0] всегда равен 0. Вычисления могут быть выполнены с использованием сдвига по 1, 2 и 4 разряда.

6. Если поток данных имеет нулевую длину, то в R0 заносится начальное значение CRC.

4.12. Инструкции, работающие с десятичными строками

Инструкции этого типа работают с десятичными строками в упакованном формате. Существуют инструкции преобразования из упакованного десятичного формата в формат числовой строки (перфоформат и зонный) и в формат числовой строки с выделенным ведущим знаком. Где это необходимо, определяется конкретный тип данных. Когда используется целая последовательность десятичных цифр, подразумевается любой из трех типов данных.

Десятичная строка определяется двумя операндами:

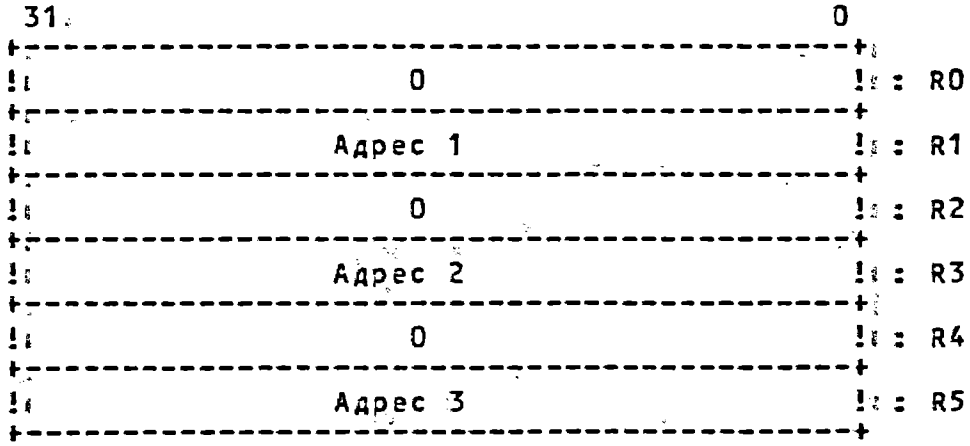
1) для всех десятичных строк длина строки представляет собой число цифр в этой строке. Число байтов в строке зависит от длины и типа десятичной строки (см. раздел 2);

2) адрес является адресом младшего адресуемого байта в строке. В этом байте содержится старшая значащая цифра для числовых строк в (п. 2.2.12) и упакованных десятичных строк в (п. 2.2.14). Для числовых строк с выделенным ведущим знаком в этом байте содержится знак (п. 2.2.13). Адрес специфицируется байтовым операндом с адресным типом доступа.

Каждая инструкция, работающая с десятичными строками, использует регистры общего назначения с R0 по R3 или с R0 по R5 для хранения управляющего блока, который содержит модифицированные адреса и их содержимое во время выполнения инструкции. По окончании инструкции регистры, содержащие адреса, программно доступны и могут быть использованы как спецификаторы строк в последующих инструкциях.

Во время выполнения инструкций проверяются условия ожидания прерывания, и, если они обнаружены, блок управления модифицируется. В PSL устанавливается разряд "первая часть выполнена" и инструкция прерывается. После прерывания выполнение инструкции возобновляется с прерванного места.

Формат блока управления



Поля адрес 1, адрес 2 и адрес 3 (если требуется) указывают адреса байтов, содержащих старшую значащую цифру первой, второй и третьей (если требуется) строк-операндов соответственно.

Инструкции, работающие с десятичными строками, рассматривают десятичные строки как целое с десятичной точкой, расположенной сразу за младшей значащей цифрой строки. Если строка, в которую должен быть помещен результат, длиннее результата, старшие цифры заполняются нулями.

4.12.1. Десятичное переполнение

Десятичное переполнение происходит в том случае, если строка-приемник не может вместить все цифры (исключая ведущие нули) результата. При переполнении строка-приемник замещается младшими значащими цифрами истинного результата с правильным знаком (даже, если запоминаемый результат равен -0). Ни старший полубайт упакованной десятичной строки с четной длиной, ни знаковый байт числовой строки с выделенным ведущим знаком не используются для хранения цифр результата.

4.12.2. Нулевые величины

Нулевой результат имеет положительный знак для всех операций, которые завершаются без десятичного переполнения, кроме инструкции CVTPT, которая не преобразует число -0 в +0. Однако, когда цифры теряются вследствие переполнения,

нулевой результат получает знак (положительный или отрицательный) истинного результата.

Десятичная строка со значением -0 считается идентичной десятичной строке со значением +0. Таким образом, например, при сравнении +0 совпадает с -0. Когда коды условий устанавливаются по результату -0, то они устанавливаются точно таким же образом, как если бы результат был равен +0; то есть N сбрасывается, а Z устанавливается.

4.12.3. Исключительная ситуация по резервному операнду

Прерывание по резервному операнду возникает в том случае, если длина десятичной строки выходит за пределы диапазона от 0 до 31, или если в инструкциях CVTSP или CVTPT встречаются запрещенный знак или цифра. PC указывает на код инструкции, вызвавшей исключительную ситуацию.

4.12.4. Непредсказуемые результаты

Если строка-источник содержит запрещенные данные, результат инструкции непредсказуем. За исключением CVTSP и CVTTP в десятичных строках правильность операнда-источника не проверяется.

Если операнд-приемник перекрывает любой из операндов-источников, результат операции в общем случае будет непредсказуем. Когда возникает прерывание по резервному операнду, в общем случае значение строки-приемника,

регистров, используемых инструкцией, и кодов условий непредсказуемы.

4.12.5. Инструкции для упакованных десятичных строк

Десятичные строки в упакованном формате, вырабатываемые инструкциями, работающими с десятичными строками, всегда имеют предпочтительное знаковое представление: 12 для "+" и 13 для "-". Десятичная упакованная строка с четной длиной всегда вырабатывается с нулевым значением в старшем полубайте первого байта строки.

Строка в упакованном десятичном формате содержит неверный полубайт, если:

- в знаковой позиции встречается цифра;
- в цифровой позиции встречается знак;
- для строк четной длины в старшем полубайте младшего адресуемого байта записана ненулевая величина.

4.12.6. Десятичные строки нулевой длины

Длина строки в упакованном десятичном формате может быть нулевой. В этом случае ее значение равно 0 (с плюсом или минусом) и занимает один байт памяти. Этот байт должен содержать цифру "0" в старшем полубайте и знак в младшем полубайте.

Длина числовой строки может быть равна 0. В этом случае строка памяти не занимает. Если операнд-приемник предс-

тавляет собой числовую строку нулевой длины, знак операции теряется. Когда специфицируется строка такого типа нулевой длины, ошибок обращения к памяти не возникает, так как обращения к памяти не происходит. Значение числовой строки нулевой длины идентично 0.

Длина числовой строки с выделенным ведущим знаком может быть равна нулю. В этом случае один байт памяти занимается под знак. Когда специфицирован операнд нулевой длины, происходит обращение к памяти и, если обнаружен неверный знак, происходит прерывание по резервному операнду. Значение числовой строки с выделенным ведущим знаком нулевой длины идентично 0.

4.12.7. Описание инструкций

В данном пункте описаны следующие инструкции:

- сложение (4 операнда) ADDP4 ADDLEN.RW, ADDADDR.AB, SUMLEN.RW, SUMADDR.AB, [R0-3.WL];
- сложение (6 операндов) ADDP ADDLEN.RW, ADD1ADDR.AB, ADD2LEN.RW, ADD2ADDR.AB, SUMLEN.RW, SUMADDR.AB, [R0-5.WL];
- арифметический сдвиг с округлением ASHP CNT.RB, SRCLEN.RW, SRCADDR.AB, ROUND.RB, DSTLEN.RW, DSTADDR.AB, [R0-3.WL];
- сравнение (3 операнда) CMPP3 LEN.RW, SRC1ADDR.AB, SRC2ADDR.AB, [R0-3.WL];
- сравненение (4 операнда) CMP4 SRC1LEN.RW, SRC1ADDR.AB, SRC2LEN.RW, SRC2ADDR.AB, [R0-3.WL];
- преобразование длинного слова в упакованную строку CVTLP SRC.RL, DSTLEN.RW, DSTADDR.AB, [R0-3.WL];

- преобразование упакованной строки в длинное слово CVTPL SRCLEN.RW, SRCADDR.AB, [RO-3.WL], DST.WL;
- преобразование упакованной строки в числовую с ведущим знаком CVTPS SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB, [RO-3.WL];
- преобразование упакованной строки в числовую строку CVTPT SRCLEN.RW, SRCADDR.AB, TBLADDR.AB, DSTLEN.RW, DSTADDR.AB, [RO-3.WL];
- преобразование числовой строки с ведущим знаком в упакованную строку CVTSP SRCLEEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB, [RO-3.WL];
- преобразование числовой строки с в упакованную строку CVTSP SRCLEN.RW, SRCADDR.AB, TBLADDR.AB, DSTLEN.RW, DSTADDR.AB, [RO-3.WL];
- деление в упакованном формате DIVP DIVRLEN.RW, DIVRADDR.AB, DIVDLEN.RW, DIVDADDR.AB, QUELEN.RW, QUOADDR.AB, [RO-5.WL, -16(SP):-1(SP).WB];
- пересылка в упакованном формате MOVP LEN.RW, SRCADDR.AB, DSTADDR.AB [RO-3.WL];
- умножение в упакованном формате MULP MULRLEN.RW, MULRADDR.AB, MULDLEN.RW, MULDADDR.AB, PRODLEN.RW, PRODADDR.AB, [RO-5.WL];
- вычитание (4 операнда) SUBP4 SUBLEN.RW, SUBADDR.AB, DIFLEN.RW, DIFADDR.AB, [RO-3.WL];
- вычитание (6 операндов) SUBP6 SUBLEN.RW, SUBADDR.AB, MINLEN.RW, MINADDR.AB, DIFLEN.RW, DIFADDR.AB, [RO-5.WL].

ADDP сложение в упакованном формате

Формат:

код операции: ADDLEN.RW, ADDADD.AB, SU1LEN.RW,
SUMADDR.AB

код операции: ADD LEN.RW, ADD1ADDR.AB, ADD2LEN.RW,
ADD2ADDR.AB, SUMLEN.RW, SUMADDR.AB

Коды условий:

N <- (строка-сумма) LSS 0;

Z <- (строка-сумма) EQL 0;

V <- (десятичное переполнение);

C <- 0;

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

20 ADDP4 сложение (4 операнда);

21 ADDP6 сложение (6 операндов)

Описание.

В 4-х операндном формате строка-слагаемое, специфицированная операндом длины ADDLEN и операндом адреса слагаемого ADDADDR, складывается со строкой-суммой, специфицированной операндом длины суммы SUMLEN и адреса суммы SUMADDR, и сумма замещается результатом.

В 6-ти операндном формате слагаемое 1, специфицированное длиной ADD1LEN и адресом ADD1ADDR, складывается со слагаемым 2, специфицированным длиной ADD2LEN и адресом ADD2ADDR. Строка-сумма, специфицированная длиной суммы

SUMLEN и адресом суммы SUMADDR, замещается результатом.

Примечания:

1. После выполнения ADDP4:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-слагаемого;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-суммы.

2. После выполнения ADDP6:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-слагаемого 1;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-слагаемого 2;

R4=0;

R5=адресу байта, содержащего старшую цифру строки-суммы.

3. Строка-сумма, регистры с R0 по R3 (или для ADDP6 с R0 по R5) и коды условий непредсказуемы, если строка-сумма перекрывает слагаемое, слагаемое 1 или 2; слагаемое, слагаемое 1, слагаемое 2 или сумма (только для четырехоперандного сложения) содержат неверный полубайт; или если происходит прерывание по резервному операнду.

ASHP арифметический сдвиг с округлением

Формат:

код операции CNT.RB, SRCLEN.RW, SRCADDR.AB,

ROUND.RB, DSTLEN.RW, DSTADDR.AB

Коды условий:

N <- (строка-приемник) LSS 0;

Z <- (строка-приемник) EQL 0;

V <- (десятичное переполнение);

C <- 0;

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

F8 ASHP арифметический сдвиг с округлением

Строка-источник, специфицированная операндами длины источника SRCLEN и адреса источника SRCADDR, масштабируется по основанию числа 10, с помощью операнда-счетчика CNT. Строка-приемник, специфицированная длиной приемника DSTLEN и адресом приемника DSTADDR, замещается результатом.

Положительное значение счетчика вызывает умножение, а отрицательное - деление; при нулевом счетчике производится простая пересылка и установка кодов условий. При использовании отрицательного счетчика результат округляется при помощи операнда округления ROUND.

Примечания:

1. После выполнения:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-источника;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-приемника.

2. Строка-приемник, регистры с R0 по R3 и коды условий непредсказуемы, если строка-приемник перекрывает строку-источник; если строка-источник содержит неверный полубайт, или если происходит прерывание по резервному операнду.

3. Когда операнд-счетчик отрицателен, результат округляется десятичным сложением разрядов 3:0 операнда округления ROUND со старшими из отбрасываемых разрядов и происходит перенос в старшую цифру, если таковой возникает. Для этого сложения считается, что операнд-источник и операнд-округления имеют одинаковые знаки.

4. Если разряды 7:4 операнда округления не равны нулю, или если разряды 3:0 операнда округления содержат неверную цифру с точки зрения упакованного десятичного формата, результат непредсказуем.

5. Когда операнд-источник является положительным нулевым значением, операнд округления не влияет на результат, за исключением случая, описанного в примечании 4.

6. Обычно операнд округления равен 5. Можно выполнить усечение, если использовать операнд округления, равный нулю.

CMPP сравнение в упакованном формате

Формат:

код операции LEN.RW, SRC1ADDR.AB, SRC2ADDR.AB

!3 операнда

код операции SRC1LEN.RW, SRC1ADDR.AB, SRC2LEN 6.RW,

SRC2ADDR.AB !4 операнда

Коды условий:

N <- (строка-источник 1) LSS (строка-источник 2);

Z <- (строка-источник 1) EQL (строка-источник 2);

V <- 0;

C <- 0;

Исключительная ситуация:

резеpаный операнд

Коды операций:

35 CMPP3 сравнение (3 операнда) упакованных десятичных строк

37 CMPP4 сравнение (4 операнда) упакованных десятичных строк

Описание.

В трехоперандном формате строка-источник 1, специфицированная длиной LEN и адресом источника SRC1ADDR, сравнивается со строкой-источником SRC2ADDR, специфицированной длиной LEN и адресом источника SRC2ADDR. Единственным действием является установка кодов условий. В четырехоперандном формате строка-источник 1, специфицированная длиной

SRC1LEN и адресом источника SRC1ADDR, сравнивается со строкой-источником 2, специфицированной длиной SRC2LEN и адресом источника SRC2ADDR. Единственным действием является установка кодов условий.

Примечания:

1. После выполнения CMPP3 или CMPP4:

R0=0;

R1=адресу байта, содержащего старшую цифру строки 1;

R2=0;

R3=адресу байта, содержащего старшую цифру строки 2.

2. Регистры с R0 по R3 и коды условий непредсказуемы, если строки-источники перекрываются, если любая из строк содержит неверный полубайт, или если происходит прерывание по резервному операнду.

CVTLP преобразование длинного слова в упакованную строку

Формат:

код операции SRC.RL, DSTLEN.RW, DSTADDR.AB

Коды условий:

N <- (строка-приемник) LSS 0;

Z <- (строка-приемник) EQL 0;

V <- (десятичное переполнение);

C <- 0;

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

F9 CVTLP преобразование длинного слова в упакованную строку

Описание.

Операнд-источник SRC преобразуется в упакованную десятичную строку, и строка-приемник, специфицированная операндами длины DSTLEN и адреса приемника DSTADDR, замещаются результатом.

Примечания:

1. После выполнения:

R0=0;

R1=0;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-приемника.

2. Строка-приемник, регистры с R0 по R3 и коды условий непредсказуемы при прерывании по резервному операнду.

3. При перекрывании операндов получаются правильные результаты.

CVTPL преобразование упакованной строки в длинное слово

Формат:

код операции SRCLEN.RW, SRCADDR.AB, DST.WL

Коды условий:

N <- DST LSS 0;

Z <- DST EQL 0;

V <- (целочисленное переполнение);

C <- 0;

Исключительная ситуация:

резервный операнд

целочисленное переполнение

Коды операций:

36 CVTPL преобразование упакованной строки в длинное слово

Описание.

Строка-источник, специфицированная операндами длины SRCLEN и адреса SRCADDR, преобразуется в длинное слово, и операнд-приемник DST замещается результатом.

Примечания:

1. После выполнения:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-источника;

R2=0;

R3=0.

2. Операнд-приемник, регистры с R0 по R3 и коды условий непредсказуемы при прерывании по резервному операнду, или в том случае, когда строка содержит неверный полубайт.

3. Операнд-приемник записывается после того, как регистры смодифицированы, таким образом, регистры с R0 по R3 могут быть использованы в качестве операнда-приемника.

4. Если значение строки-источника выходит за пределы диапазона от -2147483648 до 2147483647, то происходит целочисленное переполнение, и операнд-приемник замещается млад-

шими 32-мя разрядами результата преобразования с правильным знаком. Таким образом, знак приемника может отличаться от знака источника.

5. При перекрывании операндов получаются правильные результаты.

CVTPS преобразование упакованной строки в числовую с ведущим знаком

Формат:

код операции SRCLEN.RW, SRCADDR.AB, DSTLEN.RW, DSTADDR.AB

Коды условий:

N <- (срока-источник) LSS 0;

Z <- (строка-источник) EQL 0;

V <- (десятичное переполнение);

C <- 0;

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

08 CVTPS преобразование упакованной строки в числовую с ведущим знаком

Описание.

Исходная десятичная строка в упакованном формате, специфицированная длиной SRCLEN и адресом источника SRCADDR, преобразуется в числовую строку с выделенным ведущим зна-

ком. Строка-приемник, специфицированная длиной DSTLEN и адресом приемника DSTADDR, замещается результатом.

Преобразование осуществляется заменой младшего адресуемого байта строки-приемника символами в КОИ-7 "+" или "-", определяемыми знаком строки-источника. Оставшиеся байты приемника замещаются символами КОИ-7, соответствующими упакованным десятичным цифрам в строке-источнике.

Примечания:

1. После выполнения:

R0=0;

R1=адресу байта, содержащего первую цифру строки-источника;

R2=0;

R3=адресу знакового байта строки-приемника.

2. Строка-приемник, регистры с R0 по R3 и коды условий непредсказуемы, если строка-источник содержит неверный полубайт, или если происходит прерывание по резервному операнду.

3. Эта инструкция заносит "+" или "-" в КОИ-7 в знаковый байт строки-приемника.

4. Если происходит десятичное переполнение, величина, запоминаемая в приемнике, может отличаться от указанной по кодам условий (разряды Z и N).

5. Если в результате преобразования получается -0 без переполнения, строка-приемник в числовом формате с лидирующим знаком заменяется на представление, соответствующее +0.

CVTPT. преобразование упакованной десятичной строки в числовую строку

Формат:

код операции SRCLEN.RW, SRCADDR.AB, TBLADDR.AB,
DSTLEN.RW, DSTADDR.AB

Коды условий:

N <- (строка-источник) LSS 0;

Z <- (строка-источник) EQL 0;

V <- (десятичное переполнение);

C <- 0;

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

24 CVTPT преобразование упакованной строки в числовую строку

Описание.

Строка-источник в упакованном формате, специфицированная длиной SRCLEN и адресом источника SRCADDR, преобразуется в числовую строку. Строка-приемник, специфицированная длиной DSTLEN и адресом приемника DSTADDR, замещается результатом. Коды условий N и Z устанавливаются по значению исходной упакованной десятичной строки.

Преобразование осуществляется путем использования старшего адресуемого байта (даже в том случае, когда значение строки-источника равно 0) строки-источника (то есть

байта, содержащего знак и младшую цифру) как индекс без знака в таблицу из 256 байтов, нулевой вход в которую задается операндом адреса таблицы TBLADDR. Прочитанный из таблицы байт замещает младший байт строки-приемника. Остальные байты замещаются представлениями в КОИ-7 величин, соответствующих десятичным цифрам исходной строки в упакованном формате.

Примечания:

1. После выполнения:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-источника;

R2=0;

R3=адресу старшей цифры строки-приемника.

2. Строка-приемник, регистры с R0 по R3 и коды условий непредсказуемы, если строка-приемник перекрывает строку-источник или таблицу; если строка-источник или таблица содержат неверный полубайт, или если происходит прерывание по резервному операнду.

3. Коды условий вычисляются по значению строки-источника даже в том случае, если происходит переполнение. В частности, код условий N устанавливается тогда и только тогда, когда источник ненулевой и содержит знак минус.

4. При соответствующем задании таблицы можно осуществить преобразование в любую форму числовой строки. Предпочтительную форму для перфо-формата, зонного формата или

представления без знака (см. раздел 2). Кроме того, можно так организовать таблицу, что будет осуществляться преобразование в абсолютную величину с обратным знаком или замена знака числа. К таблице преобразований можно обращаться даже в том случае, когда длина строки-приемника равна 0.

5. Десятичное переполнение происходит в том случае, когда строка-приемник не может вместить результат, преобразованный из ненулевой десятичной строки в упакованном формате (исключая ведущие нули). Преобразование строки-источника с нулевым значением никогда не приводит к переполнению. При преобразовании нулевой строки источника в строку-приемник нулевой длины происходит переполнение.

6. Если происходит десятичное переполнение, величина, запоминаемая в приемнике, может отличаться от величины, задаваемой кодами условий (разряды N и Z).

CVTSP преобразование числовой строки с ведущим знаком в упакованную строку

Формат:

код операции SRCLEN.RW, SRCADDR.AB, hoSTLEN.RW, DSTADDR.AB

Коды условий:

N <- (строка-источник) LSS 0;

Z <- (строка-источник) EQL 0;

V <- (десятичное переполнение);

C <- 0;

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

09 CVTSP преобразование числовой строки с ведущим знаком в упакованную

Описание.

Числовая строка-источник, специфицированная длиной SRCLEN и адресом источника SRCADDR, преобразуется в упакованную десятичную строку, и строка-приемник, специфицированная адресом DSTADDR и длиной приемника DSTLEN, замещается результатом.

Примечания:

1. Прерывание по резервному операнду происходит, если:

- длина исходной строки в числовом формате с ведущим знаком выходит за пределы диапазона от 0 до 31;
- длина приемника в упакованном десятичном формате выходит за пределы диапазона от 0 до 31;
- строка-источник содержит неверный байт. Неверным байтом будет любой символ, отличный от цифр в коде КОИ-7 от 0 до 9 или от знаков "+" или "-", "пробел" в КОИ-7 в знаковом байте.

2. После выполнения:

R0=0;

R1=адресу знакового байта строки-источника;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-прием-

ника.

3. Строка-приемник, регистры с R0 по R3 и коды условий непредсказуемы, если строка-приемник перекрывает строку-источник, или если возникает прерывание по резервному операнду.

CVTTP преобразование числовой строки в упакованную строку

Формат:

код операции SRCLEN.RW, SRCADDR.AB, TBLADDR.AB,
DSTLEN.RW, DSTADDR.AB

Коды условий:

N <- (строка-приемник) LSS 0;

Z <- (строка-приемник) EQL 0;

V <- (десятичное переполнение);

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

26 CVTTP преобразование числовой строки в упакованную строку

Описание.

Строка-источник в числовом формате длиной SRCLEN и адресом источника SRCADDR, преобразуется в упакованную десятичную строку, и строка-приемник, специфицируемая адресом DSTADDR и длиной DSTLEN заменяется результатом.

Преобразование выполняется путем использования старшего адресуемого байта строки-источника в качестве индекса без знака в таблицу из 256 байтов, нулевой вход в которую задается операндом адреса таблицы TBLADDR. Прочтенный из таблицы байт замещает старший адресуемый байт строки-приемника (то есть байт, содержащий знак и младшую значащую цифру). Оставшиеся упакованные цифры строки-приемника замещаются младшими четырьмя разрядами соответствующих байтов в исходной строке.

Примечания:

1. Прерывание по резервному операнду происходит, если:

- длина числовой строки-источника выходит за пределы диапазона от 0 до 31;
- длина упакованной десятичной строки-приемника выходит за пределы диапазона от 0 до 31;
- строка-источник содержит неверный байт. Неверный байт - это любая величина, отличная от представления в КОИ-7 цифр от 0 до 9 в любом байте, кроме самого младшего;
- преобразование младшей значащей цифры дает неверную упакованную десятичную цифру или знаковый полубайт.

2. После выполнения:

R0=0;

R1=адресу старшей цифры в строке-источнике;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-приемника.

3. Строка-приемник, регистры с R0 по R3 и коды условий

непредсказуемы, если строка-приемник перекрывает строку-источник или таблицу; или если происходит прерывание по резервному операнду.

4. Если инструкция преобразования выдает -0 без переполнения, упакованная десятичная строка заменяется на представление +0, код условий N сбрасывается, а Z устанавливается.

5. Если длина строки-источника равна нулю, упакованная десятичная строка-приемник устанавливается идентично в 0, а к таблице преобразования обращения не происходит.

6. Задав соответствующим образом таблицу, можно осуществить преобразовние из любой формы числовой строки (см. в разделе 2 предпочтительные формы для зонного, беззнакового и перфо-форматов). Кроме того, таблица может быть организована таким образом, чтобы обеспечивать преобразование в абсолютную величину, абсолютную величину с обратным знаком и изменение знака на обратный.

7. Если по таблице преобразования создается знаковый полубайт, содержащий любой разрешенный знак, в упакованной десятичной строке-приемнике сохраняется предпочтительное знаковое представление.

DIVP — деление строк в упакованном формате

Формат:

код операции DIVRLEN.RW, DIVRADDR.AB, DIVDLEN.RW,
DIVDADDR.AB, QUOLEN.RW, QUOADDR.AB

Коды условий:

N <- (строка-частное) LSS 0;

Z <- (строка-частное) EQL 0;

V <- (десятичное переполнение);

Исключительная ситуация:

резервный операнд

десятичное переполнение

деление на нуль

Коды операций:

27 DIVP деление строк в упакованном формате

Описание.

Строка-делимое, специфицированная длиной DIVDLEN и адресом DIVDADDR, делится на строку-делитель, специфицированную длиной DIVRLEN и адресом делителя DIVRADDR. Строка-частное, специфицированная длиной QUOLEN и адресом частного QUOADDR, замещается результатом.

Примечания:

1. Эта инструкция создает в стеке рабочую область из 16 байтов. После выполнения стек восстанавливается в первоначальное состояние, а содержимое [(SP)-16]: [(SP)-1] непредсказуемо.

2. Деление выполняется таким образом, что:

- абсолютная величина остатка (который теряется) меньше абсолютной величины делителя;
- произведение абсолютной величины частного на абсолютную величину делителя меньше или равно абсолютной величине делимого;

- знак частного определяется по правилам алгебры из знаков делимого и делителя. Если величина частного равна нулю, знак всегда положителен.

3. После выполнения:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-делителя;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-делимого;

R4=0;

R5=адресу байта, содержащего старшую цифру строки-частного.

4. Строка-частное, регистры с R0 по R5 и коды условий непредсказуемы, если строка-частное перекрывает строки делимого или делителя, если делитель или делимое содержат неверный полубайт, если делитель равен нулю, или если происходит прерывание по резервному операнду.

MOVP пересылка строк в упакованном формате

Формат:

Код операции LEN.RW, SRCADDR.AB, DSTADDR.AB

Коды условий:

N <- (строка-приемник) LSS 0;

Z <- (строка-приемник) EQL 0;

Исключительная ситуация:

резервный операнд

Коды операций:

34 MOVP пересылка строк в упакованном формате

Описание.

Строка-приемник, специфицированная операндами длины LEN и адреса приемника DSTADDR, замещается строкой-источником, специфицированной операндами длины LEN и адреса источника SRCADDR.

Примечания:

1. После выполнения:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-источника;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-приемника.

2. Строка-приемник, регистры с R0 по R3 и коды условий непредсказуемы, если строка-приемник перекрывает строку-источник, если строка-источник содержит неверный полубайт, или если происходит прерывание по резервному операнду.

3. Если источник равен -0, то результат равен +0, N сбрасывается, а Z устанавливается.

MULP умножение строк в упакованном формате

Формат:

код операции MULRLEN.RW, MULRADDR.AB, MULDLEN.RW,
MULADDR.AB, PRODLEN.RW, PRODADDR.AB

Коды условий:

N <- (строка-произведение) LSS 0;

Z <- (строка-произведение) EQL 0;

V <- (десятичное переполнение);

C <- 0;

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

25 MULP - умножение строк в упакованном формате

Описание.

Строка-множимое, специфицированная операндами длины MULDLEN и адреса MULDADDR, умножается на строку-множитель, специфицированую операндами длины MULRLEN и адреса множителя MULRADDR. Строка-произведение, специфицированная операндами длины PRODLEN и адреса произведения PRODADDR, заменяется результатом.

Примечания:

1. После выполнения:

R0=0;

R1=адресу байта, содержащего первую цифру строки-множителя;

R2=0;

R3=адресу байта, содержащего старшую цифру строки-множимого;

R4=0;

R5=адресу байта, содержащего старшую цифру строки-произведения.

2. Строка-произведение, регистры с R0 по R5 и коды условий непредсказуемы, если строка-произведение перекрывает строку-множимое или строку-множитель, если строка-множитель или строка-множимое содержат неверный полубайт, а также при прерывании по резервному операнду.

SUBR вычитание строк в упакованном формате

Формат:

код операции: SUBLEN.RW, SUBADDR.AB, DIFLEN.RW,
DIFADDR.AB !4 операнда

код операции: SUBLEN.RW, SUBADDR.AB, MINLEN.RW,
MINADDR.AB, DIFLEN.RW, DIFADDR.AB
!6 операндов

Коды условий:

N <- (строка-разность) LSS 0;

Z <- (строка-разность) EQL 0;

V <- (десятичное переполнение)

C <- 0;

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

22 SUBP4 вычитание (4 операнда) строк в упакованном формате

23 SUBP6 вычитание (6 операндов) строк в упакованном формате

Описание.

В четырехоперандном формате строка-вычитаемое, специфицированная операндами длины SUBLEN и адреса вычитаемого SUBADDR, вычитается из строки-разности, специфицированной операндами DIFLEN и DIFADDR, и строка-разность замещается результатом.

В шестиоперандном формате строка-вычитаемое, специфицированная операндами длины SUBLEN и адреса вычитаемого SUBADDR, вычитается из строки-уменьшаемого, специфицированной операндами длины MINLEN и адреса уменьшаемого MINADDR. Строка-разность, специфицированная операндами длины DIFLEN и адреса разности DIFADDR, замещается результатом.

Примечания:

1. После выполнения SUBP4:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-вычитаемого;

R2= адресу байта, содержащего старшую цифру строки-разности.

2. После выполнения SUBP6:

R0=0;

R1=адресу байта, содержащего старшую цифру строки-вычитаемого;

R2=0;

R3=адрес байта, содержащего старшую цифру строки-уменьшаемого;

R4=0;

R5=адрес байта, содержащего старшую цифру строки-разности.

3. Строка-разность, регистры с R0 по R3 (или с R0 по R5 для SUBP6) и коды условий непредсказуемы, если строка-разность перекрывает строки вычитаемое или уменьшаемое; если вычитаемое, уменьшаемое или разность (только для четырехоперандной инструкции) содержат неверный полубайт, а также в случае прерывания по резервному операнду.

4.13. Инструкция EDIT

Инструкция EDIT предназначена для выполнения обычных функций редактирования, необходимых при выдачи данных в фиксированном формате. Ее работа заключается в преобразовании десятичной строки в упакованном формате в символьную строку. Примером такого действия может служить оператор пересылки MOVE с редактированием (PICTURE) в КОБОЛ'е или PL/1 или FORMAT в ФОРТРАН'е, но эта инструкция может быть использована также и для других применений. Ее работа состоит в преобразовании входного десятичного числа в упакованном формате в выходную символьную строку, то есть, в порождении символов для вывода. При преобразовании цифр возможны такие функции, как:

- введение в число ведущих нулей;
- запрет ведущих нулей;
- вставка и распространение знака;
- вставка и распространение специальных знаков;
- вставка специального знакового представления;

- заполнение всего поля пробелами по нулевому значению.

Операндами для инструкции EDITPC являются: дескриптор входной десятичной строки в упакованном формате и спецификация шаблона, а также начальный адрес выходной строки. Дескриптор десятичной строки в упакованном формате - это стандартная для СМ 1700 пара операндов: длина десятичной строки в цифрах (до 31) и начальный адрес строки. Спецификация шаблона - есть начальный адрес редактирующей последовательности для обработки символов по шаблону, которая интерпретируется во многом сходным образом обычными инструкциями. Выходная строка описывается только начальным адресом, так как шаблон задает длину однозначно.

Во время выполнения инструкции EDITPC она использует два символьных регистра и четыре кода условий. Один символьный регистр содержит символ-заполнитель. Это обычно пробел в коде КОИ-7, но в целях контроля он может быть заменен на звездочку. Другой символьный регистр содержит знаковый символ. Первоначально там содержится либо пробел в КОИ-7, либо знак минус в зависимости от знака входной величины. Это знаковое представление можно менять на другие, такие, как плюс/минус или плюс/пробел, а также им можно манипулировать, чтобы получить специальные символы на выходе, такие, как <КС> после выполнения коды условий содержат знак входного числа (N), признак нулевого входного операнда (Z), условие переполнения (V) и наличие значащих цифр (C). Код условий N определяется в начале инструкции и после не

изменяется (кроме корректирования входа -0). Остальные коды условий вычисляются и модифицируются в процессе выполнения инструкции. По окончании выполнения EDITPC регистры R0 по R5 содержат величины, обычные для выполнения инструкции с десятичными значениями.

EDITPC редактирование упакованной строки в символьную

Формат:

код операции SRCLEN.RW, SRCADDR.AB, PATTERN.AB, DSTADDR.AB

Коды условий:

N <- (строка-источник) LSS 0; !N <- 0 IF SRC -0

Z <- (строка-источник) EQL 0;

V <- (десятичное переполнение); !ненулевые цифры !теряются

C <- (значимость);

Исключительная ситуация:

резервный операнд

десятичное переполнение

Коды операций:

38 EDITPC редактирование упакованной строки в символьную

Описание.

Строка-приемник, специфицированная операндами шаблона PATTERN адреса приемника DSTADDR, замещается отредактиро-

ванной версией строки-источника, специфицированной операндами длины SRCLEN и адреса источника SRCADDR. Редактирование выполняется в соответствии со строкой шаблона, начинающейся с адреса шаблона и до тех пор, пока не встретится оператор конца шаблона (EO¤END). Строка-шаблон состоит из однобайтных операторов. Для некоторых операторов не требуется операндов. Для некоторых требуется счетчик повторений, который содержится в самом правом полубайте самого оператора шаблона остальные используют однобайтовый операнд, который следует непосредственно за оператором шаблона. Этот операнд представляет собой либо беззнаковую целочисленную длину, либо символьный байт.

Примечания:

1. Прерывание по резервному операнду возникает, если длина источника больше 31.

2. Строка-приемник непредсказуема, если строка-источник содержит неверный полубайт, если операнд EO¤ADJUST_INPUT выходит за пределы диапазона от 1 до 31, если строка-источник перекрывает строку-приемник, или если перекрываются строка-шаблон и строка-приемник.

3. После выполнения:

R0=длине строки-источника;

R1=адресу байта, содержащего старшую цифру строки-источника;

R2=0;

R3=адресу байта, содержащего оператор шаблон EO¤END;

R4=0;

R5=адресу байта, следующего за последним байтом строки-приемника.

Если содержимое строки-приемника непредсказуемо, регистры с R0 по R5 и коды условий также непредсказуемы.

4. Если в конце V и DV установлен, вырабатывается прерывание по числовому переполнению, за исключением того случая, когда удовлетворяются условия в примечании 9.

5. Длина приемника задается операторами шаблона в строке-шаблоне. Если она неверно сформирована, если она модифицируется во время выполнения инструкции, длина строки-приемника непредсказуема.

6. Если источник равен -0, то результат также может быть равен -0 если только не включен оператор, задающий другое значение (EO¤BLANK_ZERO или EO¤REPLACE_SIGN).

7. Содержимое строки-приемника и памяти, предшествующей ей, непредсказуемы, если длина, на которую распространяется действие оператора EO¤BLANK_ZERO или EO¤REPLACE_SIGN, равна 0 или находится за пределами строки-приемника.

8. Если для шаблона требуется больше входных цифр, чем задано, то происходит прерывание по резервному операнду и R0=-1, R3=адрес оператора шаблона, когда потребовалась несуществующая цифра. Коды условий и остальные регистры описаны в примечании. Это прерывание не постоянно.

9. Если для шаблона требуется меньше входных цифр, чем задано, то вырабатывается прерывание по резервному операнду и R3=адресу оператора шаблона EO¤END. Коды условий и

регистры таковы, как описано в примечании. Это прерывание не постоянно.

10. При неверном или зарезервированном операторе шаблона вырабатывается ошибка по резервному операнду и R3=адрес оператора шаблона, вызвавшего ошибку. Коды условий и остальные регистры таковы, как описано в примечании. Эта ошибка существует до тех пор, пока состояние определенного регистра обрабатывается в соответствии с описанием оператора-шаблона, и сохраняется состояние, специфицированное как ??? (3 символа вопросительный знак).

11. При исключительной ситуации, вызванной резервным операндом, как описано в примечаниях с 8 по 10, FPD устанавливается, а коды условий и регистры принимают следующие значения:

N=(источник имеет знак минус)

Z=все встретившиеся до сих пор цифры источника равны 0

V=потеряны ненулевые цифры

C=значимость

R0=-нули<15:0>'остаток длины источника SRCLEN <15:0>

R1=текущий адрес в источнике

R2=???'знак'заполнитель

R3=адрес редактируемого оператора шаблона, вызвавшего исключительную ситуацию

R4=???

R5=адрес следующего байта приемника,

где "нули" = число нулей в источнике, которые необходимо добавить;

"знак" = текущее содержимое регистра знака;

"заполнитель" = текущее содержимое регистра заполнителя.

Краткое описание операторов шаблонов для инструкции EDIT приведено в табл. 9.

Таблица 9

Имя	Операнд	Краткое описание
вставить:		
	1)	
EDMINSERT	CH	вставить символ заполнитель или не значащая цифра
EDMSTORE_SIGN	-	вставить знак
	2)	
EDMFILL	R	вставить заполнитель
переслать:		
EDMMOVE	R	переслать цифру, заполнитель, если цифра не значащая
EDMFLOAT	R	пересылка цифр с распространением знака
EDMEND_FLOAT	-	распространение знака закончить
фиксировать		
	3)	
EDMBLANK_ZERO	LEN	установить заполнитель в обратном порядке по нулю

Продолжение табл. 9

Имя	Операнд	Краткое описание
ED#REPLACE_SIGN	LEN	заменить на символ-заполни-
		тель, если - 0
загрузить:		
ED#LOAD_FILL	CH	загрузка символа-заполнителя
ED#LOAD_SIGN	CH	загрузка символа-знака
ED#LOAD_PLUS	CH	загрузка символа-знака, если
		число положительно
ED#LOAD_MINUS	CH	загрузка символа-знака, если
		число отрицательно
управление		
ED#SET_SIGNIF	-	установить флаг значимости
ED#CLEAR_SIGNIF	-	сбросить флаг значимости
ED#ADJUST_INPUT	LEN	выравнить длину источника
ED#END	-	закончить редактирование

---

1) один символ;

2) счетчик повторений в диапазоне от 1 до 15;

3) длина в диапазоне от 1 до 255.

Кодирование оператора шаблона для EDIT представлено в табл. 10

Таблица 10

Код	Оператор
00	EO¤END
01	EO¤END_FLOAT
02	EO¤CLEAR_SIGNIF
03	EO¤SET_SIGNIF
04	EO¤STORE_SIGN
05..1F	резерв
20..3F	резерв
	1)
40	EO¤LOAD_FILL
	1)
41	EO¤LOAD_SIGN
	1)
42	EO¤LOAD_PLUS
	1)
43	EO¤LOAD_MINUS
	1)
44	EO¤INSERT
	2)
45	EO¤BLANK_ZERO
	2)
46	EO¤REPLACE_SIGN
	2)
47	EO¤ADJUST_INRUT

Продолжение табл. 10

Код	Оператор
48..5F	резерв
60..7F	резерв
80,90,A0	резерв
	3)
81..8F	EOпFILL
	3)
91..9F	EOпMOVE
	3)
A1..AF	EOпFLOAT
B0..FE	резерв
FF	резерв

1) символ расположен в следующем байте;

2) длина (без знака) расположена в следующем байте;

3) счетчик повторений, разряды <3:0>.

Каждый оператор шаблона описан способом, сходным с описаниями обычных инструкций. В каждом случае, если имеется операнд, он представляет собой либо счетчик повторений (R), от 1 до 15, либо беззнаковый байт (LEN), либо символьный байт (CH).

Кроме того, используются следующие определения:

Заполнитель=R2<7:0>

Знак=R2<15:8>

EO¤INSERT вставить символ

Цель:

вставить фиксированный символ, заменить символом-заполнителем, если символ не значащий

Формат:

шаблон CH

Оператор:

44 EO¤INSERT вставить символ

Описание.

За оператором-шаблоном следует символ. Если установлен флаг значимости, то символ помещается в приемник. Если флаг значимости не установлен, то содержимое регистра-заполнителя помещается в приемник.

Примечание. Этот оператор используется для вставки символов-разделителей (например, запятая) и фиксированных символов (например - косая черта). Для вставления фиксированных символов необходимо, чтобы флаг значимости был установлен (операторами EO¤SET_SIGNIF или EO¤END_FLOAT).

EO¤STORE_SIGN вставить знак

Цель:

вставить символ знака.

Формат:

маска

Оператор:

04. EO¤STORE_SIGN вставить знак

Описание.

Содержимое регистра знака помещается в приемник.

Примечание. Этот оператор используется для занесения любого арифметического знака, кроме знаков для чисел с плавающей запятой. Ему должны предшествовать операторы EO¤LOAD_PLUS и/или EO¤LOAD_MINUS, если нежелательна установка знака по умолчанию.

EO¤FILL вставить символ-заполнитель

Цель:

вставить символ-заполнитель.

Формат:

шаблон

Оператор:

8X. EO¤FILL вставить символ-заполнитель

Описание.

Правый полубайт оператора-шаблона является счетчиком повторений. Содержимое регистра-заполнителя помещается в приемник столько раз, каково значение счетчика повторений.

Примечание. Этот оператор используется для вставки символов-заполнителей.

00152-01 97 01-1

EO¤MOVE пересылка цифр

Цель:

пересылка цифр с заполнением незначащих (ведущих нулей) заполнителем

Формат:

шаблон

Оператор:

9X EO¤MOVE пересылка цифр

Описание.

Правый полубайт оператора шаблона является счетчиком повторений. Алгоритм исполняется число раз, заданное счетчиком. Очередная цифра пересылается из источника в приемник. Если цифра ненулевая, устанавливается значимость, а разряд Z сбрасывается. Если цифра незначащая (то есть ведущий нуль), она замещается в приемнике символом-заполнителем.

Примечания:

1. Если R больше, чем число цифр, оставшееся с строке-приемнике, вырабатывается прерывание по резервному операнду.

2. Этот оператор используется для пересылки цифр без распространения знака. Если требуется подавление ведущих нулей, бит значимости должен быть сброшен.

3. Если требуется контрольная защита (*), то перед шаблоном EO¤MOVE должен стоять шаблон EO¤LOAD_FILL.

EO¤FLOAT распространение знака

Цель:

Пересылка цифр с распространением знака через

Незначащие цифры

Формат:

Шаблон

Оператор:

AX. EO¤FLOAT распространение знака

Описание.

Правый полубайт оператора шаблона является счетчиком повторений. Следующий алгоритм выполняется число раз, заданное счетчиком повторений. Проверяется очередная цифра из источника. Если она ненулевая, и разряд значимости еще не установлен, тогда содержимое регистра знака запоминается в приемнике, разряд значимости устанавливается, а разряд Z сбрасывается. Если цифра значащая, она заносится в приемник, в противном случае в приемник заносится содержимое регистра символа-заполнителя.

Примечания:

1. Если R больше, чем число цифр, оставшихся в строке-приемнике, выполняется прерывание по резервному операнду.

2. Этот оператор шаблона используется для пересылки цифр с распространением знака. Так же, как и в случае EO¤STORE_SIGN, знак должен быть уже установлен. В последовательность из одного или более операторов EO¤FLOAT могут

включаться по очереди операторы EO¤INSERT и EO¤FILL. Перед выполнением первого оператора шаблона в последовательности разряд значимости должен быть сброшен. Последовательность должна заканчиваться оператором EO¤END_FLOAT.

3. Этот оператор используется для пересылки цифр с распространением знака доллара. Этот знак должен быть уже установлен оператором EO¤LOAD_SIGN. В последовательности из одного и более операторов EO¤FLOAT могут встречаться операторы EO¤INSERT и EO¤FILL. Перед первым оператором в последовательности разряд значимости должен быть сброшен. Последовательность должна заканчиваться одним оператором EO¤END_FLOAT.

EO¤END_FLOAT конец распространения знака

Цель:

завершение операции распространения знака

Формат:

шаблон

Оператор:

01 EO¤SEND_FLOAT конец распространения знака

Описание.

Если распространяющийся знак до сих пор не был помещен в приемник (то есть, если разряд значимости не установлен), то содержимое регистра знака сохраняется в приемнике, а разряд значимости устанавливается.

Примечание. Этот оператор используется после последо-

вательности из одного или более операторов шаблона EO¤FLOAT, которая начинается со сброса разряда значимости. Последовательность операторов EO¤FLOAT может содержать операторы EO¤INSERT и EO¤FILL.

EO¤BLANK_ZERO по нулю заполнить пробелами в обратном порядке

Цель:

заполнение приемника по нулевому значению пробелами.

Формат:

шаблон LEN

Оператор:

45 EO¤BLANK_ZERO по нулю заполнить пробелами

Описание.

За оператором шаблона следует операнд длины, выраженный целочисленной величиной без знака. Если значение строки-источника равно нулю, то содержимое регистра символа-заполнителя заносится в последние байты строки-приемника.

Примечания:

1. Длина должна быть ненулевая и находится в пределах уже созданной строки-приемника. В противном случае содержимое строки-приемника и ячеек памяти перед ней непредсказуемо.

2. Этот оператор используется для подавления любых символов, записанных в приемник, например таких, как знак,

или цифры, следующие после десятичной точки.

EO¤REPLACE_SIGN заменить знак по нулю

Цель:

по нулю зафиксировать определенный знак приемника.

Формат:

шаблон LEN

Оператор:

46 EO¤REPLACE_SIGN заменить знак по нулю

Описание.

За оператором шаблона следует операнд-длина, выраженный целочисленной байтовой величиной без знака. Если значение строки-приемника равно нулю (то есть, если Z установлен), то содержимое регистра-заполнителя заносится в тот байт строки-приемника, который находится перед текущим положением на число байтов, равное длине.

Примечания:

1. Длина должна быть ненулевая и в пределах уже созданной строки-приемника в противном случае содержимое строки-приемника и ячеек памяти перед ней непредсказуемо.

2. Этот оператор можно использовать для корректировки занесенного знака (eo¤END_FLOAT или EO¤STORE_SIGN) в том случае, если был занесен минус, а значение источника оказалось равным нулю.

EO¤LOAD загрузка регистра

Цель:

поменять значение регистра знака или регистра-заполнителя

Формат:

шаблон CH

Оператор:

40 EO¤LOAD_FILL загрузить регистр символа-заполнителя

41 EO¤LOAD_SIGN загрузить регистр знака

42 EO¤LOAD_PLUS загрузить регистр знака по плюсу

43 EO¤LOAD_MINUS загрузить регистр знака по минусу

Описание.

За оператором шаблона следует символ. Для EO¤LOAD_FILL этот символ помещается в регистр символа заполнителя. Для EO¤LOAD_SIGN этот символ помещается в регистр знака. Для EO¤LOAD_PLUS этот символ помещается в регистр знака в том случае, если строка-источник имеет положительный знак. Для EO¤LOAD_MINUS этот символ помещается в регистр знака в том случае, если строка-источник имеет отрицательный знак.

Примечания:

1. EO¤LOAD_FILL используется для установки контрольной защиты (* вместо пробела).

2. EO¤LOAD_SIGN используется для установки распространения знака "¤".

3. EO¤LOAD_PLUS используется для установки печати

положительного знака вместо пробела.

4. EO¤LOAD_MINUS используется для установки отрицательного знака.

EO¤SIGNIF установление флага значимости

Цель:

управление индикатором значимости

Формат:

шаблон

Оператор:

02 EO¤CLEAR_SIGNIF сброс флага значимости

03 EO¤SET_SIGNIF установка флага значимости

Описание.

Устанавливается или сбрасывается разряд значимости. Это осуществляет управление ведущими нулями (ведущими нулями здесь называются ведущие цифры, для которых индикатор значимости сброшен).

Примечания:

1. EO¤CLEAR_SIGNIF используется для инициализации подавления ведущих нулей (EO¤MOVE) или для распространения знака (EO¤FLOAT), следующего за вставленным символом (EO¤INSERT с установлением значимости).

2. EO¤SET_SIGNIF используется для того, чтобы избежать подавления ведущих нулей (перед eo¤MOVE).

EO¤ADJUST_INPUT настроить длину

Цель:

регулировка строки по длине, если входная и выходная строки имеют различные длины

Формат:

шаблон LEN

Оператор:

47 EO¤ADJUST_INPUT настроить длину

Описание.

За оператором шаблона следует символ, который представляет собой целое без знака в виде байта и обозначает длину в диапазоне от 1 до 31. Если входная строка имеет цифр больше, чем указано, то лишние ведущие цифры читаются и отбрасываются. Если любая из отброшенных цифр не нуль, тогда разряд переполнения устанавливается, разряд значимости устанавливается, разряд Z очищается. Если входная строка имеет меньше цифр, чем указано в длине строки, временный счетчик устанавливается на количество ведущих нулей. Этот счетчик хранится как отрицательное число в R0<31:16>.

Примечание. Если длина не лежит в диапазоне от 1 до 31, выходная строка, коды условий и содержимое регистров от R0 до R15 непредсказуемо.

EO¤END конец редактирования

Цель:

конец операции редактирования

Формат:

шаблон

Оператор:

DD EOMEND конец редактирования

Описание.

Оператор является заключительным ограничителем операции редактирования, описываемой инструкцией EDITPC.

Примечания:

1. Если существуют еще не обработанные входные цифры, возникает ошибка резервного операнда.

2. Если входное значение было - 0, то разряд состояния N очищается.

4.14. Прочие инструкции СМ 1700

В документе [1] описаны следующие инструкции:

- в разделе 1: PROBE (READ, WRITE) проверка допустимости (R-чтения, W-записи) PROBE(R,W) MODE.RB, LEN.RW, BASE.AB;
- в разделе 2: сменить режим CHM(K,E,S,U) PARAM.RW, [-(YSP).W*] и возврат из исключительной ситуации или прерывания REI [(SP)+.R*]
- в разделе 3: загрузить контекст процессора LDPCTX [PCB.R*, -(KSP).W*] и сохранить контекст процессора SVPCTX [(SP)+.R*, PCB.W*];
- в разделе 4: записать в привилегированный регистр MTPR SRC.RL, PROCREG.RL и читать из привилегированного регистра MFPR PROCRED.RL, DST.WL.

BUG запрос на сообщение об ошибках

Формат:

код операции MESSAGE.BX

Коды условий:

N <- N;

Z <- Z;

V <- V;

C <- C;

Исключительная ситуация:

резервная инструкция

Коды операций:

FEFF	BUGW	запрос на сообщение об ошибках с идентификатором сообщения размером в слово или длинное слово
FDFF	BUGL	

Описание.

Аппаратура рассматривает этот код операции как резервный и вырабатывает прерывание. Операционная система считает ее запросом на вывод сообщений об обнаруженных программных ошибках. Строчный идентификатор сообщений расширяется нулями до длинного слова (BUGW) и интерпретируется как условная величина.

Если процесс является привилегированным по отношению к выдаче сообщений об ошибках, делается запись в системном журнале. Если процесс непривилегированный, выдается сообщение о резервной инструкции.

Перечень ссылочных документов

1. Операционная система МОС ВП. Архитектура и система машинных инструкций. Справочный материал. 00152-01 97 01-2

Лист регистрации изменений

Номера листов (страниц)					Всего листов (страниц) в докум.	N документа	Входящий N сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					

Министерство приборостроения, средств автоматизации

и систем управления

Операционная система МОС ВП

АРХИТЕКТУРА И СИСТЕМА МАШИННЫХ ИНСТРУКЦИЙ

Справочный материал

Лист утверждения

26.00152-01 97 01-1-ЛУ

Представители предприятия-разработчика

Руководитель темы
Зам. директора

_____ Хрущев С.Н.
"___"_____ 1987 г.

Ответственный исполнитель темы
Зав. отделением

_____ Остапенко Г.П.
"___"_____ 1987 г.

Зав. лабораторией

_____ Аксенов А.В.
"___"_____ 1987 г.

Научный сотрудник

_____ Суслов С.Н.
"___"_____ 1987 г.

Ст.техник

_____ Нефедова Н.Х.
"___"_____ 1987 г.

Нормоконтролер

_____ Скворцов В.А.
"___"_____ 1987 г.

1987

Перв. примен.
26.00152-01

Литера О