КОМПЛЕКС ВЫЧИСЛИТЕЛЬНЫЙ СМ 1600

Заводской № 902 Год выпуска 1986

Техническое описание

Архитектура и система команд

ведущего процессора

1.700.010 ТО1

Часть 2

Книга № 2

УТВЕРЖДЕН

I.700.0IOTO-ЛУ

КОМПЛЕКС ВЫЧИСЛИТЕЛЬНЫЙ

СМ1600

Техническое описание

Часть 2

Архитектура и система команд ведущего процессора

I.700.0IOTOI

I983

Инв.№ подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
16-2078	03.4.6.			

АННОТАЦИЯ

Настоящий материал содержит описание архитектуры двухпроцессорного комплекса СМ1600. В нем изложены основные принципиальные решения, принятые при создании оборудования комплекса, которые не только обуславливают схемное и конструктивное его исполнение, но и должны также учитываться при разработке программных средств, в частности операционных систем этого комплекса.

Материал изложен в двух частях; в первой содержится описание архитектуры комплекса и система команд ведущего процессора, во второй - система команд специального процессора.

Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата | Перв. примен. | Справ. №

Изм.	Стр.	№ докум.	Подп.	Дата
Разраб.				23.11.84
Пров.				23.03.85
Н. контр.				23.04.85
Утв.				

I.700.0I0T0I

ВК СМ1600
Техническое описание
Часть 2
Архитектура и система команд ведущего процессора

Лит.	Стр.	Страниц
	3	II9

Копировал а Формат А4

СОДЕРЖАНИЕ

Стр.

1. Введение 5
2. Структура комплекса СМ1600 6
3. Система прерываний 8
4. Системный интерфейс и распределение адресов памяти 11
5. Ведущий процессор 21
   5.1. Общие сведения 21
   5.2. Стек 22
   5.3. Слово состояния процессора 23
   5.4. Внутренние прерывания 25
   5.5. Диспетчер памяти 26
   5.6. Система команд 33
      5.6.1. Общие сведения 33
      5.6.2. Режимы адресации 34
      5.6.3. Работа с байтами 41
   5.7. Группа одноадресных команд 42
   5.8. Группа двухадресных команд 48
   5.9. Команды программного управления 55
   5.10. Команды оперативной группы 65
   5.11. Операторы условного кода 68
   5.12. Команды плавающей запятой 70
      5.12.1. Общие сведения 70
      5.12.2. Формат данных 73
      5.12.3. Команды 83
      5.12.4. Примеры программирования 116
      Приложение 1 118

I. ВВЕДЕНИЕ

Настоящий документ является описанием архитектуры вычислительного комплекса СМI600.

К понятию архитектуры комплекса относится структура обрабатывающего устройства, структура и организация оперативного запоминающего устройства и принципы организации взаимодействия между устройствами в комплексе (системный интерфейс).

СМI600 является вычислительным комплексом, обрабатывающее устройство которого состоит из двух процессоров различного типа, называемых в дальнейшем ведущим и специальным (спецпроцессором). Эти процессоры являются самостоятельными функциональными и конструктивными единицами комплекса и имеют различные системы команд.

Взаимодействие между обоими процессорами осуществляется через механизм прерываний: каждый процессор может выдать сигнал прерывания программы другого процессора и потребовать внимания для выполнения определенной внутрисистемной процедуры.

2. СТРУКТУРА КОМПЛЕКСА СМ1600

Структура комплекса СМ1600 показана на рис. I. Комплекс построен по модульному принципу. Все основные функциональные модули комплекса имеют по одному выходу на общую шину (интерфейс "Общая шина" описан в I.700.0I0T03), по которой происходит обмен информацией между данным модулем и другими модулями комплекса. Исключение составляет устройство оперативной памяти, которое имеет два идентичных выхода; один - на общую шину, другой непосредственно в спецпроцессор. Отдельный выход на спецпроцессор служит для повышения производительности обрабатывающего устройства в целом, поскольку позволяет одновременную (параллельную работу обоих процессоров.

Оперативная память состоит из 4-х кубов, по 64 Кбайт каждый. Ведущий процессор может обращаться по любому адресу ОЗУ, а спецпроцессор из-за ограниченной длины адресного слова - только в пределах одного куба. Передача управления или пересылка данных за пределы блока осуществляется спецпроцессором только через управляющие программы, с помощью системы приоритетных прерываний. Аналогично, с помощью этой же системы приоритетных прерываний (арбитра ведущего процессора) решается, какому из модулей вычислительного комплекса следует занять шину в определенный момент времени, поскольку шина является общей для обоих процессоров памяти и внешних устройств.

Ф26 ГОСТ 2.104-68

Структура комплекса СМ1600
Интерфейс ввода-вывода

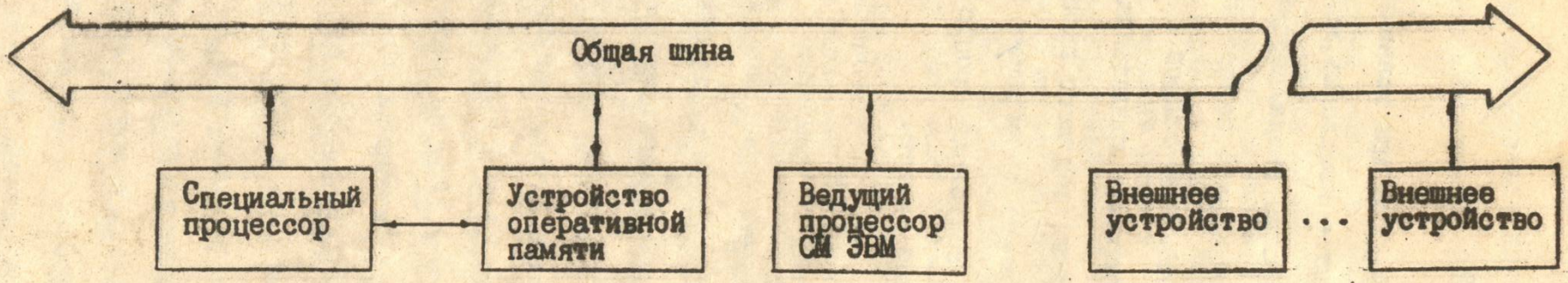


Рис. I

3. СИСТЕМА ПРЕРЫВАНИЙ

Суть системы прерываний состоит в том, что при возникновении определенного события в каком-либо устройстве комплекса, например, окончания операции обмена данными в устройстве ввода/вывода, это устройство по специальной сигнальной линии выдает запрос на прерывание. Аппаратура прерывания идентифицирует этот запрос и обеспечивает прекращение выполнения текущей программы процессора, запоминание адреса следующей команды и состояния ведущего процессора в стек и передачу управления другой программе (программе обслуживания прерывания), адрес которой и новое состояние процессора хранится в фиксированном месте оперативной памяти.

Каждому устройству, подключенному к шине, в том числе и обоим процессорам, присваивается определенный приоритет.

Имеется 5 уровней приоритета: один внепроцессорный и четыре программных. Внепроцессорный уровень используется для обмена информаций с устройствами, для которых прием и передача данных связаны с физическим перемещением носителя, причем невозможен его останов в произвольном месте; такими устройствами являются магнитный диск или магнитная лента. Принцип обмена информацией для таких устройств называется прямым доступом. Другим внешним устройством и спецпроцессору назначаются фиксированные программные приоритеты, а принцип обмена информацией называется соответственно программным. Ведущий процессор по отношению к шине рассматривается также как и любое другое внешнее устройство, но с изменяемым приоритетом. Уровень его приоритета задается разрядами 5,6,7 регистра слова состояния ведущего процессора и может быть изменен программным путем. Обычно ведущему процессору присваивается самый низкий приоритет и поэтому любое другое устройство может вызвать

прерывание его работы. Повышение его приоритета приводит к блокировке запросов на прерывание, поступающих от внешних устройств с приоритетом того же уровня и ниже.

Внепроцессорный (или внепрограммный уровень) не может быть заблокирован процессорами. Запросы на прерывание по внепроцессорному уровню не прерывают работу процессоров, а только приостанавливают их, причем ведущий процессор приостанавливается в любом случае, а спецпроцессор - только в случае, если он работает с адресами памяти, размещающимися в том же кубе, что и поле обмена с устройством, работающим по внепроцессорному уровню приоритета.

Спецпроцессор также имеет средства маскирования запросов на прерывание спецпроцессора. Он может быть заблокирован установкой значений разряда 8 регистра управления и состояния RCSS (см, распределение адресов памяти), равным нулю.

Каждое внешнее устройство и спецпроцессор снабжаются аппаратным указателем, называемым вектором прерывания, в виде фиксированной пары ячеек оперативной памяти, которые определяют адрес начала подпрограммы обслуживания устройства.(Конкретные адреса векторов прерываний приведены в следующем параграфе). Аппаратура обработки прерывания автоматически передает управление обслуживающей программе, относящейся к конкретному устройству, исключая при этом необходимость анализа запросов на прерывание.

Приоритеты прерываний от спецпроцессора и внешних устройств и приоритет ведущего процессора независимы. Это позволяет изменять обработку прерываний в реальном времени в зависимости от внешних условий, т.е. выполнение обслуживающих программ может продолжаться или прерываться. Аппаратура системных прерываний непрерывно сравнивает текущий приоритет ведущего процессора с уровнями запросов на обслуживание и удовлетворяет запросы от устройств с более вы-

соким приоритетом. Обслуживание некоторого устройства может быть прервано для обслуживания устройства с более высоким приоритетом и продолжено после его окончания.

К каждому из приоритетных уровней может быть подключено неограниченное (в архитектурном смысле) число внешних устройств. Среди них более высокий приоритет получают устройства, расположенные ближе к процессору.

Кроме прерываний от внешних устройств и спецпроцессора для ведущего процессора и спецпроцессора существуют еще и внутренние прерывания, о которых подробнее описывается при описании внутренней структуры процессоров.

Здесь укажем лишь, что внутренние прерывания для обоих процессоров имеют более высокий приоритет, чем внешние. Если при выполнении программы обработки некоторого прерывания приходит запрос на прерывание более высокого приоритета, то запоминается информация о первой программе и выполнение ее откладывается на время обслуживания этого нового прерывания.

Устройства, входящие в состав СМ1600, имеют следующие уровни приоритетов:

Устройство	Приоритет	Значение бит 5/7
Магнитный диск СМ5408	5	101
при обмене данными	Внепроцессорный	-
Магнитная лента СМ5301	5	101
при обмене данными	Внепроцессорный	
Печатающее устройство СМ6315	4	100
Устройство считывания перфокарт	6	110
Устройство ввода/вывода перфолент	4	100

Устройство	Приоритет	Значения бит 5/7
Видеотерминал СМ7204	4	I00
Спецпроцессор	5	I0I
Ведущий процессор	4/7	I00/III

4. СИСТЕМНЫЙ ИНТЕРФЕЙС И РАСПРЕДЕЛЕНИЕ АДРЕСОВ ПАМЯТИ

Как упоминалось выше, в комплексе СМ1600 все устройства – ведущий процессор, спецпроцессор, оперативное запоминающее устройство и периферийные устройства – подключаются к единственной в системе магистрали передачи информации, называемой ОБЩЕЙ ШИНОЙ(ОШ).

Физически ОШ представляет собой унифицированную магистраль из 56 функционально-объединенных линий, по которым передается вся необходимая для функционирования комплекса информация. Имеется также ряд вспомогательных линий.

Структура с одношинным интерфейсом обеспечивает единый метод связи для всех устройств системы, включая оперативную память и устройства прямого доступа. Использование единственного асинхронно действующего канала ОШ позволяет иметь общий для всех устройств комплекса алгоритм связи и, следовательно, унифицированную аппаратуру сопряжения.

Процессор использует установленный набор сигналов интерфейса как для связи с памятью, так и для связи с периферийными устройствами. Последние также используют этот набор сигналов, когда устанавливают связь с процессором, памятью или другими периферийными устройствами, подключенными к ОШ.

Одношинная структура интерфейса обеспечивает общую схему адресации внутренних регистров процессора, регистров периферийных

устройств и ячеек оперативной памяти, что позволяет использовать весь комплект адресных команд процессора для выполнения операций ввода-вывода.

В состав интерфейса периферийных устройств (ПУ) входят регистры-источники и/или приемники информации. В соответствии с архитектурой каждому регистру устройства присваивается свой адрес, отличающий его от других регистров, периферийных устройств, подключенных к ОШ (см. распределение памяти). Этот адрес аналогичен адресу ячейки памяти. Структура системы с общей магистралью ввода-вывода позволяет процессору рассматривать регистры ПУ как активные ячейки оперативной памяти и обращаться к ним с помощью команд, в силу чего не требуются специальные команды ввода-вывода.

Достоинство принятой схемы обращения к периферийным устройствам состоит в том, что ведущий процессор может оперировать с данными из регистров этих устройств непосредственно, т.е. без предварительной пересылки их в память или свои регистры, используя для этого весь набор команд и все возможности режимов адресации. Данные также могут передаваться от одного буферного регистра в другой через ОШ, обходя ведущий процессор (его регистры) полностью. В силу этого ряд прерываний не возникает и общая производительность процессора увеличивается.

Процедура взаимодействия устройства в системе с одношинным интерфейсом такова, что в любой операции обмена сигналами по ОШ всегда участвуют два устройства, связанные между собой как ЗАДАТЧИК (управляющее устройство) и ИСПОЛНИТЕЛЬ (управляемое устройство).

ЗАДАТЧИК управляет работой ШИНЫ при обмене данными с другим устройством, называемым исполнителем. Двух и более, работающих одновременно на ОШ, задатчиков не может быть.

Примером взаимосвязи двух устройств могут служить:

процессор (задатчик), извлекающий команду из оперативной памяти (исполнитель).

Оперативная память в СМ1600 всегда является исполнителем.

ОШ используется процессорами и всеми периферийными устройствами с разделением во времени в соответствии с системой приоритетов устройств, описанной выше. Какому из устройств занимать ШИНУ в текущий момент времени решает АРБИТР процессора (схема управления приоритетными прерываниями).

Устройства могут запрашивать управление ШИНОЙ по линиям запросов передачи ОШ, поступающим в арбитр. Запрос удовлетворяется, если его приоритет выше всех остальных. Новый задатчик принимает управление ШИНОЙ на себя когда текущий задатчик освободит ее. Новый задатчик затем может вызвать прерывание работы процессора или же начать передачу информации без участия процессора.

В принципе, каждое устройство на ОШ, кроме устройств оперативной памяти, может становиться задатчиком.

По отношению к интерфейсу ввода-вывода ведущий процессор также рассматривается как периферийное устройство, но с изменяемым приоритетом (тогда как приоритеты периферийных устройств фиксированы). Уровень приоритета процессора задается с помощью трех разрядов регистра слова состояния ведущего процессора и может изменяться программно.

В соответствии со своей структурой интерфейс ОШ обеспечивает три возможности обмена данными между устройствами.

Первая возможность заключается в том, что каждое передаваемое слово (или байт) данных обрабатывается программой ведущего процессора. При этом процессор занят только выполнением программы обслуживания выбранного устройства (ее приоритет устанавливается в этом режиме максимальным, например, седьмым) и сам обмен проис-

ходит на максимальной программной скорости, определяемой временем выполнения используемых команд. Описанная возможность особенно эффективна для организации программной передачи массивов данных.

Вторая возможность обслуживания работы устройств связана с прерыванием фоновой программы ведущего процессора. При этом время выполнения цикла передачи слова (или байта) данных определяется суммой времени перехода на обслуживающую программу, выполнения самой программы и возврата на фоновую программу. Скорость обмена по такому программному каналу ниже, чем в первом случае.

Третья возможность предусматривает передачу данных между устройствами системы напрямую, без участия процессора (канал прямого доступа). Скорость обмена определяется пропускной способностью собственно интерфейса ОШ и быстродействием участвующих в обмене устройств.

При обмене пс ОШ используется асинхронный метод связи. Суть его состоит в том, что каждый управляющий сигнал, выданный задатчиком, обязательно должен подтверждаться ответным сигналом исполняющего устройства, после чего обмен продолжается до завершения всей процедуры передачи данных. В случае неполучения ответного сигнала от исполнителя, задающее устройство фиксирует ошибку в работе интерфейса.

Большинство линий магистрали ОШ служат для двухсторонней передачи сигналов. К таким линиям все устройства комплекса, в том числе и процессоры подключаются параллельно по схеме "проводное или". Это означает, что входные провода могут действовать и в качестве выходных, поэтому один и тот же регистр периферийного устройства может использоваться как для ввода, так и для вывода данных. Самые старшие 4 Кслов адресного пространства (адреса 777776 по 760000) резервированы для регистров периферийных устройств. Естественно, что ряд этих адресов представляет собой резерв либо

не используются в СМ1600.

По адресам, начиная с нулевого до 377, расположена зона векторов прерываний, каждый из которых состоит из пары слов,хранящих адрес программы обработки прерывания и состояние ведущего процессора; в эту же зону входят векторы, так называемых, "ловушек",хранящих информацию, аналогичную векторам прерываний, для различных "непланируемых", но возможных событий таких, как колебания или отказы питания, сбои памяти и т.п. Остальная часть памяти отводится для стека ведущего процессора и под общую память для программ и данных. Ее адреса от 400 до границы зоны регистров, т.е. 757776.

Ниже приводится детальное распределение адресов общей шины.

I. Адреса векторов прерывания и ловушек.

000 резерв

004 исход времени, нечетный адрес и переполнение стека ведущего процессора

010 нелегальные или резервные команды ведущего процессора

014 внутреннее прерывание по биту Т (ВРТ)

020 внутреннее прерывание по команде IOT

024 отказ питания

030 внутреннее прерывание по команде EMT

034 внутреннее прерывание по команде TRAP

040–054 четыре вектора системного обеспечения

060 – клавиатура дисплея

064 экран дисплея

070 устройство ввода с перфоленты

074 устройство вывода на перфоленту

100 часы реального времени (таймер)

104 резерв (для программируемых часов)

110 "

114 ошибка памяти при работе ведущего процессора

I20-I64 не используются в СМ1600

I70 спецпроцессор

I74 печатающее устройство СМ63I5 (второе)

200 печатающее устройство СМ63I5

204 не используется

2I0 устройства магнитных дисков СМ5408

2I4-220 не используются

224 стандартная магнитная лента СМ5302

230 считыватель перфокарт Р6I0

234-240 не используется

244 ошибка при работе с плавающей запятой

250 ошибка диспетчера памяти

254 не используется

260 резерв для устройства кассетного НМД

264 резерв для накопителя на гибких НМД

266-374 не используются

2. Адреса регистров устройств (в порядке убывания)

777776 слово состояния ведущего процессора PS

777774-777750 - не используются

777746-777744 - регистры управления и состояния памяти

746 регистр управления

744 регистр состояния

777742-777720 не используются

777716-777710 регистры ведущего процессора

777707-777700 общие регистры

707 R7 (счетчик адресов PC)

708 R6 (указатель стека SP)

705 R5

704 R4

703 R3

702 R2

700 RI

700 R0

777676–777660 не используются

777656–777640 регистры адресов страниц пользователя

777636–777620 не используются

777616–777600 регистры описаний страниц пользователя

777576–777572 регистры состояния диспетчера памяти

777570 регистр пульта ведущего процессора

777566–777560 дисплей

566 регистр данных вывода

564 регистр состояния вывода

562 регистр данных ввода

560 регистр состояния ввода

777556–777550 устройство ввода–вывода перфоленточное

556 регистр данных вывода

554 регистр состояния вывода

552 регистр данных ввода

550 регистр состояния ввода

777546 регистр состояния часов реального времени (таймер)

777544–777520 не используются

775516–775514 печатающее устройство

516 данные вывода

514 состояние вывода

777512–777504 не используются

777502–777500 резерв для устройства кассетной магнитной ленты

502 данные

500 состояния

777476–777440 зона адресов диска СМ5408

476 регистр обслуживания 3 (RKMR3)

474 регистр обслуживания 2 (RKMR2)

472 регистр кода коррекции ошибок (RKECPT)

470 регистр положения ошибки (RKECPS)

466 регистр обслуживания I (RKMR1)

464 регистр буфера данных (RKDB)

462 не используется

460 регистр цилиндра (RKDC)

456 регистр внимания и смещения (RKAS/OF)

454 регистр ошибок (RKER)

452 регистр состояния накопителя (RKDS)

450 регистр команд и состояния 2 (RKCS2)

446 регистр дорожки и сектора (RKDA)

444 регистр адреса шины (RKBA)

442 регистр счетчиков слов (RKWC)

440 регистр состояния и управления I (RKCS1)

777436-777420 не используются

777316-777300 зона адресов расширителя арифметики

316 арифметический сдвиг

314 логический сдвиг

312 нормализация

310 счетчик шаго/регистр состояния

306 множимое

304 произведение частное

302 аккумулятор

300 делитель

777276-777174 не используется

777172-777170 накопитель на гибком МД СМ5603

172 буферный регистр данных

170 регистр команд и состояний

777166 не используется

777164–777160 считыватель с перфокарт P6I0

I64 8-битный регистр данных

I62 I2-битный регистр данных

I60 регистр команд и состояния

777156–776700 не используются

776676–776500 асинхронный адаптер связи БСАДС СМ850I (DLII)

Примечание. Адреса 776500–776526 временно используются для подключения 3-х местных дисплеев.

776476–774000 не используются

773776–773000 часть аппаратного загрузчика и первичной диагностики

772776–772536 не используется

772534–772520 накопитель на магнитной ленте СМ530I (TMII)

534 регистр обслуживания

532 регистр линий считывания

530 буферный регистр

526 регистр текущего адреса памяти

524 счетчик байт

522 регистр команд

520 регистр состояния

772516–772360 не используются

772356–772340 регистры адресов страниц системы

772356–772340 регистры адресов страниц системы

772336–772320 не используются

772316–772300 регистры описания страниц системы

770456–770454 специальный процессор

456 - регистр дескриптора состояния спецпроцессора RSDP

454 - регистр управления и состояния RCSS

Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата

16-2978 | 23.4.6.

770452–766000 не используются

765776–765000 эмулятор консоли и часть первичных тестов

764776–760020 не используются

764016–764014 печатающее устройство (второе)

764012–764010 не используется

760006–760000 диагностический резерв

Для СМ ЭВМ принято единое или общее распределение адресов общей шины для всех устройств, входящих в номенклатуру СМ ЭВМ. Приведенное выше распределение соответствует общему распределению для СМ ЭВМ, но в нем указаны только те устройства, которые в настоящий момент включены, или которые будут включены в ближайшем будущем в состав СМ1600. Поэтому при выборе адресов для новых устройств данным распределением пользоваться не рекомендуется, поскольку адреса, указанные в нем, как неиспользуемые, могут быть уже выделены.

Назначение отдельных регистров, отведенных для устройств, выполняемые ими функции описываются подробно в описаниях соответствующих устройств.

Ниже приводится сводная таблица для устройств, входящих в типовой состав комплекса СМ1600, с указанием основных данных по связи их с системой (табл. I).

Таблица I

Характеристика устройства	Мнемоническое имя		Режим обмена	Приоритет прерывания	Адрес вектора	Регистры
	русский	английский				
Ведущий процессор	СМ1600.2620	–	ПД	4/7		Описаны отдельно
Спецпроцессор	СМ2104.0506	–	ПД	5	170	770456–454
НМД	СМ5408	РК06	ПД	5	210	777476–440
НМЛ	СМ5301	TVII	ПД	5	224	772534–520

Продолжение табл. I

Характеристика устройства	Мнемоническое имя		Режим обмена	Приоритет прерывания	Адрес вектора	Регистры
	русское	английское				
АЦПУ	СМ6315	LPII	ПР	4	200	777516-514
Считыватель п/к	Р6I0	CRII	ПР	6	230	777166-160
Ввод-вывод п/л	СМ6204	PTII	ПР	4	070/074	777556-550
Дисплей (клавиатура/экран):						
основной	СМ7204	VT50	ПР	4	060/064	777566-560
дополнительные:						
I	СМ7204	VT50	ПР	4	300/304	776506-500
2	СМ7204	VT50	ПР	4	3I0/3I4	776516-510
3	СМ7204	VT50	ПР	4	320/324	776526-520

5. ВЕДУЩИЙ ПРОЦЕССОР

5.I. Общие сведения

Ведущий процессор выполняет логические и арифметические операции над данными. Вычисления ведутся параллельно, в дополнительном коде. Арифметико-логические команды могут иметь в качестве операндов как слова, так и байты. Слова - это пара смежных байт, причем первым или младшим в этой паре является байт с четным адресом. Этот же адрес является также адресом слова. Заметим, что в спецпроцессоре старшинство байт внутри слова обратное.

В процессоре имеются 8 общих или универсальных регистров (777700-777707), которые могут быть использованы как накопители, индексные регистры, указатели адресов, таблицы списков и т.п., а также в качестве указателей области памяти для временного хранения

данных (стека). Конкретное использование этих регистров зависит от выбранного режима адресации, а регистры R7 и R6 имеют особое назначение: R7 является счетчиком адресов или команд PC, а R6 - указателем стека SP.

R7 содержит адрес следующей исполняемой команды и, хотя он тоже является общим регистром, обычно его используют только для формирования адресов.

Система адресации предусматривает возможность обращения к 64 Кбайтам, что определяется наличием шестнадцатиразрядного регистра адреса. Обращение процессора к оперативной памяти осуществляется через общую шину, аналогично обращению к регистрам внешних устройств.

5.2. Стек

Стеком называется такой способ организации массива элементов памяти, при котором запись или извлечение элементов производится по принципу: последнее записанное - первое считанное. Это означает, что из стека всегда выбирается тот элемент, который был последним в него записан. Адрес, по которому производится выборка или запись элемента, называется вершиной стека. Архитектура комплекса позволяет легко организовать стеки со скользящей вершиной. В качестве указателя вершины стека можно использовать любой из регистров (кроме R7 - счетчика команд), а режим автоувеличения и автоуменьшения позволяют автоматически регулировать положение вершины стека. Некоторые команды ведущего процессора используют R6 в качестве указателя стека, в который они заносят данные для временного хранения. Поэтому R6 называется указателем "аппаратного" стека. Область памяти, в которой должен располагаться аппаратный стек, выбирает программист. Поэтому, если в программе имеются команды, использующие аппаратный стек, программист должен установить в SP первоначальное

значение адреса вершины стека. При записи в аппаратный стек (при исполнении соответствующих команд) процессор уменьшает содержимое SP на 2, после чего записывает новый элемент по адресу, содержащемуся в SP. При выборке из стека, процессор выбирает элемент по адресу, содержащемуся в SP, затем увеличивает содержимое SP на 2. Программист также может заносить свои элементы в аппаратный стек. При этом он должен соблюдать правило: перед записью в стек содержимое SP должно уменьшаться на 2, после выборки из стека содержимое SP должно увеличиваться на 2.

В аппаратном стеке могут храниться только слова. Использование SP для указания нечетных адресов или несуществующей памяти вызывает останов (двойная ошибка шины).

С помощью остальных регистров программист может организовать стеки, состоящие как из слов, так и из байт.

При выполнении некоторых команд, требующих временного запоминания данных (например, адреса возврата при обращении к подпрограмме), процессор использует R6 ,как указатель стека . Этим же стеком может пользоваться также и программист, который и определяет область памяти для стека, то есть указывает в R6 адрес вершины стека.

Граница стека - 400 (восьмеричное). После режимов адресации 4 и 5 , использующих R6 , и после команды JSR и прерываний контролируется переполнение стека. Переполнение обслуживается прерыванием по переполнению через вектор 4.

5.3. Слово состояния процессора

Регистр слова состояния процессора PS содержит информацию о предыдущем и текущем режиме, о текущем приоритете процессора и условный код результата выполненной команды. Назначение и использование разрядов PS показано на рис. 2. Это 16-битовый регистр,

который имеет адрес 777776.

Слово состояния процессора

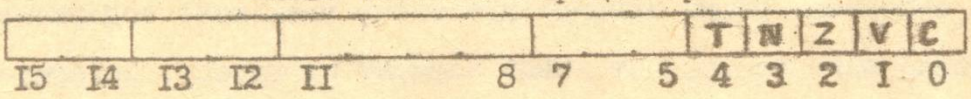


Рис. 2.

Здесь:

0/3 - коды условий;

4 - разряд слежения;

5/7 - приоритет процессора;

8/II - не используются;

I2/I3 - предыдущий режим процессора;

I4/I5 - текущий режим процессора

Коды условий содержат информацию о результате последней операции в процессоре.

Коды условий устанавливаются следующим образом:

Z = I, если результат был равен нулю;

N = I, если результат был отрицательным;

C = I, если в результате был перенос из старшего разряда;

V = I, если в результате было арифметическое переполнение.

Разряд слежения может быть установлен или очищен программистом, используя команды RTI, RTT или после прерывания. Если перед исполнением какой-либо команды этот разряд установлен, то после ее исполнения происходит внутреннее прерывание.

Процессору может быть присвоен программистом один из восьми уровней приоритета. Присвоение осуществляется установкой соответствующего кода (0/7) в разряды 5/7 слова состояния процессора. В случае установки кода 7 ни одно из внешних устройств не сможет прервать текущую программу.

Ведущий процессор может работать в одном из двух режимов: пользовательском и системном. В режиме пользователя программе запрещается использование команды HALT, кроме того будет игнорироваться команда RESET. Эти два режима позволяют полностью защищать выполнение программ в мультипрограммном режиме, обеспечивая двумя различными наборами процессорных стеков и регистров управления памятью (регистры диспетчера) для ее распределения. Разряды I4/I5 содержат информацию о текущем режиме процессора, код II - пользовательский режим, код 00 - системный режим. Разряды I2/I3 указывают предыдущий режим, т.е. в каком режиме находился процессор перед последним прерыванием (код II - пользовательский, код 00 - системный).

5.4. Внутренние прерывания

Система прерываний СМ1600 в общем была описана выше. Здесь более подробно рассматриваются только внутренние прерывания ведущего процессора.

При возникновении прерывания по внутренним причинам управление передается по адресу, записанному в векторе прерывания. Этот адрес указывается либо системной программой, либо пользователем. Существуют следующие типы внутренних прерываний ведущего процессора:

I) прерывание при нечетной адресации.

Это прерывание возникает при попытке исполнить команду, в которой указан нечетный адрес слова. Вектор прерывания должен быть в ячейках 4 и 6;

2) прерывание при пропадании питания.

При снижении сети переменного питания ниже нормального уровня происходит прерывание через вектор 24-26.

При этом отводится 2 мс на исполнение программы, адрес которой

записан в ячейке 24. При восстановлении питания на нормальный уровень снова происходит передача управления через вектор 24;

3) прерывание при адресации несуществующей памяти.

При попытке адресовать ячейку памяти, адрес которой больше чем верхняя граница адресации блока ОЗУ, происходит прерывание через вектор 4;

4) прерывание по неправильной команде.

При попытке исполнить команду с кодом операции, не входящим в систему команд, происходит прерывание через вектор I0;

5) прерывание по разряду слежения.

Если перед исполнителем какой-либо команды в слове состояния процессора установлен разряд слежения, то после ее исполнения происходит прерывание через вектор I4.

5.5. Диспетчер памяти

Диспетчер памяти предназначен для:

I) расширения объема оперативной памяти от 28 Кслов до I24 Кслов;

2) обеспечение виртуальной адресации и защиты памяти;

3) разделение областей памяти между программами пользователя и программами операционной системы;

4) обеспечение управляющей информации операционных систем, работающих в мультипрограммном режиме.

I6-разрядная длина слова ограничивает зону возможных адресов оперативной памяти до 32 Кслов. Диспетчер памяти позволяет увеличить объем адресуемой оперативной памяти до I28 Кслов путем преобразования I6-разрядного виртуального адреса, генерируемого ведущим процессором в I8-разрядный физический адрес.

Вся область виртуальных адресов разделена на 8 страниц, каждой из которых соответствует регистр диспетчера APR (регистр активной страницы). Страница может содержать от I до I28 блоков

по 32 слова в каждой. Таким образом, максимальная длина страницы - 4096 слов, а минимальная - 32 слова. Максимальная же длина программы, используя все 8 страниц - 32768 (4096 х 8) слов.

Диспетчер памяти имеет два набора из восьми регистров активной страницы (APR). Каждый APR состоит из регистра адреса страницы (PAR) и регистра дискриптора страницы (PDR). Эти регистры всегда используются в паре и содержат всю информацию, необходимую для размещения и описания текущих активных страниц для каждого режима работы. Один набор PAR/PDR используется в режиме системы, а другой используется в режиме пользователя. Биты текущего режима (или в некоторых случаях биты предыдущего режима) слова состояния процессора определяют, который набор будет выбран при каждом обращении к памяти. Программа, работающая в одном режиме,не может использовать наборов PAR/PDR другого режима для обращения к памяти. Таким образом, два набора являются главным средством, обеспечивающим полностью защищенную среду для мультипрограммных систем с разделением времени.

К каждому PAR и PDR каждого набора присвоен специальный адрес. Таблица 2 является полным списком адресных присваиваний.

Таблица 2

Регистры активных страниц системы			Регистры активных страниц пользователя		
№	PAR	PDR	№	PAR	PDR
0	772340	772300	0	777640	777600
I	772342	772302	I	777642	777602
2	772344	772304	2	777644	777604
3	772346	772306	3	777646	777606
4	772350	772310	4	777650	777610

Продолжение табл. 2

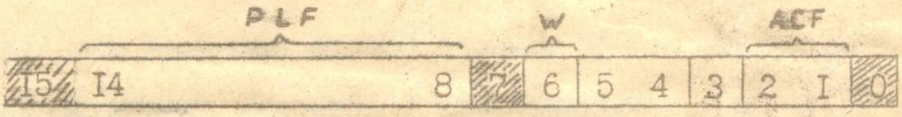
Регистры активных страниц системы			Регистры активных страниц пользователя		
№	PAR	PDR	№	PAR	PDR
5	772352	772312	5	777652	777612
6	772354	772314	6	777654	777614
7	772356	772316	7	777656	777616

Примечание. Внешние устройства не имеют доступа к этим регистрам.

Регистр адреса страниц (PAR) содержит 12-битовое поле адреса страницы и загружается под управлением программы. Биты 0/11 определяют базовый адрес страницы, а биты 15/12 - не используются.

Регистр дескриптора страницы (PDR) содержит информацию относительно использования страницы.

Формат регистра:



Поле управления доступом (ACF) описывает право обращения к определенной странице. В табл. 3 перечислены возможные значения ACF, их описания и функции. Поле управления доступом загружается под управлением программы.

Таблица 3

ACF	Описание страницы	Описание функции
00	недоступное	Любая попытка обращения к недоступной странице вызывает прерывание

Продолжение табл.3

ACF	Описание страницы	Описание функции
		по вектору 250
0I	доступна только для чтения	Любая попытка записи вызывает прерывание по вектору 250
I0	не используется	Любая попытка обращения вызывает прерывание по вектору 250
II	доступна для записи и для чтения	Запись и чтение разрешены

Поле направления расширения страницы ED [разряд 3] указывает, в каком направлении может быть расширена страница. Если ED =0, то страница может быть расширена в сторону увеличения адресов, если ED = I - страница может быть расширена в сторону уменьшения адресов. Бит ED загружается в PDR под управлением программы.

Бит "запись в страницу" (W) занимает 6 разряд и указывает, была ли произведена запись в страницу после загрузки ее в память. Бит " W " устанавливается автоматически логикой управления диспетчера памяти, а очищается загрузкой регистров PAR или PDR соответствующей страницы.

Поле длины страницы (PLF), расположенное в I4/8 разрядах, определяет разрешенную длину страницы в блоках из 32 слов. Максимальная длина I28 блоков. Поле P L F загружается под управлением программы.

При расширении страницы вверх (ED =0) число, устанавливаемое

в поле PLF, должно быть на I меньше числа требуемых блоков. Это вызвано тем, что нумерация начинается с 0. При расширении страницы вниз число блоков берется в дополнительном коде.

Основная информация, необходимая для построения физического адреса, приходит из виртуального адреса и регистра активных страниц (APF). Разряды I5/I3 виртуального адреса указывают, который из восьми регистров активной страницы будет использован для формирования физического адреса (т.е. номер набора PAR/PDR). Разряды I2/6 указывают номер блока (от 0 до I27) внутри страницы, разряды 5/0 интерпретируются как номер слова в блоке.

Формирование физического адреса показано на рис. 3.

Схема формирования физического адреса

Рис. 3.

Логическая последовательность построения физического адреса следующая:

I. Выбрать набор регистров активных страниц в зависимости от режима.

2. Поле активной страницы виртуального адреса используется для выборки регистра активной страницы (APR0 - APR7).

3. Поле адреса страницы выбранного регистра активной страницы содержит

начальный адрес текущей активной страницы в качестве номера блока в физической памяти.

4. Номер блока из виртуального адреса прибавляется к номеру блока из поля адреса страницы, чтобы получить номер блока в физической памяти, который будет содержать физический адрес.

5. Смещение в блоке из поля смещения виртуального адреса присоединяется к номеру физического номера блока для получения истинного I8-битового физического адреса.

Таблица 4 перечисляет пределы виртуальных адресов, которые указывают каждый из наборов PAR/PDR .

Таблица 4

Границы виртуального адреса	Набор PAR/PDR
000000 - I7776	0
020000 - 37776	I
040000 - 57776	2
060000 - 77776	3
I00000 - II7776	4
I20000 - I37776	5
I40000 - I57776	6
I60000 - I77776	7

Работой и состоянием диспетчера памяти управляют два регистра: SR0 и SR2. Прерывания, генерируемые с помощью аппаратурных средств защиты, векторизируются через виртуальную ячейку ядра 250. Регистры состояния SR0 и SR2 используются для определения причин прекращения. Таким образом, программа ядра должна убедиться, что виртуальный адрес ядра 250 правильно отображен; иначе произойдет зацикливание, которое требует вмешательства с пульта.

Регистр состояния SR0 (слово по адресу 777572) содержит флажки ошибок прекращения, возбуждения диспетчера памяти и другую существенную информацию, необходимую для операционной системы.

Биты I5/I3 могут быть установлены как аппаратно, так и программным способом, и указывают на ошибку доступа в память. Аппаратно эти биты устанавливаются только тогда, если бит 0 либо 8 равен I.

Бит I5 указывает на попытку обратиться на недоступную страницу (ACF равно 00 или I0, или PS [I5,I4] равно 0I или I0).

Бит I4 устанавливается при попытке обращения к адресу, выходящему за границы разрешенной области, указанной в PLF регистра PDP соответствующей страницы.

Бит I3 является битом прерывания из-за нарушения режима только для чтения, т.е. устанавливается при попытке записи в страницу, доступную для чтения.

Бит 8 в SR0 устанавливается под управлением программы и указывает на использование диспетчера памяти в диагностическом режиме, при котором происходит перемещение адресов только при последнем обращении за приемником.

Биты 6, 5 указывают режим процессора (00 или II), при котором возникла ошибка.

Биты 3/I содержат номер страницы и могут быть использованы для идентификации страницы.

Бит 0 является битом разрешения работы диспетчера памяти. Этот бит устанавливается под управлением программы. При установке этого бита в "I" все адреса перемещаются и защищаются всеми видами защиты, заложенными в диспетчере памяти. При установке бита 0 в "0" адреса не перемещаются и не защищаются.

Регистр состояния SR2 (слова с адресом 777576) содержит I6-разрядный виртуальный адрес, загружаемый в начале выборки каждой

инструкции, но не меняется, если происходит ошибка выборки команды. SR2 только читаемый регистр. Попытка записи в этот регистр не изменит его содержания.

5.6. Система команд

5.6.I. Общие сведения

Команды ведущего процессора могут иметь различное число операндов. Таким образом, можно выделить нуль адресные (например, останов), одноадресные (например, очистка) и двухадресные (например, сложение) команды.

Все команды можно делить на 6 групп:

I. Одноадресные команды.

2. Двухадресные команды (арифметические и логические команды).

3. Команды программного управления (переходы, команды подпрограмм, прерывания).

4. Команды оперативной группы (операции управления процессором).

5. Операторы условного кода (команды изменения битов слова состояния процессора).

6. Команды плавающей запятой.

Список обозначений, применяемых при описании команд, приведен в приложении I.

Команды содержат следующую информацию:

код операции;

номер выбранного универсального регистра для обнаружения операнда источника и/или операнда приемника;

режим адресации, определяющий способ использования регистра.

Универсальные регистры могут использоваться в качестве:

накопителей - в этом случае данные изменяются внутри регистра;

указателей - при этом содержимое регистра является адресом операнда;

указателей, содержимое которых автоматически изменяется после исполнения команды;

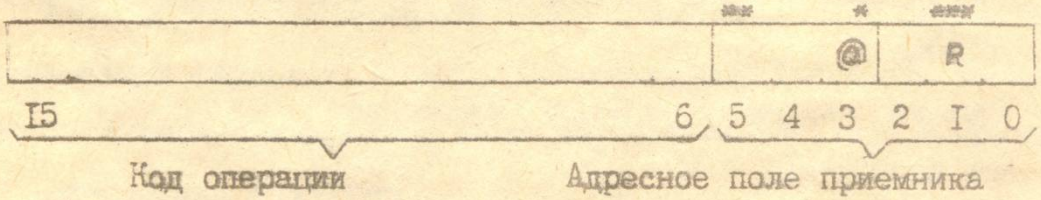
индексных регистров , когда содержимое регистра складывается с содержимым памяти, следующей непосредственно за командой, а полученная сумма является адресом операнда.

Любой регистр может быть использован программой в качестве указателя стека; однако некоторые команды, связанные с выходом на подпрограмму и возвратом и обслуживанием прерывания, автоматически используют R6, как аппаратурный стековый указатель, называемый SP. R7 используется процессором в качестве счетчика команд PC.

5.6.2. Режим адресации

Всего для ведущего процессора имеется I2 режимов адресации, позволяющих различное использование содержимого регистров. Режимы адресации применяются в командах, производящих арифметическо-логическую обработку данных (например: сложение, пересылка, логическое "или" и т.п.). Эти команды можно разделить на одноадресные и двухадресные команды.

Формат первого слова всех одноадресных команд (таких, как очистка, инкрементирование, проверка) следующий:



* Определяет прямую или косвенную адресацию;

** Определяет режим использования выбранного регистра;

*** Определяет один из восьми универсальных регистров.

В разрядах с I5 по 6 находится код операции, который определяет тип исполняемой команды.

Как для прямой, так и для косвенной адресации содержимое разрядов режима определяет число слов памяти, занимаемое командой (т.е. одно или два). При использовании режимов индексации команда всегда занимает два слова.

Операции, которые требуют двух операндов (такие, как сложение, вычитание, пересылка и сравнение), выполняются двухадресными командами. Первый операнд называется операндом источника (или просто источником), а второй - операндом приемника (приемником). Разряды в полях адреса источника и приемника могут определять различные режимы адресации и различные регистры. Формат первого слова двухадресной команды:

* разряд прямой/косвенной адресации

** режим использования выбранного регистра

*** выбранный регистр.

Адресное поле источника используется для определения первого операнда. Приемник содержит адрес второго операнда и результата. Например, команда **ADD** A,B складывает содержимое ячейки A (источник) с содержимым ячейки B (приемник). После исполнения команды результат содержится в B, а содержимое A не изменяется.

В табл. 5 перечислены режимы адресации и функции этих режимов.

Таблица 5

Режимы адресации

Режим	Адресное поле	Наименование	Синтаксис	Описание
		Прямая адресация		
0	0R	Регистровая	R_n	Регистр содержит операнд
2	2R	Автоинкрементная	$(R_n)+$	Регистр содержит адрес операнда. После обращения содержимое регистра автоматически увеличивается
4	4R	Автодекрементная	$-(R_n)$	Содержимое регистра автоматически уменьшается и тогда регистр содержит адрес операнда
6	6R	Индексная	$X(R_n)$	Значение X (запоминается в слове, следующим за командой) складывается с (R_n) для формирования адреса операнда. Ни X, ни (R_n) не изменяются
		Косвенная адресация		
I	IR	Косвенно--регилтровая	$@R_n$ или $@(R_n)$	Регистр содержит адрес операнда
3	3R	Косвенно-автоинкрементная	$@(R_n)+$	Регистр вначале используется как указатель слова, сдержащего адрес операн-

Продолжение табл. 5

Режим	Адресное поле	Наименование	Синтаксис	Описание
				да, затем автоматически увеличивается (всегда на 2, даже для байтовых команд).
5	5R	Косвенно-автодекрементная	@-(Rn)	Регистр автоматически уменьшается (всегда на 2, даже для байтовых команд) и тогда используется как указатель слова, содержащего адрес операнда.
7	7R	Косвенно-индексная	@X(Rn)	Значение X (запоминается в памяти в слове, следующем за командой) и (Rn) складываются и сумма используется как указатель слова, содержащего адрес операнда. Ни X, ни (Rn) не изменяются.
		Адресация PC		
2	27	Непосредственная	#n	Операнд следует за командой
3	37	Абсолютная	@#A	Абсолютный адрес следует за командой.
6	67	Относительная	A	Адрес A относительно команды следует за данной командой.
7	77	Косвенно-относительная	@A	Адрес ячейки, содержащей адрес A относительно команды, следует за командой.

Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата

16-2918 Изм. 4.6.

Адресация PC фактически ничем не отличается от других режимов; специфика здесь только в том, что в качестве универсального регистра используется счетчик команд PC (регистр 7).

Хотя PC может быть использован в любом из стандартных режимов адресации, на практике имеет смысл работа с PC только в четырех из этих режимов. При работе с PC эти режимы принято называть непосредственным, абсолютным, относительным и косвенно-относительным.

Заметим, что перед тем, как команда будет выбрана и исполнена процессором, PC указывает на первое слово команды. Процессор выбирает первое слово, одновременно увеличивая PC на 2. Режим 27 является режимом операнда-источника (автоувеличение регистра 7, т.е. PC), здесь PC используется как указатель операнда (т.е. второго слова команды), а затем увеличивается на 2 и указывает на следующую команду.

Режим 37 эквивалентен косвенному режиму с автоувеличением. Содержимое второго слова команды интерпретируется как адрес операнда.

Режим 67 является режимом индексации с использованием счетчика команд. Значение, находящееся во втором или третьем слове команды, являются не адресом операнда, а числом, которое добавляется к содержимому PC; полученная сумма интерпретируется как адрес операнда.

Режим 77 сходен с относительным режимом, однако, сумма PC и второго слова команды является здесь не адресом операнда, а адресом адреса операнда.

Несколько примеров адресации PC приведено в табл. 6.

Имеет смысл отдельно рассмотреть особенности абсолютной и относительной адресации. Дело в том, что относительный режим наиболее часто используется в программах Ассемблера. Это происходит потому,

Таблица 6

Мнемоника	Код команды	Описание
ADD# I0,R3	062703 0000I0	I0 складывается с ссдержимым R3

До		После	
00I020 062703	PC=00I020	00I020 062703	PC=00I024
00I022 0000I0	R3=000020	00I022 0000I0	R3=000030
00I024		00I024	

CLR @#II00	005037 00II00	Засылка нулей в ячейку II00

До		После	
000020 005037	PC=000020	000020 005037	PC=000024
000022 00II00		000022 00II00	
00II00 I77777		00II00 000000	

INC A	005267 000054	Содержимое ячейки A увеличивается на I

До		После	
00I020 005267	PC=00I020	00I020 005267	
00I022 000054	I024	00I022 000054	
00I024	+ 54	00I0<4	
	II00		PC=00I024
00II00 000000		00II00 00000I	

CLR @A	005077 000020	Засылка нулей в ячейку, адрес которой находится в A; адрес A получается путем сложения PC со вторым словом команды

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл. 16-2978

Изм.	Стр.	№ докум.	Подп.	Дата	I.700.0I0T0I	Стр 39

Ф26 ГОСТ 2.104-68 Копировал Формат А4

Продолжение табл. 6

Мнемоника	Код команды	Описание
	До	После
001020 005077	PC=001020	001020 005077 PC=001024
001022 000020		001022 000020
001024	1024 + 20 1044	001024
001044 010100		001044 010100
010100 177777		010100 000000

что любая одно - и двухадресная команда должна использовать универсальный регистр.

В результате, когда программист пишет

CLR 100

MOV X,Y

значение, которое запоминается во втором или третьем слове команды не является адресом операнда; это число складывается с содержимым PC и тогда уже становится адресом операнда. Таким образом, это число равно X + PC,

где X - действительный адрес.

Если оператор Ассемблера MOV 100, R3 транслируется в ячейке 20, то команда располагается в памяти следующим образом:

ячейка 20; 016703

ячейка 22; 000054.

Процессор в ходе исполнения команды MOV добавляет 2 к со-

держимому PC, который указывает теперь на ячейку 22. Режим источника – 67, т.е. индексация с использованием PC. Процессор выбирает слово, на которое указывает PC и увеличивает PC на 2. Теперь PC указывает на ячейку 24. При начислении адреса операнда-источника содержимое второго слова команды складывается с регистром, указанным в поле режима. Таким образом, адрес операнда равен содержимому второго слова команды + PC = 54 + 24 = I00.

Этот режим называется относительным потому, что адрес вычисляется относительно текущего содержимого PC. Содержимое второго слова команды является расстоянием (в байтах) между адресом операнда и текущим адресом в PC, если положение команды и операнда изменится, то расстояние между ними не изменится и команда по-прежнему будет работать правильно.

5.6.3. Работа с байтами

Команды могут оперировать как со словами, так и непосредственно с байтами. Байтовые инструкции с прямой адресацией выбирают младший байт указанного регистра.

Байтовые инструкции имеют тот же формат, что и соответствующие со словами, тот же алгоритм, то же описание и также устанавливают код условий по байту результата.

Коды байтовых операций отличаются от соответствующих кодов операций для слов единицей в старшем разряде, а мнемонические их обозначения – дополнительной буквой B.

Время их выполнения – то же при работе с четными байтами и несколько увеличенное при выборке байтов по нечетным адресам (старших байтов).

Расположение слов и байтов в памяти:

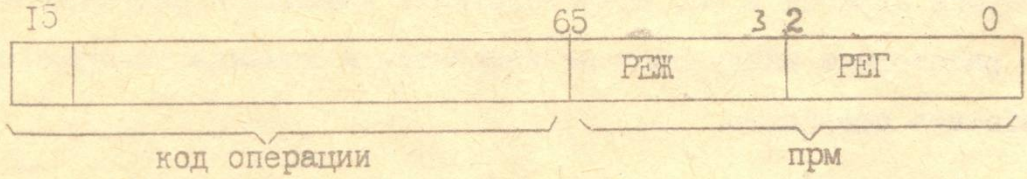
байт I	байт 0	слово 0
байт 3	байт 2	слово 2
байт n+I	байт n	слово n

Рис. 4

При приеме из памяти байт поступает в младшие разряды регистра процессора, а его знак расширяется в старших (7 - I5) разрядах.

5.7. Группа одноадресных команд

Одноадресные команды имеют следующий формат:



Список одноадресных команд:

I. CLR, CLRB	8. TST, TSTB
2. INC, INCB	9. ASR, ASRB
3. DEC, DECB	I0. ASL, ASLB
4. COM, COMB	II. ROR, RORB
5. NEG, NEGB	I2. ROL, ROLB
6. ADC, ADCB	I3. SWAB
7. SBC, SBCB	I4. SXT

Очистка - CLR, CLRB

По этой команде содержимое указанного приемника заменяется нулями.

Код операции: 0050, I050

Алгоритм: 0 → (прм)

Код условий: Z - устанавливается;

N - очищается;

C - очищается;

V - очищается.

Инкремент - INC, INCB

По этой команде к содержимому приемника прибавляется I.

Код операции: 0052, I052.

Алгоритм: (прм) + I → (прм)

Код условий: Z - устанавливается, если результат равен 0, иначе - очищается;

N - устанавливается, если результат меньше 0, иначе - очищается;

C - не изменяется;

V - устанавливается, если (прм) был 077777, иначе - очищается.

Декремент - DEC, DECB

Содержимое приемника уменьшается на I.

Код операции: 0053, I053.

Алгоритм: (прм) - I → (прм).

Код условий Z - устанавливается, если результат равен 0, иначе очищается;

N - устанавливается, если результат меньше 0, иначе очищается;

C - не изменяется;

V - устанавливается, если (прм) был I00000, иначе очищается.

Инверсия - COM, COMB

Содержимое приемника заменяется его логической инверсией (включается каждый бит, который равен нулю, и гасится каждый бит, равный единице).

Код операции: 005I, I05I.

Алгоритм: $(\overline{\text{прм}})$ → (прм).

Код условий: Z - устанавливается, если результат равен 0, иначе - очищается;

N - устанавливается, если старший разряд результата равен I, иначе - очищается;

C - устанавливается;

V - сбрасывается.

Отрицание - NEG, NEGB

Содержимое приемника заменяется его дополнением до 2. Число I00000 заменяется самим собой.

Код операции: 0054, I054.

Алгоритм : - (прм) → (прм)

Код условий: Z - устанавливается, если результат равен 0, иначе - очищается;

N - устанавливается, если результат меньше 0, иначе - очищается;

C - сбрасывается, если результат равен 0, иначе - устанавливается;

V - устанавливается, если результат равен I00000, иначе - очищается.

Сложение с переносом - ADC, ADCB

К содержимому приемника добавляется содержимое бита C. Это разрешает перенос, полученный при сложении младших слов/байтов, добавить к результату сложения старших разрядов.

Код операции: 0055, I055.

Алгоритм: (прм) + (C) → (прм)

Код условий: Z - устанавливается, если результат равен 0, иначе - очищается;

N - устанавливается, если результат меньше 0, иначе - очищается;

C - устанавливается, если (прм) = I77777, а C было равно I, иначе - очищается;

V - устанавливается, если (прм) = 077777, а C было равно I, иначе - очищается.

Вычитание переноса - SBC, SBCB

Из содержимого приемника вычитается содержимое бита С. Это разрешает перенос, полученный при вычитании младших слов /байтов, вычесть из старших разрядов результата.

Код операции: 0056, I056.

Алгоритм: (прм) - (С) → (прм).

Код условий: Z - устанавливается, если результат равен 0;

N - устанавливается, если результат меньше 0, иначе - очищается;

V - устанавливается, если (прм) был I00000, иначе - очищается;

C - сбрасывается, если (прм) был равен 0 и С=I, иначе - устанавливается.

Проверка - TST, TSTB

Согласно содержимому приемника включаются биты условного кода N и Z .

Код операции: 0057, I057.

Алгоритм: (прм) → (прм)

Код условий: Z - устанавливается, если результат равен 0, иначе-очищается;

N - устанавливается, если результат меньше 0, иначе - очищается;

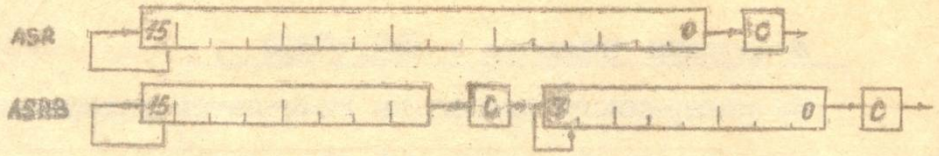
C - сбрасывается;

V - сбрасывается.

Арифметический сдвиг вправо - ASR, ASRB

Все биты приемника сдвигаются вправо на один разряд. Старший бит повторяется. Бит С загружается младшим битом приемника. Операция реализует деление числа со знаком на 2.

Код операции: 0062, I062.

Алгоритм: 

Код условий: Z – устанавливается, если результат равен 0, в остальных случаях гасится;

N – устанавливается, если старший разряд результата содержит I, в остальных случаях – сбрасывается;

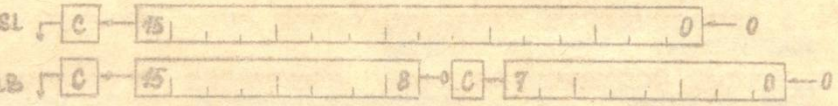
C – загружается содержимым младшего разряда приемника;

V – загружается результат исключающего ИЛИ битов N и C (после окончания операции сдвига).

Арифметический сдвиг влево – ASL , ASL B

Все биты приемника сдвигаются влево на I разряд, в нулевой разряд заносится 0. Старший бит приемника загружается в бит C. Операция выполняет умножение числа со знаком на 2.

Код операции: 0063, I063.

Алгоритм: 

Код условий: Z – устанавливается в I, если результат равен 0, иначе – сбрасывается;

N – устанавливается в I, если старший разряд результата содержит I, иначе гасится;

C – загружается содержимым старшего разряда приемника;

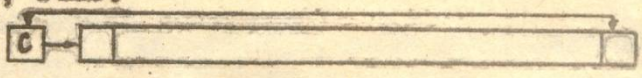
V – загружается результат исключающего ИЛИ бита и C (после окончания операции сдвига).

Циклический сдвиг вправо – ROR, RORB

Все биты приемника сдвигаются вправо на I разряд, нулевой бит загружается в C, а предыдущее значение бита C – в старший бит при-

приемника.

Код операции: 0060, I060.

Алгоритм: 

Коды условий: Z – устанавливается в I, если результат равен 0, иначе сбрасывается;

N – устанавливается в I, если старший бит результата равен I (результат меньше 0), в остальных случаях гасится;

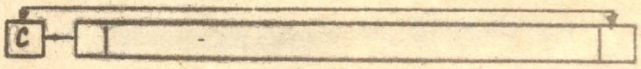
C – загружается младшим разрядом содержимого приемника;

V – заносится результат исключающего ИЛИ бита N и C (по окончании операции сдвига).

Циклический сдвиг влево – ROL , ROLB

Все биты приемника сдвигаются на один разряд влево. Старший бит заносится в бит C, а предыдущее значение бита C загружается в младший бит приемника.

Код операции: 006I, I06I.

Алгоритм: 

Код условий: Z – устанавливается в I, если все разряды результата равны 0, иначе – сбрасывается;

N – устанавливается в I, если старший разряд результата равен I, иначе – сбрасывается;

C – загружается содержимым старшего разряда приемника;

V – загружается результат исключающего ИЛИ бита и C (после окончания команды сдвига).

Перестановка байтов – SWAB

Старший байт и младший байт слова приемника меняются местами.

Код операции: 0003.

Код условий: Z – устанавливается в I, если младший байт ре-

зультата равен 0; иначе - сбрасывается.

N - устанавливается в I; если старший разряд младшего байта результата (бит 7) содержит I, иначе - сбрасывается;

C - сбрасывается;

V - сбрасывается.

Распространение знака - SXT

Если бит N равен I, на место операнда - приемника помещается - I, если бит N погашен, на место операнда - приемника помещается 0. Данная команда особенно полезна в арифметике повышенной точности, так как она разрешает растяжение знака через кратные слова.

Код операции: 0067

Алгоритм: 0 ⟶ (прм), если N погашен;

-I ⟶ (прм), если N включен.

Код условий Z - включается, если N равно 0;

N - не изменяется;

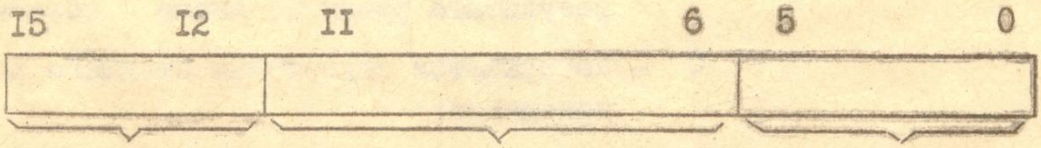
C - не изменяется;

V - гасится.

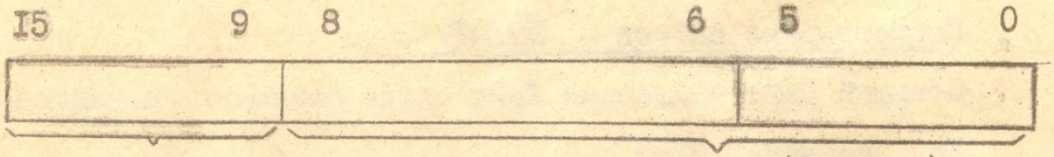
5.8. Группа двухадресных команд

Форматы:

1



2



Список двухадресных команд:

I. MOV, MOVB	7. SUB
2. CMP, CMPB	8. MUL
3. BIT, BITB	9. DIV
4. BIC, BICB	I0. ASH
5. BIS, BISB	II. ASHC
6. ADD	I2. XOR

Примечание. Команды I-7 имеют формат I, 8-I2 - формат 2.

Пересылка - MOV, MOVB

Операнд источника передается по адресу приемника, предыдущее содержимое приемника теряется.

Код операции: OI, II.

Алгоритм: (ист) → (прм).

Код условий: Z - устанавливается в I, если содержимое источника равно 0, иначе - сбрасывается;

N - устанавливается в I, если содержимое источника меньше 0, иначе - сбрасывается;

C - не изменяется;

V - сбрасывается.

Сравнение - CMP, CMPB

Сравниваются операнды источника и приемника, и включаются условные коды. Оба операнда не изменяются. За командой сравнения обычно следует команда условного перехода. Заметим, что в отличии от команды вычитания операция выполняется как (ист) - (прм), а не как (прим) - (ист).

Код операции: 02, I2.

Алгоритм: (ист) - (прм).

Код условий: Z – устанавливается в I, если результат равен 0, иначе – сбрасывается;

N – устанавливается в I, если результат меньше 0, иначе – сбрасывается;

C – сбрасывается, если был перенос из старшего разряда результата, иначе – устанавливается в I;

V – устанавливается в I, если операнды имели разные знаки, а знаки результата и приемника совпадают; иначе – сбрасываются.

Проверка битов – BIT, BITB

Для сравнения операндов источника и приемника выполняется логическое И и соответственно изменяются условные коды. Операнды источника и приемника не изменяются. Команда может использоваться для проверки, являются ли погашенными биты источника, соответствующие включенным битам приемника.

Код операции: 03, I3.

Алгоритм: (ист) $\wedge$ (прм)

Код условий: Z – устанавливается в I, если результат равен 0, иначе – сбрасывается;

N – устанавливается в I, если старший разряд результата равен I, иначе – сбрасывается;

C – не изменяется;

V – сбрасывается.

Очистка битов – BIC, BICB

Разряды источника, содержащие I, сбрасывают соответствующие разряды приемника. Начальное содержимое приемника теряется. Содержимое источника не изменяется. Команда используется для очистки нужных разрядов слова или байта.

Код операции: 04, I4.

Алгоритм: $(\overline{\text{ист}}) \wedge (\text{прм}) \longrightarrow (\text{прм})$.

Код условий: Z - устанавливается в I, если результат равен 0, иначе - сбрасывается;

N - устанавливается в I, если старший разряд результата равен I, иначе - сбрасывается.

C - не изменяется;

V - сбрасывается.

Установка битов - BIS, BISB

Выполняется операция "ИЛИ" (логическое сложение) над обоими операндами: результат запоминается по адресу приемника. Начальное содержимое приемника теряется. Командой устанавливаются в I нужные разряды слова.

Код операции: 05, I5.

Алгоритм: (ист) $\vee$ (прм) $\longrightarrow$ (прм).

Код условий: Z - устанавливается в I, если результат равен 0, иначе - сбрасывается.

N - устанавливается в I, если старший бит результата равен I, иначе - сбрасывается;

C - не изменяется;

V - сбрасывается.

Сложение - ADD

Операнд - источник добавляется к операнду - приемнику и результат запоминается по адресу приемника. Исходное содержимое приемника теряется, содержимое источника не изменяется.

Код операции: 06.

Алгоритм: (ист) + (прм) $\longrightarrow$ (прм).

Код условий: Z - устанавливается в I, если результат операции равен 0, иначе - сбрасывается;

N - устанавливается в I, если результат меньше 0, иначе - сбрасывается;

C – устанавливается в I, если был перенос из старшего разряда результата, иначе – сбрасывается;

V – устанавливается в I, если в результате операции появляется арифметическое переполнение, т.е. знаки операндов совпадали, а результат имеет противоположный знак, иначе – сбрасывается.

Вычитание – SUB

Операнд – источник вычитается из операнда-приемника и результат запоминается по адресу приемника. Исходное значение приемника теряется, содержимое источника не изменяется.

Код операции: I6.

Алгоритм: (прм) – (ист) → (прм).

Код условий: Z – устанавливается в I, если результат равен 0, иначе – сбрасывается;

N – устанавливается в I, если результат меньше 0, иначе – сбрасывается;

C – сбрасывается, если был перенос из старшего разряда результата, иначе – устанавливается в I;

V – устанавливается в I наличием арифметического переполнения при выполнении операции (т.е. операнды имели разные знаки, а знаки результата и источника совпадают), иначе – сбрасываются.

Умножение – MUL

Содержимое регистра приемника и содержимое источника берется, как дополнение целых чисел до двух, умножаются и результат запоминается в регистре приемника и последующем регистре, если R четный. Если R нечетный, запоминаются только младшие разряды результата. Формат в Ассемблере: MUL ист,R

Код операции: 070.

Алгоритм: R·X(ист) → R , Rv1

Код условий: N – устанавливается, если произведение меньше нуля, в противном случае – очищается;

Z – устанавливается, если произведение равно 0, в противном случае очищается;

V – очищается;

C – устанавливается, если результат меньше, чем -2^{15} или больше, или равно $2^{15} - 1$.

Деление – DIV

32-разрядное делимое в регистрах R и Rv1 делится на 16-разрядный делитель (операнд источника). Частное остается в R . Остаток имеет тот же самый знак, как и делимое, и запоминается в Rv1 . R должен быть четным.

Код операции: 071.

Алгоритм: R , Rv1/(ист) → R , Rv1

Код условий: N – устанавливается, если частное меньше нуля, в противном случае очищается;

Z – устанавливается, если частное равно нулю, в противном случае очищается;

V – устанавливается, если источник равен нулю или абсолютное значение содержимого регистра больше абсолютного значения содержимого источника. (В этом случае команда отвергается, так как для представления частного требуется свыше 15 разрядов).

C – устанавливается, если делается попытка деления на ноль, в противном случае очищается.

Арифметический сдвиг - ASH

Содержимое регистра сдвигается влево или вправо на число разрядов, указанное в источнике. Счетчиком сдвигов служит 6 младших битов операнда источника. Диапазон сдвигов находится в пределах от -32 до +31. Если счетчик положительный, число сдвигается влево, если отрицательный - вправо.

Код операции: 072.

Алгоритм: R сдвигается арифметически на *n* разрядов вправо или влево, где *n* = (ист).

Код условий:

N - устанавливается, если результат меньше нуля, в противном случае очищается;

Z - устанавливается, если результат равен 0, в противном случае - очищается;

V - устанавливается, если знак регистра в процессе сдвига меняется, в противном случае очищается;

C - заносится последний бит, выдвинутый из регистра.

Комбинированный арифметический сдвиг - ASHC

Содержимое регистра R с четным номером и следующего за ним регистра R∨1 обрабатывается как 32-разрядное слово. Регистр R∨1 представляет разряды с 0 по 15, а регистр R - с 16 по 32. Объединенное таким образом число может сдвигаться на 31 разряд влево и на 32 разрядов вправо. Если в команде указывается регистр с нечетным номером, тогда обрабатывается содержимое только одного регистра. В этом случае при указании сдвига вправо осуществляется циклический (кольцевой) сдвиг максимум на 16 разрядов.

Код операции: 073.

Алгоритм: R , R∨1, двойное слово, сдвигается на *n* разрядов вправо или влево, где *n* занимает младших шесть

разрядов источника.

Код условий: N – устанавливается, если результат меньше 0, иначе очищается;

Z – устанавливается, если результат равен нулю, иначе очищается;

V – устанавливается, если знаковый разряд меняется в процессе сдвига, иначе очищается;

C – заносится старший бит сдвига вправо, т.е. последний бит, выдвинутый из 32-битового операнда.

Исключающее ИЛИ-XOR

Исключающее ИЛИ регистра и операнда приемника запоминается по адресу приемника. Содержимое регистра не изменяется. Формат в Ассемблере XOR R,прм.

Код операции: 074.

Алгоритм: R v (прм) → (прм).

Код условий: N – устанавливается, если результат меньше 0, иначе гасится;

Z – устанавливается, если результат равен 0, иначе гасится;

V – гасится;

C – не изменяется.

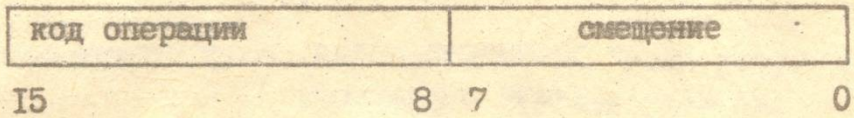
5.9. Команды программного управления

В этом разделе описываются команды перехода и команды JMP, JSR, RTS, MARK, SOB, EMT, TRAP, IOT, BPT.

Команды программного управления (переходы, команды подпрограмм, прерывания) не изменяют условные коды в слове состояния процессора.

Команды перехода (BR, BNE, BEQ, BPL , BMI, BGE, BLT, BGT, BLE, BHI, BLOS , BVC , BVS , BCC, BHIS , BCS , BLO) занимают

одно слово памяти и имеют формат:



Старший байт команды содержит код операции; младший байт содержит 8-разрядное смещение со знаком, которое определяет адрес перехода относительно счетчика команд. Этот адрес вычисляется процессором по следующим правилам:

1) знаковый разряд копируется в разряды 8-15;

2) результат умножается на 2. Полученное таким образом значение смещения показывает количество байт, а не слов;

3) результат складывается со счетчиком команд для получения адреса перехода.

Для того, чтобы сформировать байт смещения, Ассемблер производит обратное преобразование адреса перехода.

Необходимо помнить, что когда смещение складывается с PC, счетчик команд указывает на следующее слово за командой перехода.

8-разрядное смещение позволяет осуществлять переход на 200_8 слов относительно текущего значения PC назад и на 177_8 слов (376_8 байтов) вперед.

При переходе от 16-разрядного слова к 8-разрядному байту необходимо учитывать, что восьмеричное представление нарушает границы байтов в слове.

При выполнении описываемой группы команд процессором анализируется логическое выражение, в котором в качестве булевых переменных используются разряды кодов условий слова состояния процессора. Если это выражение истинно, то осуществляется переход, если - ложно, то управление передается следующей по порядку команде.

Переход - BR

Команда выполняет безусловный переход в области от - I28 до +I27 слов.

Код операции: 0004.

Алгоритм: PC + (2 x смещение) → PC.

Переход, если не равно - BNE

Проверяется состояние бита Z и выполняется переход, если бит Z погашен. BNE является дополнением операции BEQ. Она используется для проверки наличия неравенства, следуя за командой CMP, для проверки, что биты, включенные в приемнике, являются включенными и в источнике, следуя за командой BIT, и в основном для проверки неравенства нулю результата предыдущей команды.

Код операции: 00I0.

Алгоритм: PC + (2 x смещение) → PC, если Z = 0.

Переход, если равно - BEQ

Проверяется состояние бита Z и выполняется переход, если бит Z установлен. Например, команда используется для проверки равенства, следуя за командой CMP, для проверки, что биты , погашенные в приемнике, являются установленные в источнике, следуя за командой BIT и, вообще, для проверки нулевого результата предыдущей операции.

Код операции: 00I4.

Алгоритм: PC + (2 x смещения) → PC, если Z=I.

Переход , если плюс - BPL

Проверяется состояние бита N и выполняется переход, если N погашен. BPL является дополнением команды BMI.

Код операции: I000.

Алгоритм: PC + (2 x смещение) → PC, если N=0.

Переход, если минус - BMI

Проверяется состояние бита N и выполняется переход, если N установлен. Используется для проверки знака (самого старшего бита) результата предыдущей операции.

Код операции: I004.

Алгоритм: PC + (2 x смещение) ⟶ PC, если N = I

Переход, если больше или равно - BGE

Выполняется переход, если N и V или оба погашены, или оба установлены. BGE является дополнением операции BLT. Поэтому BGE всегда вызывает переход, если она следует за командой сложения двух положительных чисел, а также, если результат равен 0.

Код операции: 0020

Алгоритм: PC + (2 x смещение) ⟶ PC, если N⊻V =0.

Переход, если меньше - BLT

Выполняется переход, если исключающее ИЛИ битов V и N равно I. Поэтому команда всегда вызывает переход, если она следует за операцией сложения двух отрицательных чисел, даже при наличии переполнения, а также всегда вызывает переход за командой CMP, выполняемой над отрицательным источником и положительным приемником (даже при наличии переполнения). BLT не выполняет перехода, следуя за командой CMP, выполняемой над положительным источником и отрицательным приемником. Не выполняет перехода, если результат предыдущей операции был равен нулю (без переполнения).

Код операции: 0024.

Алгоритм: PC + (2 x смещение) ⟶ PC, если N⊻V = I.

Переход, если больше - BGT

Команда аналогична команде BGE , за исключением, что не вызывает перехода при нулевом результате.

Код операции: 0030.

Алгоритм: PC + (2 x смещение) ⟶ PC, если $Z \vee (N \neq V) = 0$.

Переход, если меньше и равно - BLE

Операция аналогична команде BLT, но дополнительно вызывает переход, если результат предыдущей операции был нулевым.

Код операции: 0034.

Алгоритм: PC + (2 x смещение) ⟶ PC, если $Z \vee (N \neq V) = 1$.

Беззнаковый переход, если больше - BHI

Выполняется переход, если предыдущая операция не выработала ни переноса, ни нулевого результата. Это имеет место при операции сравнения (CMP), если источник больше приемника по абсолютной величине.

Код операции: 1010.

Алгоритм: PC + (2 x смещение) ⟶ PC, если $Z = C = 0$

Беззнаковый переход, если меньше или равно - BLOS

Выполняется переход, если предыдущая операция выработала или перенос, или нулевой результат. Команда является дополнением команды BHI. Переход выполняется после команды, если источник является равным или меньше приемника по абсолютной величине.

Код операции: 1014.

Алгоритм: PC + (2 x смещение) ⟶ PC, если $C \vee Z = 1$.

Переход по отсутствию переполнения - BVC

Проверяется состояние бита V и выполняется переход, если этот бит погашен.

Код операции: 1020.

Алгоритм: PC + (2 x смещение) ⟶ PC, если $V = 0$.

Переход по переполнению - BVS

Проверяется состояние бита V , если бит включен, то выполняется переход . BVS используется для обнаружения арифметического переполнения предыдущей операции.

Код операции: I024.

Алгоритм: PC + (2 x смещение) ⟶ PC, если V =I.

Переход по отсутствию переноса - BCC

Беззнаковый переход, если больше или равно - BHIS

Проверяется состояние бита C и выполняется переход, если C погашен.

Код операции: I030.

Алгоритм: PC + (2 x смещение) ⟶ PC, если C=0.

Переход при наличии переноса - BCS

Беззнаковый переход, если меньше - BLO

Проверяется состояние бита C и выполняется переход, если C установлен.

Код операции: I034.

Алгоритм: PC + (2 x смещение) ⟶ PC, если C = I.

Абсолютный переход - JMP

Команда обеспечивает более гибкие переходники в программах, чем остальные команды переходов. Управление может передаваться в любое место памяти и сопровождается полной свободой подбора режимов адресации, за исключением нулевого регистрового режима. При выполнении абсолютного перехода с нулевым режимом адресации происходит внутреннее прерывание через вектор I0.

Формат команды JMP совпадает с форматом одноадресных команд (см.п.5.6.2).

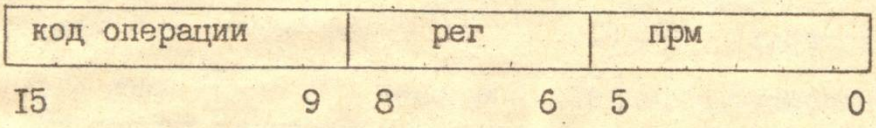
Код операции: 000I.

Алгоритм: (прм) → PC.

Переход к подпрограмме JSR

При выполнении команды JSR содержимое регистра, указанного в команде, запоминается в стеке, адрес следующей команды (содержимое PC) помещается в регистр, а адрес подпрограммы (содержимое приемника) - в PC, т.е. выполняется переход по адресу приемника. Поэтому подпрограммы, вложенные внутри подпрограммы на любой глубине, могут быть вызваны, используя тот самый регистр. Возврат из подпрограммы должен выполняться командой RTS .

Формат команды JSR :



Код операции: 004.

Алгоритм: (прм) → BP, где BP - внутренний регистр процессора.

(рег) → (SP)

PC → рег

BP → PC

Возврат из подпрограммы - RTS

Содержимое регистра, указанного в команде передается в PC, а затем в этот регистр заносится его первоначальное содержимое из стека. После этого указатель стека увеличивается на 2 (возвращается в то состояние, в котором он был до обращения к подпрограмме). Возврат из подпрограммы обычно выполняется через тот же самый регистр, который использовался при ее вызове в команде JSR .

Например:

```
JSR R5,@ # ПРОГ        ; ПРОГ - адрес подпрограммы
WORD AI, A2            ; AI и A2 параметры для подпрограммы I.
```

```
ПРГ1  MOV (R5)+,R1
      MOV (R5)+,R2
      .
      .
      .
      RTS  R5
```

Если подпрограмма не требует параметров, в качестве регистра можно использовать PC. Поэтому подпрограмма, вызванная JSR PC, ПРМ содержит возврат RTS PC.

Формат команды:

код операции	рег

I5 4 3 0

Код операции: 00020.

Алгоритм: (рег) → PC

(SP) → (рег).

Возврат с очисткой стека – MARK

Команда используется, как часть стандартной подпрограммы, и облегчает процедуру очистки стека при выходе из подпрограммы.

Формат в Ассемблере

Например:

```
MOV  R5,-(SP)      ; Содержимое регистра 5 послать в стек
MOV  П1,-(SP)      ; Параметры П1,П2,...ПN послать
                   ; в стек для последующего использо-
                   ; вания их подпрограммой.
MOV  ПN,-(SP)      ;
MOV  MARKN,-(SP)   ; Послать команду MARK  в стек
MOV  R,R5          ; Установить адрес команды
```

```
JSR PC,ПРОГ   ; Переход к подпрограмме
```

В данный момент стек выглядит так:

старое содержимое R5

П1

П2

.

.

.

ПN

MARK N

старое PC

И программа находится в адресе ПРОГ, от которого начинается подпрограмма.

```
ПРОГ:         ; Выполняется подпрограмма
RTS R5        ; Возврат.
```

Этим вызывается размещение содержимого регистра 5 в PC, который затем участвует в выполнении команды **MARK N**. Содержимое старого PC помещается в R5.

Команда **MARK N** выполняет:

указатель стека устанавливает для указания старого значения регистра 5;

значение регистра 5 (старое PC) помещает в PC;

содержимое старого R5 заносит в R5, тем заканчивается возврат из подпрограммы.

Код операции команды MARK: 0064

Формат:

код операции	N

15 6 5 0

Алгоритм: SP + 2 x N → SP, где N – число параметров

R5 → PC

(SP) → R5

Переход по счетчику - SOB

Содержимое указанного в команде регистра изменяется на I. Если результат не равен 0 , то из PC (PC указывает адрес следующей команды) вычитается удвоенное смещение. Смещение интерпретируется, как шестибитовое положительное число. Очевидно, что переход вперед невозможен. Если результат равен 0, то выполняется следующая команда. Данная команда обеспечивает эффективным методом управления циклами.

Код операции: 077.

Формат:

код операции	рег	смещение

I5 8 6 5 0

Алгоритм: Рег - I → рег, если результат не равен 0, то PC - (2 x смещение) → PC.

Команды перехода к подпрограмме со сменой состояния

Выполняются процедуры прерывания через вектора 30, 34, 20, I4.

Обозначение: EMT, TRAP, IOT, BPT

Код операции: I040, I044, 000004, 000003

Формат EMT и TRAP:

код операции	кп

I5 8 7 0

где КП - код пользователя.

Алгоритм:

EMT	TRAP
PS → (SP)	PS → (SP)
PC → (SP)	PC → (SP)
(30) → PC	(34) → PC
(32) → PS	(36) → PS

IOT	BPT
PS ⟶ (SP), PC ⟶ (SP)	PS ⟶ (SP), PC ⟶ (SP)
(20) ⟶ PC, (22) ⟶ PS	(I4) ⟶ PC, (I6) ⟶ PS

5.IO. Команды оперативной группы

В состав этой группы включены следующие команды:

I. RTI	7. MFPS
2. RTT	8. MTPS
3. MFPI	9. HALT
4. MFPD	IO. WAIT
5. MTPI	II. RESET
6. MTPD	

RTI

Команда используется для выхода из прерывания или из подпрограммы обслуживания прерывания. Значение счетчика команд (PC) и слова состояния процессора (PS) восстанавливается из стека. Если включается бит T в PS , то прерывание по биту T выполняется сразу после команды RTI.

Код операции: 000002

Алгоритм: (SP) ⟶ PC

(SP) ⟶ PS

RTT

Команда похожа на команду RTI, за исключением, что гарантируется выполнение первой команды следующей за RTT, т.е. если включен бит T, то первая команда за RTT выполняется до следующего прерывания по "T".

Код операции: 000006

Алгоритм: (SP) ⟶ PC

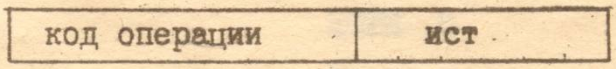
(SP) ⟶ PS

MFPI, MFPD

Слова из адреса в предыдущей области заносятся в текущий стек. Адрес операнда источника вычисляется в текущем режиме с использованием регистра адреса страниц, выбранного с учетом разрядов I4 и I5 в PS .

Код операции: 0065, I065.

Формат:



Алгоритм: (ИСТ) ⟶ (ВР), где ВР - внутренний регистр процессора .

(ВР) ⟶ (SP)

Условные коды:

N - устанавливается, если источник меньше 0, иначе очищается;

Z - устанавливается, если источник = 0, иначе очищается;

C - очищается;

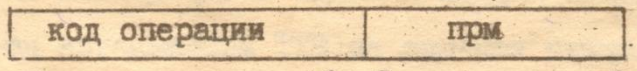
V - не изменяется.

MTPI, MTPD

По этим командам считывается слово из стека текущего режима, определенного I5 и I4 битами PS и запоминается по адресу предыдущей области PS (биты I3, I2). Адрес приемника подсчитывается в текущем режиме (в зависимости от битов I4 и I5 в PS).

Код операции: 0066, I066.

Формат:



Алгоритм: (SP) ⟶ (ВР), где ВР - внутренний регистр процессора.

(ВР) ⟶ (прм)

Условные коды: N - включается, если приемник меньше 0, иначе гасится;

Z - включается, если приемник равен 0, иначе гасится;

C - гасится;

V - не изменяется.

MFPS

8 битов содержимого PS (слова состояния процессора) пересылаются в приемник. Если имеется нулевой режим адресации, 7 бит из PS ,как знак, распространяется через младший байт регистра и операнд - приемник трактуется, как байтовый адрес.

Код операции: I067.

Алгоритм: PS ⟶ (прм), прм - младшие 8 битов.

Условные коды: N - устанавливается, если PS бит 7=I,иначе гасится;

Z - устанавливается, если PS[0/I7] =0,иначе гасится;

C - гасится;

V - не изменяется.

MTPS

8 битов операнда источника заменяет текущее содержимое PS . Адрес операнда источника трактуется, как байтовый адрес. Заметим, что PS бит 4 не может быть установлен данной командой. Источник не изменяется.

Код операции: I064.

Алгоритм: (ист) ⟶ PS.

HALT - Останов

Команда вызывает прекращение работы процессора. Управление передается на пульт. На индикацию пульта выводится адрес команды плюс цва. PC указывает следующую команду, подлежащую выполнению.

При нажатии клавиши "продолжение" на пульте работа процессора возобновляется.

Код операции: 000000.

Условные коды не изменяются.

WAIT - Ожидание

Разрешает процессору отказаться от использования шины, пока он ожидает внешнего прерывания. Получив команду WAIT, процессор не будет запрашивать шину для выборки команд или операндов из памяти. Это повышает скорость передачи данных между устройством и памятью, так как запросы из устройств не сталкиваются с запросами, порожденными процессором. При выполнении процедуры прерывания PC и PS , записанные в стек, указывают адрес, следующей команды. Выход из подпрограммы прерывания (т.е. выполнение команды RTI) возобновляет прерванный процесс на команде, следующей WAIT.

Код операции: 00000I.

Условные коды не изменяются.

RESET - Сброс

Все устройства на общей шине переводятся в исходное состояние.

Код операции: 000005.

Условные коды не изменяются.

5.II. Операторы условного кода

Эти команды имеют следующий формат:

0	0	0	0	0	0	0	0	I	0	I	I/0	N	Z	V	C

I5 5 4 3 2 I0

Разряд 4 указывает, что следует производить по команде установку (I) или сброс (0) кода условий. Разряды 0-3 указывают, какие

из разрядов слова состояния должны быть установлены в I или сброшены (в соответствующие разряды команды записываются единицы; в разряды, на которые воздействовать не нужно - нули). Одна инструкция может установить в I любое сочетание разрядов слова состояния; таким же способом можно и сбросить любое их сочетание.

В табл. 7 перечислены обозначения операторов условного кода в формате Ассемблера.

Таблица 7

Мнемоника	Код операции	Команда
CLC	000241	Гашение условного кода C
CLV	000242	Гашение условного кода V
CLZ	000244	Гашение условного кода Z
CLN	000250	Гашение условного кода N
CCC	000257	Гашение всех битов условного кода
SEC	000261	Включение условного кода C
SEV	000262	Включение условного кода V
SEZ	000264	Включение условного кода Z
SEN	000270	Включение условного кода N
SCC	000277	Включение всех битов условного кода
NOP	000240	Нуль - операция
	000260	

5.I2. Команды плавающей запятой

5.I2.I. Общие сведения

Часть ведущего процессора, выполняющую операции с плавающей запятой, будем называть процессором плавающей запятой.

Процессор плавающей запятой выполняет все арифметические операции с плавающей запятой, преобразование данных из целых чисел в плавающий формат и наоборот. Он использует те же способы адресации и средства памяти, что и ведущий процессор. Команды процессора плавающей запятой могут обращаться к аккумуляторам плавающей запятой, к общим регистрам или к любой ячейке оперативной памяти.

Имеется шесть аккумуляторов плавающей запятой (AC0–AC5), которые используются в командах плавающей запятой. Аккумуляторы (AC) являются 64-битовыми сверхоперативными накопителями. В зависимости от команды аккумулятор может быть использован длиной в 32 бита или в 64 бита. Если AC используется длиной в 64 бита, данные занимает весь аккумулятор, а если в 32 бита, то данные занимают только 32 левых (старших) бита.

На рис. 5 показан формат слова состояния процессора плавающей запятой (FPS) и наименование отдельных битов. Каждый бит описан в табл. 8.

Формат слова состояния процессора плавающей запятой

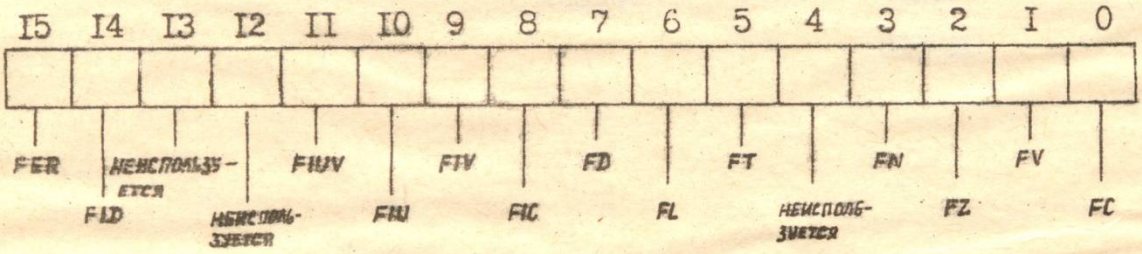


Рис. 5

Таблица 8

Бит	Наименование	Функция
I5	*FER*	Этот бит обозначает условие ошибки процессора плавающей запятой
I4	*FID*	Блокировка прерываний плавающей запятой. Когда этот бит установлен, все прерывания блокируются. Нормально сброшен
I3	Не используется	
I2	Не используется	
I1	*FIUV*	Прерывание плавающей запятой при неопределенной переменной. Когда это бит установлен, и из памяти поступает - 0, происходит прерывание. Если бит не установлен, 0 может быть загружен и запомнен; однако, любая арифметическая операция будет воспринимать его, как положительный 0.
I0	*FIU*	Прерывание плавающей запятой по потере значимости. Когда этот бит установлен, условие потери значимости вызывает прерывание плавающей запятой по потере значимости. Результат операции вызывающей прерывание является правильным за исключением порядка, который превышает 400_8. Если *FIU* не установлен и происходит прерывание, результат устанавливается в нуль
9	*FIV*	Прерывание плавающей запятой по переполнению. Когда этот бит установлен, переполнение плавающей запятой вызывает прерывание. Результат операции, вызывающей прерывание, является правильным за исключением порядка, который превышает 400_8. Если бит *FIV* не установлен, результат операции будет таким же; единственное отличие будет состоять в том, что прерывание не произойдет.

Продолжение табл. 8

Бит	Наименование	Функция
8	FIC	Прерывание плавающей запятой по ошибке преобразования в целое число. Когда этот бит установлен, и команда преобразования из плавающего формата в целое число вызывает установку бита FC (указывая на ошибку преобразования), происходит прерывание. Когда происходит преобразование, регистр приемника сбрасывается, а регистр источника остается без изменения. Когда FIC сброшен, результат операции будет тот же, но прерывание не произойдет
7	FD	Бит режима удвоенной точности. Этот бит, когда он установлен, задает формат удвоенной точности, а когда сброшен, задает формат одинарной точности
6	FL	Бит режима длинных целых чисел. Этот бит используется во время преобразований между форматами с плавающей запятой и целых чисел. Если он установлен, для целого числа задается формат удвоенной точности длиной 32 бита в дополнительном коде; если сброшен, для целого числа задается формат одинарной точности длиной 16 бит в дополнительном коде
5	FT	Бит отсечения. Этот бит, когда он установлен, вызывает отсечение результата любой операции плавающей запятой, вместо округления
4	Не используется	
3-0	FN,FZ,FV,FC	Эти биты являются четырьмя кодами условий плавающей запятой, которые могут быть загружены в биты N, Z, V и C слова состояния процессора (PS), соответственно. Это осуществляется командой перезаписи кодов условий плавающей запятой (CFCC). Для выяснения, как каждая ко-

Продолжение табл. 8

Бит	Наименование	Функция
		манда влияет на коды условий, обращайтесь к табл. I0.

Прерывания процессора плавающей запятой происходит через вектор 244. Может произойти шесть возможных прерываний. Коды прерывания записываются в регистр кодов особых случаев FEC и принимают следующие значения:

2 – Ошибка кода операции плавающей запятой;
прерывание вызывается при ошибочном коде операции.

4 – Деление на нуль плавающей запятой

6 – Ошибка преобразования плавающего формата в целое число.

I0 – Переполнение плавающей запятой.

I2 – Потеря значимости плавающей запятой.

I4 – Неопределенная переменная плавающей запятой.

Если в *FPS* установлен бит *FID* , то все прерывания блокируются.

В дополнение к FEC процессор содержит I6-тибитовый адресный регистр особых случаев плавающей запятой FEA, в котором запоминается адрес последней команды плавающей запятой, вызвавшей особый случай плавающей запятой.

5.I2.2. Форматы данных

Команды плавающей запятой используют целые числа или числа с плавающей запятой. Для лучшего понимания материала приведено несколько примеров чисел с плавающей запятой.

Пример I: Представить десятичное число 75 двоичным нормализованным числом с плавающей запятой

$$75_{I0} = I00I0II_2$$

$$I00I0II,0 \times 2^0 = 0,I00I0II \times 2^7$$

мантисса = 0,I00I0II, порядок = III (т.е. 7)

Пример 2: Представить число 0,25 двоичным нормализованным числом с плавающей запятой

$$0{,}25_{10} = 0{,}01_2$$

$$0{,}01 \times 2^0 = 0{,}1 \times 2^{-1}$$

мантисса = 0,I, порядок = - I

Пример 3: Число 7_{10} сложить с числом 40_{10}, используя представление с плавающей запятой.

$$40_{10} = 50_8 = 0{,}101\ 000\ 000\ 000\ 00 \times 2^6$$

$$7_{10} = 7_8 = 0{,}111\ 000\ 000\ 000\ 000 \times 2^3 = 7_8 = 7_{10}$$

Отметим, что сначала выравниваются порядки, а после этого складываются мантиссы; величина порядка определяет окончательное положение двоичной запятой.

Для выравнивания порядков сдвинуть мантиссу с меньшим порядком на три разряда вправо и увеличить порядок на 3, тогда сложить обе мантиссы.

$$+0{,}101\ 000\ 000\ 000\ 000 \times 2^6 = 50_8 = 40_{10}$$

$$+0{,}000\ 111\ 000\ 000\ 000 \times 2^6 = 7_8 = 7_{10}$$

$$+0{,}101\ 111\ 000\ 000\ 000 \times 2^6 = 57_8 = 47_{10}$$

Для нахождения величины результата в целочисленном выражении сдвинуть двоичную запятую на шесть разрядов вправо.

$$\overbrace{0{,}\underbrace{101}_{5}\ \underbrace{111}_{7}}\ 000\ 000\ 000$$

Пример 4: Умножить 7_{10} на 40_{10}

В умножении с плавающей запятой мантиссы перемножаются, а порядки складываются.

При выполнении умножения с плавающей запятой выравнивания положения двоичной запятой не требуется.

I. $0,111\,0000 \times 2^{3} = 7_{8} = 7_{10}$

x

$0,101\,0000 \times 2^{6} = 50_{8} = 40_{10}$

111

0000

11100

$,1000\,1100\,0000\,00 \times 2^{9} = 280_{10}$ (Результат уже в нормализованной форме).

2. Сдвинуть двоичную запятую на девять разрядов вправо.

4 3 0

$,1000\,1100\,0,\ 00000 = 430_{8} = 280_{10}$

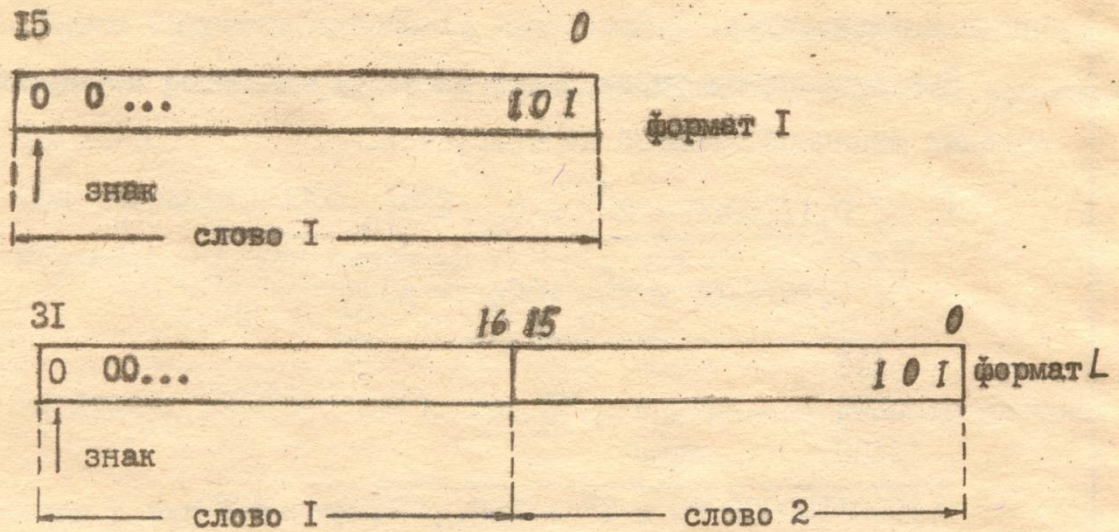
Существует четыре различных формата операндов процессора плавающей запятой:

короткий целочисленный формат (I), длинный целочисленный формат (L), формат плавающей запятой одинарной точности (F) и формат плавающей запятой удвоенной точности (D).

Короткий формат целых чисел имеет длину 16 бит, а длинный формат целых чисел имеет длину 32 бита. Слова данных (операнды) в целочисленном формате представляются в дополнительном коде. Как в I, так и в L форматах самый старший бит слова данных является знаковым битом. Рисунок 6 показывают целые числа 5 и -5 в I и L форматах. Целые числа обрабатываются при выполнении команд загрузки порядка и загрузки и преобразования целого числа в плавающий формат (см.п.5.12.3).

Целое число = 5

Форматы целых чисел



Целое число = - 5

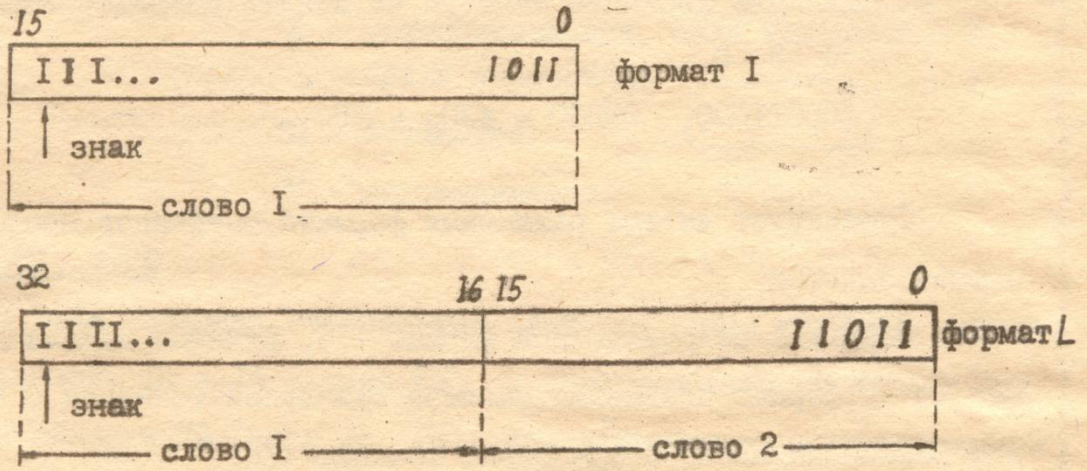
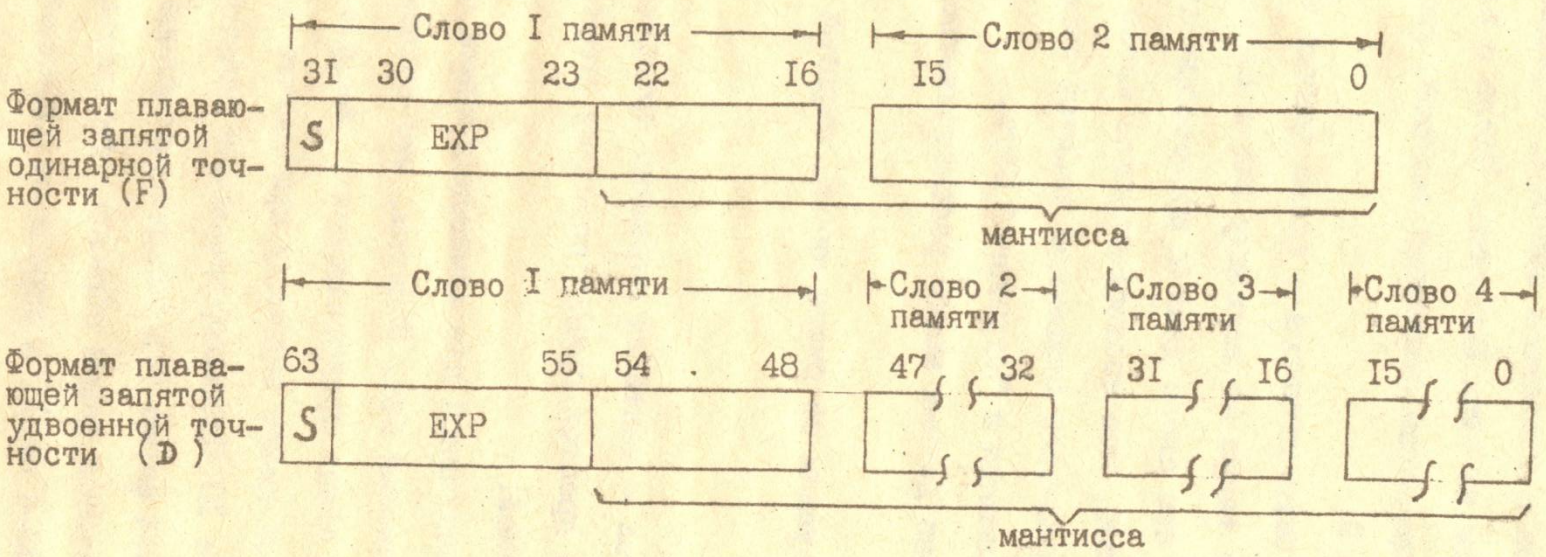


Рис. 6

Числа с плавающей запятой одинарной точности (формат F) имеют длину 32 бита, а удвоенной точности (формат D) имеет длину 64 бита. На рис. 7 показан формат чисел с плавающей запятой. Самый старший бит является знаком мантиссы, следующие 8 битов содержат величину порядка, выраженную представлением с избытком 200, оставшиеся биты (23 для одинарной точности, 55 для удвоенной точности) содержат

Форматы данных с плавающей запятой



S = Знак

EXP = Порядок в представлении с избытком 200

МАНТИССА = 23 или 55 битов в формате знак с величиной

Рис. 7

мантиссу.

Рисунок 8 показывает форматы, в которых числа с плавающей запятой размещаются в памяти. Числа с плавающей запятой, передаваемые в память, должны быть в одном из этих форматов. Числа с плавающей запятой, воспринимаемые процессором плавающей запятой, размещаются, как показано на рисунке 9.

Бит знака, биты порядка и биты мантиссы в слове данных процессора плавающей запятой имеют такую же величину как и слова данных в памяти. Заметим, однако, что слово данных процессора плавающей запятой содержит больше битов, чем соответствующее слово в памяти. Это связано с тем, что процессор плавающей запятой имеет средства генерации бита переполнения и "скрытого" бита.

С целью исследования, слово данных может рассматриваться разделенное на две части:

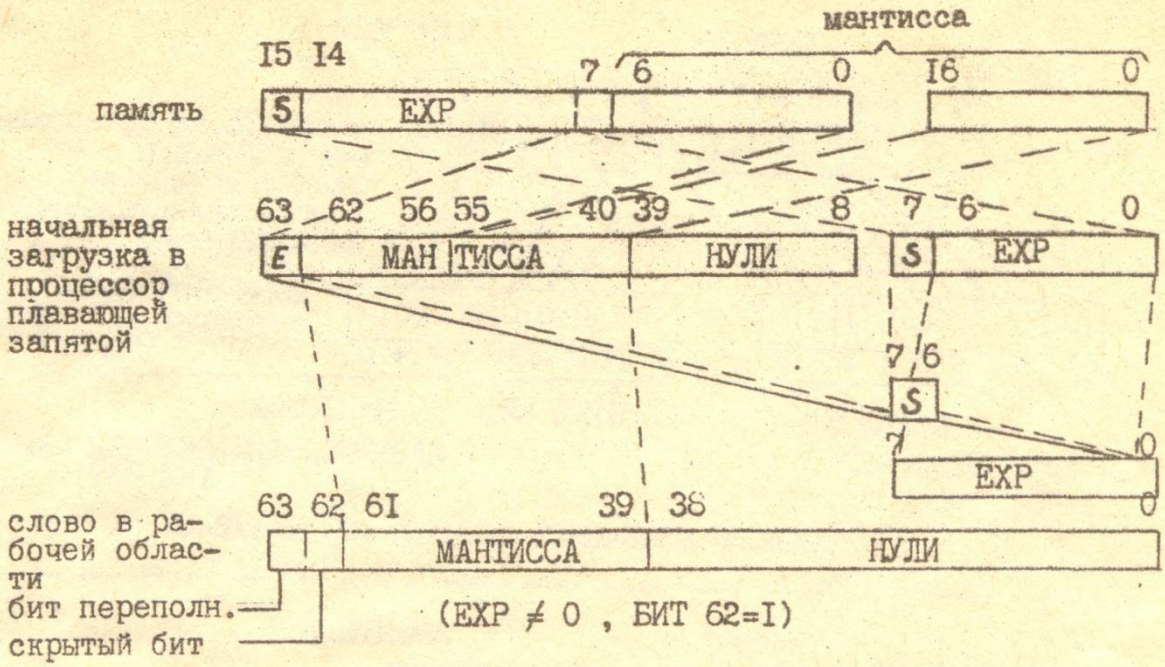
I. Мантисса, с соответствующим ей битом знака, скрытый бит и бит переполнения.

2. Порядок.

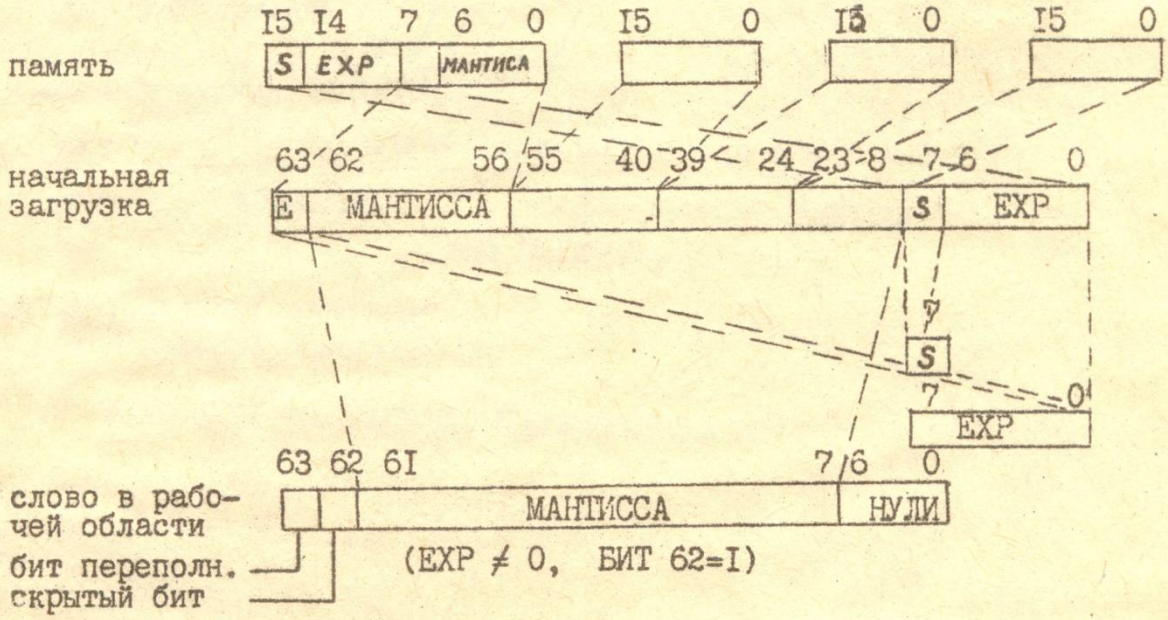
Мантисса плавающей запятой - мантисса выражена представлением знак с величиной. Последующий простой пример поясняет понятие представления знаком с величиной.

В представлении знак с величиной для изменения знака числа требуется изменить только знаковый бит. Заметим, что положительное число выражается одинаково в обеих представлениях. Ненормализованные мантиссы плавающей запятой имеют величину приблизительно от 0 до 2, как показано на рисунке I0. Процессор плавающей запятой, однако, приводит к нормализованной форме все ненормализованные мантиссы. Так что, мантиссы упорядочиваются таким образом, что слева от двоичной запятой находится 0 (63-й бит), а справа от двоичной запятой находится I (62-й бит). Таким образом, величина мантисс находится в пределах от 0,I000 ... до 0,IIII или от I/2 до приблизительно I.

Слова данных с плавающей запятой



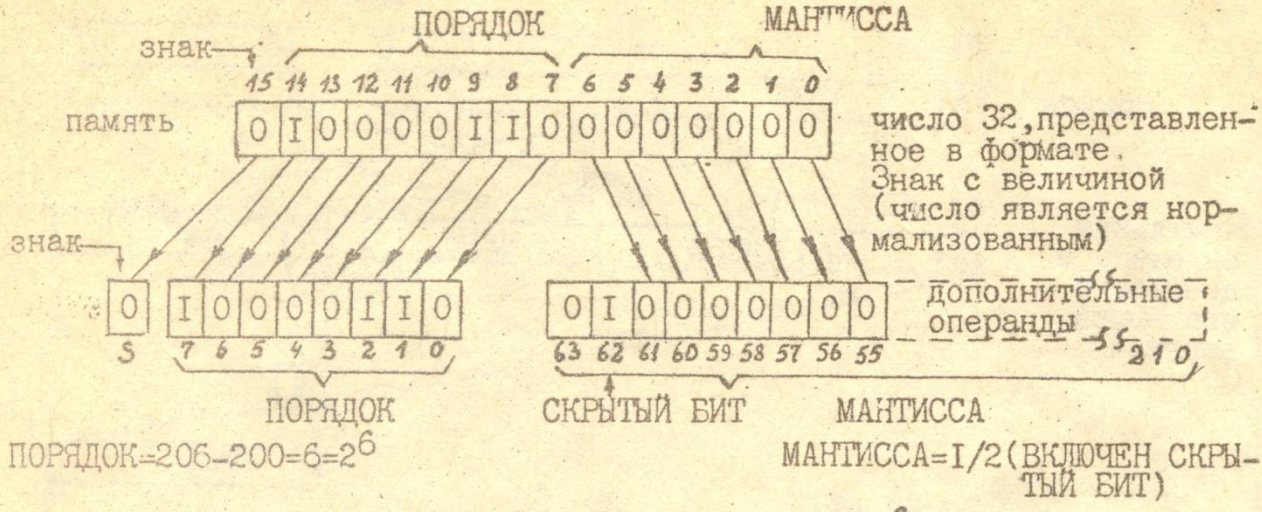
а. Одинарная точность



б. Удвоенная точность

Рис. 8

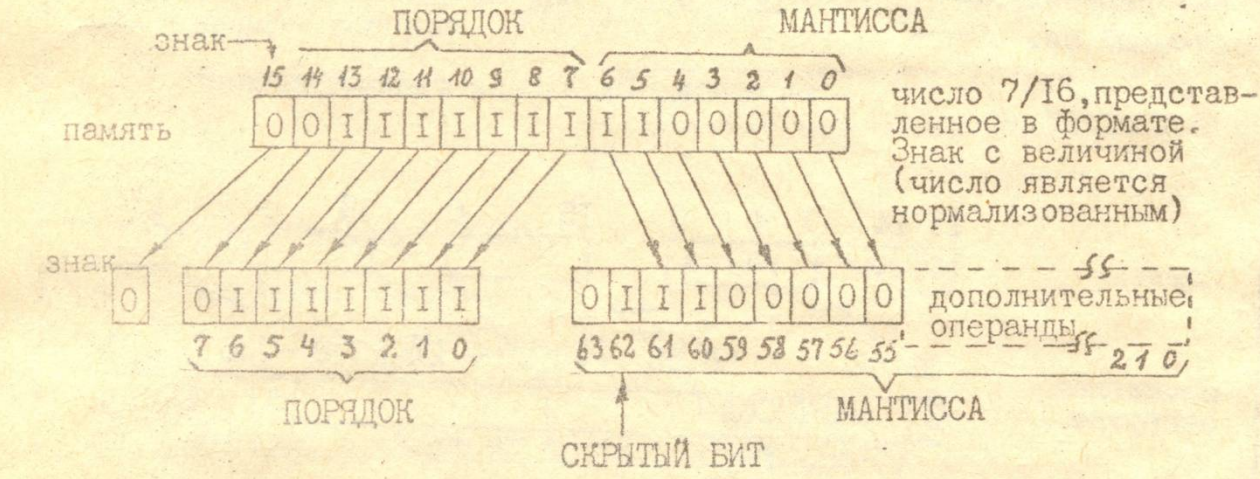
Интерпретация чисел с плавающей запятой



ПОРЯДОК=206-200=6=2^6

МАНТИССА=I/2(ВКЛЮЧЕН СКРЫТЫЙ БИТ)

ЧИСЛО С ПЛАВАЮЩЕЙ ЗАПЯТОЙ =2^6 x I/2=32



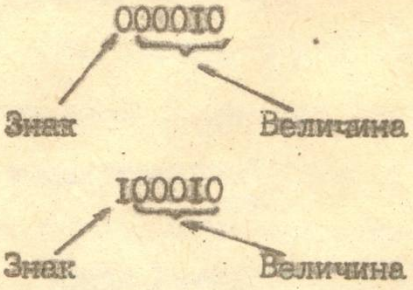
ПОРЯДОК = I77-200= -I=2^{-I}

МАНТИССА=I/2+I/4+I/8=7/8 (включен скрытый бит)

ЧИСЛО С ПЛАВАЮЩЕЙ ЗАПЯТОЙ =2^{-I}x7/8=7/I6

Рис. 9

	Представление в дополнительном коде	Представление знак с величиной
+2	000010	000010
-2	111110	100010



Бит переполнения мантиссы (63-й бит) устанавливается в 1 во время выполнения определенных арифметических операций. Например, во время сложения, когда суммирование таких чисел как 0,1000....+ +0,1000... вызывает переполнение и дает результат 1,000... Это результат может быть нормализован таким образом, что сдвигается мантисса на один разряд вправо и увеличивает порядок на единицу.

62-й бит называется скрытым битом. Вспомним, что после нормализации мантиссы, бит находящийся непосредственно справа от двоичной запятой (62-й бит) всегда является 1. Этот бит опускается, когда мантисса засылается в память, и присоединяется, когда мантисса воспринимается из памяти. Эта процедура предоставляет один дополнительный значащий бит в представлении мантиссы плавающей запятой.

Ненормализованная мантисса плавающей запятой

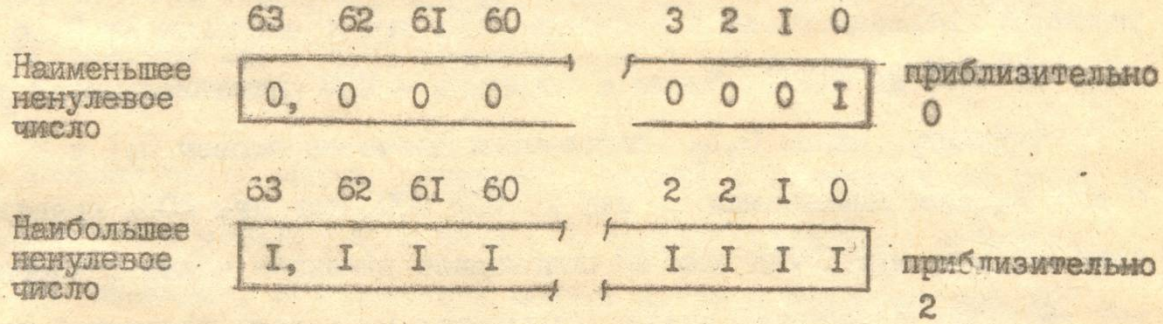


Рис. 10

8-битовый порядок плавающей запятой выражен представлением с избытком 200. Ниже показывается взаимосвязь между порядками, представленными в дополнительном коде, и порядками, представленными с избытком 200.

	Дополнительный код	С избытком 200
Положительные порядки	177 Наибольший положительный порядок	377 Наибольший положительный порядок
	0 Наименьший положительный порядок	200 Наименьший положительный порядок
Отрицательные порядки	377 Наименьший отрицательный порядок	177 Наименьший отрицательный порядок
	200 Наибольший отрицательный порядок	0 Наибольший отрицательный порядок

Заметим, что порядок, выраженный представлением с избытком 200, получается путем простого сложения числа 200 с порядком, выраженным в дополнительном коде. Таким образом, 8-битовый порядок, выраженный представлением с избытком 200, имеет диапазон от 0 до 377 (или от -200 до +177). Число с порядком - 200 рассматривается как 0.

Например, число $0,1_2$ фактически является числом $0,1 \times 2^0$ и его порядок выражается как 10 000 000, так как 200_8 представляет порядок нуля. Рисунок 10 показывает диапазон чисел с плавающей запятой, с которыми может обращаться процессор плавающей запятой.

5.12.3. Команды

Команда плавающей запятой может иметь один из пяти форматов. Эти форматы приведены на рис. II.

Формат команд плавающей запятой

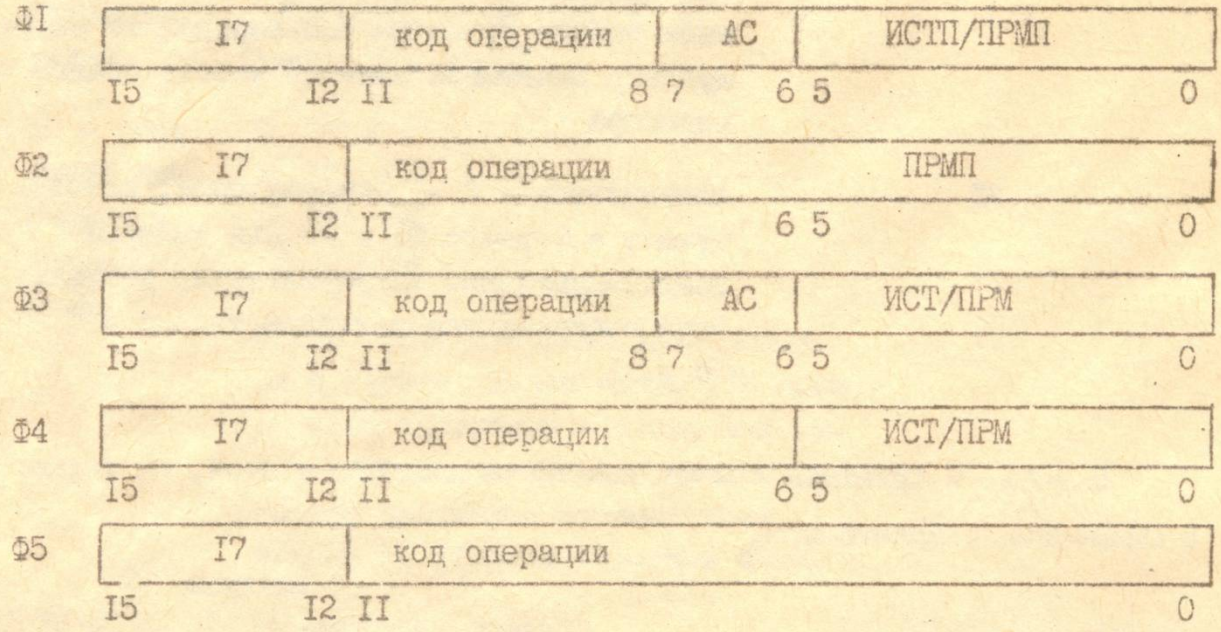


Рис. II

Поля команд различных форматов расшифровываются следующим образом.

Название	Описание
Код операции	Число битов в этом поле является различным, в зависимости от формата; все команды плавающей запятой определяются 4-битовым (I5/I2 биты) кодом операции 17_8, следующие биты используются для определения существующих операций плавающей запятой
ИСТ	Источник - 6-битовое поле источника
ПРМ	Приемник - 6-битовое поле приемника
ИСТП	Источник плавающей запятой - 6-битовое поле, используемое только в формате ФI . Оно идентично полю ИСТ, за исключением режима 0, когда оно определяет аккумулятор плавающей запятой вместо общего регистра

Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата

Название	Описание
ПРМП	Приемник плавающей запятой - 6-битовое поле, используемое в форматах Ф1 и Ф2. Оно идентично полю ПРМ , за исключением режима 0, когда оно определяет аккумулятор плавающей запятой вместо общего регистра
АС	Аккумулятор - 2 - битовое поле, используемое в формата Ф1 и Ф3 для выборки сверхоперативных аккумуляторов, разрешается выбирать аккумуляторы только с 0 по 3.

В табл. 9 приведен список команд плавающей запятой, их формат и мнемоника в Ассемблере.

Таблица 9

Формат команды	Команда	Мнемоника
Ф2	ВЗЯТЬ АБСОЛЮТНОЕ ЗНАЧЕНИЕ	ABSF ПРИП ABSD ПРМП
Ф1	СЛОЖИТЬ	ADDF ИСТП,АС ADDD ИСТП,АС
Ф2	ОЧИСТИТЬ	CLRF ПРМП CLRD ПРМП
Ф4	СРАВНИТЬ	CMPF ИСТП,АС CMPD ИСТП,АС
Ф5	ПЕРЕПИСАТЬ КОДЫ УСЛОВИЙ ПЛАВАЮЩЕЙ ЗАПЯТОЙ	CFCC
Ф1	РАЗДЕЛИТЬ	DIVF ИСТП,АС DIVD ИСТП,АС
Ф1	ЗАГРУЗИТЬ	LDF ИСТП,АС LDD ИСТП,АС

Продолжение табл. 9

Формат команды	Команда	Мнемоника
Ф1	ЗАГРУЗИТЬ И ПРЕОБРАЗОВАТЬ	LDCFD ИСТП,AC LDCDF ИСТП,AC
Ф3	ЗАГРУЗИТЬ И ПРЕОБРАЗОВАТЬ ЦЕЛОЕ ЧИСЛО	LDCIF ИСТ,AC LDCID ИСТ,AC LDCLF ИСТ,AC LDCLD ИСТ,AC
Ф3	ЗАГРУЗИТЬ ПОРЯДОК	LDEXP ИСТ,AC
Ф4	ЗАГРУЗИТЬ СОСТОЯНИЕ ПРОЦЕССОРА ПЛАВАЮЩЕЙ ЗАПЯТОЙ	LDFPS ИСТ
Ф1	ВЗЯТЬ МОДУЛЬ	MODF ИСТП,AC MODD ИСТП,AC
Ф1	УМНОЖИТЬ	MULF ИСТП,AC MULD ИСТП,AC
Ф2	ОТРИЦАТЬ	NEGF ПРМП NEGD ПРМП
Ф5	УСТАНОВИТЬ РЕЖИМ УДВОЕННОЙ ТОЧНОСТИ	SETD
Ф5	УСТАНОВИТЬ РЕЖИМ ПЛАВАЮЩЕЙ ЗАПЯТОЙ	SETF
Ф5	УСТАНОВИТЬ РЕЖИМ ЦЕЛЫХ ЧИСЕЛ	SETI
Ф5	УСТАНОВИТЬ РЕЖИМ ДЛИННЫХ ЦЕЛЫХ ЧИСЕЛ	SETL
Ф1	ЗАПОМНИТЬ	STF AC,ПРМП STD AC,ПРМП
Ф1	ЗАПОМНИТЬ И ПРЕОБРАЗОВАТЬ	STCFD AC,ПРМП STCDF AC,ПРМП
Ф3	ЗАПОМНИТЬ И ПРЕОБРАЗОВАТЬ ПЛАВАЮЩИЙ ФОРМАТ В ЦЕЛОЕ ЧИСЛО	STCFI AC,ПРМ STCFL AC,ПРМ STCDI AC,ПРМ

Инв.№ подл. | Подп. и дата | Взам.инв.№ | Инв.№ дубл. | Подп. и дата

Продолжение табл. 9

Формат команды	Команды	Мнемоника
		STCDL AC,ПРМ
Ф3	ЗАПОМНИТЬ ПОРЯДОК	STEXP AC,ПРМ
Ф4	ЗАПОМНИТЬ СЛОВО СОСТОЯНИЯ ПРОЦЕССОРА ПЛАВАЮЩЕЙ ЗАПЯТОЙ	STFPS ПРМ
Ф4	ЗАПОМНИТЬ СОСТОЯНИЕ	STST ПРМ
Ф1	ВЫЧЕСТЬ	SUBF ИСТП,AC
		SUBD ИСТП,AC
Ф2	ПРОВЕРИТЬ	TSTF ИСТП
		TSTD ИСТП

Какой из форматов плавающей запятой одинарный или удвоенный будет выбранным определяется битом FD из FPS . Если бит FD сброшен, выбирается формат одинарной точности, который обозначается F . Если бит FD установлен, выбирается формат удвоенной точности, который обозначается D . Например, NEGF и NEGD .

Бит FL определяет, какой из целочисленных форматов короткий или длинный является выбранным. Если FL сброшен, выбирается короткий целочисленный формат, который обозначается I, если FL установлен, выбирается длинный целочисленный формат, который обозначается L . Например, команды SETI и SETL .

В табл. I0 все команды плавающей запятой выражены в символическом представлении. Далее, информация, приведенная в табл. I0, дополняется. Некоторые команды используют нижеперечисленные обозначения:

I. MAX - обозначает наибольшее возможное число, которое может быть представлено в формате плавающей запятой. Это число имеет по-

рядок 377 (с избытком 200) и мантиссу, состоящую из одних единиц. Заметим, что МАХ зависит от установленного формата. МИН обозначает наименьшее возможное число, которое не идентично 0. Это число имеет порядок 001 и мантиссу, состоящую из одних нулей за исключением скрытого бита.

2. При описании адресных ячеек используются следующие обозначения:

АБС (адрес) = абсолютное значение (адрес)

EXP (адрес) = порядок (адрес) с избытком 200 .

3. Некоторые восьмеричные коды, приведенные в табл. 10 включены в форму математических выражений. Вычисления над этими кодами могут быть выполнены, как показано в нижеследующих примерах.

Пример 1: команда *LDFPS*

Режим 3, выбран регистр 7 (формат команды Ф4).

I70I00 + ИСТ

Поле ИСТ равно 37

Базовый код операции I70I00

ИСТ и базовый код операции складываются и дают результат I70I37

Пример 2: команда *LDF*

АС2, режим 2, выбран регистр 6 (формат команды ФI)

I72400 + АС × I00 + ИСТП

АС=2

2 × I00 = 200

I72400 + 200 = I72600

ИСТП равно 26

I72600 + 26 = I72626

4. AC∨I означает, что поле аккумулятора (биты 6 и 7 в форматах ФI и Ф3) логически складывается (выполняется операция ИЛИ) с OI.

Пример:

Поле аккумулятора = биты 6 и 7 = AC2 = I0

AC∨I = II

Таблица I0

Мнемоника	Описание команды	Восьмеричный код
ABSD ПРМП	Получение абсолютного значения	I70600 + ПРМП
	ПРМП ← минус (ПРМП), если ПРМП ≤ 0; иначе ПРМП ← (ПРМП)	Формат Ф2
	FC ← 0	
	FV ← 0	
	FZ ← I, если EXP (ПРМП) = 0; иначе FZ ← 0.	
	FN ← 0	
ADDF ИСТП,AC	Сложение с плавающей запятой	I72000+AC × I00 + + ИСТП
ADDD ИСТП,AC	AC ← (AC) + (ИСТП), если |(AC)+(ИСТП)| ≥ МИН; иначе AC ← 0	Формат ФI
	FC ← 0	
	FV ← I, если |AC| ≥ MAX; иначе FV ← 0	
	FZ ← I, если (AC)=0; иначе FZ ← 0	

Мнемоника	Описание команды	Восьмеричный код
	FN ← 1, если (AC) < 0; иначе FN ← 0	
CLRF ПРМП CLRD ПРМП	Очистка ПРМП ← 0 FC ← 0 FV ← 0 FZ ← 0 FN ← 0	170400 + ПРМП Формат Ф2
CMPF ИСТП, AC	Сравнение с плавающей запятой	173400+AC × 100+ +ИСТП
CMPD ИСТП, AC	FC ← 0 FV ← 0 FZ ← 1, если (ИСТП)-(AC)=0; иначе FZ ← 0 FN ← 1, если (ИСТП)-(AC)<0, иначе FN ← 0	Формат Ф1
CFCC	Перезапись кодов условий плавающей запятой C ← FC V ← FV Z ← FZ N ← FN	170000 Формат Ф5
DIVF ИСТП, AC DIVD ИСТП,	Деление с плавающей запятой AC ← (AC)/(ИСТП), если |(AC)/(ИСТП)| ≥ МИН; иначе AC ← 0	174400+AC × 100+ +ИСТП Формат Ф1

Продолжение табл. I0

Мнемоника	Описание команды	Восьмеричный код
	FC ← 0 FV ← I, если |(AC)| ≥ MAX; иначе FV ← 0 FZ ← I, если EXP(AC)=0; иначе FZ ← 0 FN ← I, если (AC) < 0; иначе FN 0	
LDF ИСТП,AC LDD ИСТП,AC	Загрузка с плавающей запятой AC ← (ИСТП) FC ← 0 FV ← 0 FZ ← I, если (AC)=0; иначе FZ ← 0 FN ← I, если (AC) 0; иначе FN ← 0	I72400+AC×I00+ИСТП Формат ФI
LDCDF ИСТП, AC LDCFD ИСТП, AC DF - двойная точность плавающего формата в одинарную FD -одинарная точность плавающего формата в двойную	Загрузка и преобразование двойной точности плавающего формата в одинарную или одинарного формата в удвоенный AC ← $C_{F,D}$ или $C_{D,F}$ (ИСТП), где - $C_{F,D}$ преобразование одинарного плавающего в удвоенный плавающий формат, а $C_{D,F}$ - удвоенного в одинарный FC ← 0 FV ← I, если |AC| ≥ MAX; иначе FV ← 0 FZ ← I, если (AC)=0; иначе FZ ← 0 FN ← I,если (AC) 0; иначе FN ← 0	I77400+AC×I00+ИСТП Формат ФI

Продолжение табл. IO

Мнемоника	Описание команды	Восьмеричный код
	Если текущим форматом является плавающий формат одинарной точности (FD=0), источник воспринимается как число удвоенной точности и преобразовывается в одинарную точность. Если бит отсечения установлен, число отсекается, в противном случае оно округляется. Если текущим форматом является формат удвоенной точности (FD =I), источник воспринимается как число одинарной точности и загружается в левую половину AC. Младшая половина AC очищается.	
LDCIF ИСТ,AC LDCID ИСТ,AC	Загрузка и преобразование целого числа в плавающий формат	I77000+AC * I00 + + ИСТ Формат Ф3
LDCLF ИСТ,AC LDCLD ИСТ,AC	AC ← C IL, FD (ИСТ) FC ← 0 FV ← 0 FZ ← I, если (AC)=0; иначе FZ ← 0 FN ← I, если (AC) 0; иначе FN ← 0	
IF -одинарное целое число в одинарный плавающий формат	C IL, FD - обозначает преобразование целого числа в дополнительном коде точностью I или L,	

Продолжение табл. I0

Мнемоника	Описание команды	Восьмеричный код
FD – одинарное целое число в удвоенный плавающий формат LF – длинное целое число в одинарный плавающий формат. LD – длинное целое число в удвоенный плавающий формат	в число с плавающей запятой точностью F или D . Если IL =0, определяется I6-тибитовое целое число (I), а если IL =I, определяется 32- битовое длинное целое число. Если FD =0, определяется 32- битовый плавающий формат (F), а если FD =I, определяется 64- битовый плавающий формат (D).Если определен формат 32- битового целого числа и используется режим непосредственной адресации или 0-ой режим, I6- левых битов регистра источника сохраняются, а оставшиеся I6 битов сбрасываются до преобразования	
ADEXP ИСТ,AC	Загрузка порядка AC EXP ← (ИСТ) +200 только в случае, если АБС (ИСТ)≤I77 FC ←0 FV ←I, если (ИСТ) < I77; иначе FV ←0 FZ ←I, если EXP(AC)=0; иначе FZ ←0 FN ←I, если (AC) < 0, иначе FN ←0	I76400+AC×I00+ +ИСТ Формат Ф3

Продолжение табл. I0

Мнемоника	Описание команды	Восьмеричный код
LDFPS ИСТ	Загрузка слова состояния программы процессора плавающей запятой FPS ← (ИСТ)	I70I00 + ИСТ Формат Ф4
MODF ИСТП, АС	Команда по модулю плавающей запятой	I7I400+АС × I00 + + ИСТП Формат Ф4
MODD ИСТП, АС	АС V I ← целая часть (АС) × (ИСТП) АС ← дробная часть (АС) × × (ИСТП) - (АС V I), если |(АС) × × (ИСТП)| ≥ МИН или FIU =I; иначе АС ← 0 FC ← 0 FV ← I, если |АС| ≥ МАХ; иначе FV ← 0 FZ ← I, если (АС)=0; иначе FZ ← 0 FN ← I, если (АС) < 0; иначе FN ← 0 Произведение АС и ИСТП содержит 48 битов в формате плавающей запятой одинарной точности или 59 битов в формате плавающей запятой удвоенной точности. Определяется целая часть произведения [(АС) × (ИСТП)] и запоминается в АС V 1. Дробная часть выделяется и запоминается в АС. Заметим, что умножение на I0 может быть выполнено с нулевой ошибкой, позволяя отбросить десятичные	

Продолжение табл. I0

Мнемоника	Описание команды	Восьмеричный код
	разряды без потери точности	
MULF ИСТП,AC MULD ИСТП,AC	Умножение с плавающей запятой AC ← (AC) × (ИСТП), если (AC) × (ИСТП) ≥ МИН; иначе AC ← 0 FC ← 0 FV ← I, если |AC| ≥ MAX ; иначе FV ← 0 FZ ← I, если (AC)=0; иначе FZ ← 0 FN ← I, если (AC) < 0; иначе FN ← 0	I7I000+AC × I00 ИСТП Формат Ф1
NEGF ПРМП NEGD ПРМП	Отрицание ПРМП ← минус (ПРМП), если EXP (ПРМП) ≠ 0; иначе ПРМП ← 0 FC ← 0 FV ← 0 FZ ← I, если EXP (ПРМП)=0; иначе FZ ← 0 FN ← I, если (ПРМП) < 0; иначе FN ← 0	I70700+ ПРМП Формат Ф2
SETD	Установка режима удвоенного плавающего формата FD ← I	I700II Формат Ф5
SETF	Установка одинарного плавающего формата FD ← 0	I7000I Формат Ф5
SETI	Установка формата коротких целых чисел FL ← 0	I70002 Формат Ф5

Продолжение табл. I0

Мнемоника	Описание команды	Восьмеричный код
	FL ← 0	
SETL	Установка формата длинных целых чисел FL ← I	I700I2 Формат Ф5
STF AC,ПРМП STD AC,ПРМП	Запоминание плавающего числа ПРМП ← (AC) FC ← FC FV ← FV FZ ← FZ FN ← FN	I74000+AC × I00+ + ПРМП Формат ФI
STCFD AC, ПРМП STCDF AC, ПРМП	Запоминание и преобразование одинарного в удвоенный плавающий формат или удвоенного в одинарный плавающий формат.	I76000+AC × I00 + + ПРМП Формат Ф1
F@ – одинарный в удвоенный плавающий формат DF – удвоенный в одинарный плавающий формат	ПРМП ← $C_{F,D}$ или $C_{D,F}$ (AC), где $C_{F,D}$ – преобразование одинарного плавающего в удвоенный плавающий, а $C_{D,F}$ – удвоенного в одинарный FC ← 0 FV ← I, если |AC| ≥ MAX ; иначе FV ← 0 FZ ← I, если (AC)=0, иначе FZ ← 0 FN ← I, если (AC) < 0; иначе NF ← 0 Команда STCFD является противоположной команде STCDF ; таким образом, если источник	

Продолжение табл. I0

Мнемоника	Описание команды	Восьмеричный код
	рассматривается как плавающий формат одинарной точности (FD =0), источник воспринимается как число одинарной точности и преобразовывается в удвоенную точность. Если бит отсечения является установленным, число отсекается, в противном случае оно округляется. Если источник рассматривается как плавающий формат удвоенной точности (FD =I), источник воспринимается как число удвоенной точности и загружается в левые (старшие) разряды AC. Младшая половина AC очищается	
STCFI AC,ПРМ STCFL AC,ПРМ STCDI AC,ПРМ STCDL AC,ПРМ	Запоминание и преобразование плавающего формата в целое число Приемник воспринимает преобразованный AC, если преобразованное целое число может быть представлено I6 битами (короткое целое число) или 32 битами (длинное целое число). В противном случае, приемник обнуляется и включается бит FZ.	I75400+AC*I00 + +ПРМ Формат Ф3
FI - одинарный плавающий в одинарное целое FL - одинарный плавающий в длинное целое DI - удвоенный плавающий в	FV ← 0 FZ ← I, если (ПРМ)=0; иначе FZ ← 0	

Продолжение табл. I0

Мнемоника	Описание команды	Восьмеричный код
одинарное целое	FN ← I, если (ПРМ) < 0; иначе FN ← 0	
DL – удвоенный плавающий в длинное целое	C ← FC V ← FV Z ← FZ N ← FN Когда преобразование производится в длинное целое число (32 бита) и режим адресации нулевой или задан непосредственный режим, в регистре приемника запоминаются только I6 старших значащих бита.	
STEXP AC, ПРМ	Запоминание порядка ПРМ ← AC EXP – 200_8 FC ← 0 FV ← 0 FZ ← I, если (ПРМ)=0; иначе FZ ← 0 FN ← I, если (ПРМ) < 0; иначе FN ← 0 C ← FC V ← FV Z ← FZ N ← FN	I75000+AC×I00 + + ПРМП Формат Ф3
STFPS ПРМП	Запоминание слова состояния процессора плавающей запятой (FPS) ПРМ ← (FPS)	I70200+ ПРМП Формат Ф4

Мнемоника		Описание команды	Восьмеричный код
STST	ПРМП	Запоминание состояния ПРМ ← (FEC); ПРМ+2 (FEA) ,если не установлен 0-й или непосредственный режим адресации	I70300+ПРМП Формат Ф4
SUBF SUBD	ИСТП,AC ИСТП,AC	Вычитание с плавающей запятой AC ← (AC)-(ИСТП), если /(AC)-(ИСТП)/ ⩾ МИН; иначе AC ← 0 FC ← 0 FV ← I, если /AC/ ⩾ MAX; иначе FV ← 0 FZ ← I, если (AC)=0; иначе FZ ← 0 FN ← I, если (AC) < 0; иначе FN ← 0	I73000+AC x I00 + +ИСТП Формат Ф1
TSTF TSTD	ПРМП ПРМП	Проверка плавающего числа FC ← 0 FV ← 0 FZ ← I, если EXP (ПРМП)=0; иначе FZ ← 0 FN ← I, если (ПРМП) < 0; иначе FN ← 0	I70500+ ПРМП Формат Ф2

Арифметические команды

Арифметические команды (сложение, вычитание, умножение, деление) требуют одного операнда из источника (аккумулятора плавающей запятой в режиме адресации 0, или ячейки памяти в другом случае) и одного операнда из аккумулятора приемника. Результат запоминается в приемнике аккумулятора.

Команда сравнения всегда требует одного операнда из источника и одного операнда из аккумулятора приемника. После выполнения команды оба операнда остаются на своих местах и изменения результата не происходит.

Команда по модулю плавающей запятой MOD

Команда по модулю для плавающего формата заставляет процессор умножить два операнда плавающей запятой, разделить результат на целую и дробную части и запомнить одну или обе части, как числа с плавающей запятой. Целая часть числа направляется в аккумулятор с печатным номером, а мантисса направляется в аккумулятор с четным номером.

Целая часть числа, выраженная числом с плавающей запятой, содержит порядок больше чем 201 в представлении с избытком 200, что означает, что целое число имеет десятичное значение несколько больше, чем единица и меньше, чем MAX, где MAX является наибольшим числом, которое может быть представлено в процессоре плавающей запятой.

Дробная часть числа, выраженная числом с плавающей запятой, содержит порядок меньше или равный 201 в представлении с избытком 200. Это означает, что мантисса меньше единицы и больше чем МИН, где МИН является наименьшим возможным числом, которое может быть представлено в процессоре плавающей запятой.

Команда загрузки

Команда загрузки заставляет процессор выбрать операнд из источника и переписать его в аккумулятор приемника. Источником является аккумулятор плавающей запятой в режиме адресации 0 или ячейка памяти в других случаях.

Команда запоминания

Команда запоминания заставляет выбрать операнд из аккумулятора источника и переслать его в приемник. Этим приемником является аккумулятор плавающей запятой в режиме адресации 0 или ячейка памяти в других случаях.

Команды загрузки и преобразования (из удвоенного в одинарный формат, из одинарного в удвоенный формат)

Команда загрузки и преобразования из удвоенного в одинарный формат (LDCDF) воспринимает источник, как число с плавающей запятой удвоенной точности. Это число преобразовывает в одинарную точность и результат размещает в аккумуляторе приемника. Если бит состояния плавающего отсечения (FT) установлен, число отсекается. Если бит FT не установлен, число округляется путем присуммирования I к сегменту одинарной точности, если старший бит сегмента удвоенной точности является I, в зависимости от предыдущего условия установки бита FD (рис. I2). Если старший бит сегмента удвоенной точности является 0, слово одинарной точности после округления остается без изменения.

Округление при преобразовании из удвоенной в одинарную точность

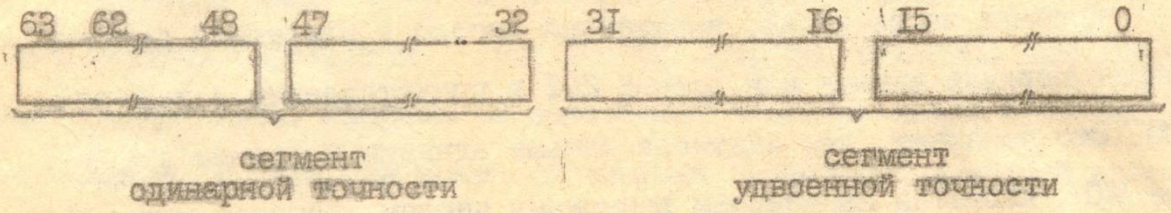


Рис. I2

Команда загрузки и преобразования *одинарного* в удвоенный формат (LDCFD) воспринимает источник как число одинарной точности. Это число преобразовывает в удвоенную точность путем прибавления 32 нулей к слову одинарной точности и размещает этот результат в аккумулятор приемника.

Заметим, что для обеих команд загрузки и преобразования, число, подлежащее преобразованию в источнике, сохраняется без изменения (в аккумуляторе плавающей запятой в режиме адресации 0, или в ячейке памяти в других случаях) и передается в аккумулятор приемника после преобразования.

Команды запоминания и преобразования (из удвоенного в одинарный плавающий формат, из одинарного в удвоенный формат)

Команда запоминания и преобразования удвоенного в одинарный плавающий формат (STCDF) преобразовывает число удвоенной точности, размещенный в аккумуляторе источника, в число одинарной точности, и передает этот результат в заданный приемник. Если бит плавающего отсечения (FT) установлен, число с плавающей запятой отсекается. Если бит FT не установлен, число округляется. Если старший бит (бит 31) сегмента удвоенной точности является 1, 1 присуммируется к сегменту одинарной точности слова, в зависимости от предыдущего условия бита FD (рис. 12); в других случаях сегмент одинарной точности остается без изменения.

Команда запоминания и преобразования одинарного плавающего в удвоенный формат (STCFD) преобразовывает число одинарной точности, расположенное в аккумуляторе источника, в число удвоенной точности, и передает этот результат в заданный приемник. Удвоенная точность из одинарной получается добавлением нулей к эквивалентному сегменту удвоенной точности слова (рис. 13).

Заметим, что для обеих команд запоминания и преобразования, число, подлежащее преобразованию в аккумуляторе источника, сохраня-

ется без изменения и передается в приемник (в аккумулятор плавающей запятой в режиме адресации 0 или в ячейку памяти в других случаях) после преобразования.

Команда очистки

Команда очистки очищает число с плавающей запятой путем установки всех битов в 0.

Команда проверки

Команда проверки проверяет знак и порядок числа с плавающей запятой и изменяет соответственно состояние процессора плавающей запятой. Проверяемое число извлекается из приемника (аккумулятора плавающей запятой в режиме адресации 0, или ячейки памяти в других случаях). Биты FC и FV сбрасываются. Бит FN устанавливается только в том случае, если приемник отрицательный. Бит FZ устанавливается только в случае, если порядок приемника является нулем. Если бит FIUV слова состояния FPS установлен, происходит прерывание (после того, как команда проверки выполнена), когда появляется минус 0.

Отсечение при преобразовании из одинарной в удвоенную точность

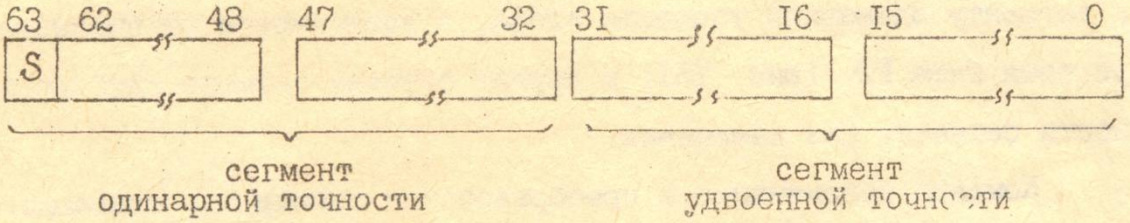


Рис. 13

Команда по абсолютной величине

Команда абсолютной величины заставляет процессор взять абсолютное значение числа с плавающей запятой путем установки его знака в 0. Если определен нулевой режим адресации, знак число в аккумуляторе приемника плавающей запятой превращается в 0. Порядок чис-

ла проверяется также и, если он является 0, нули записываются в аккумулятор. Если порядок не нулевой, аккумулятор остается без изменения.

Если нулевой режим адресации не определен, бит знака определенного слова данных в памяти устанавливается в 0. Порядок этого слова проверяется, и если он является 0, полное слово данных в памяти устанавливается в 0. Если порядок не нулевой, целый порядок восстанавливается в памяти.

Команды абсолютной величины и отрицания являются единственными командами, которые могут считывать ячейки памяти и записывать в них.

Команда отрицания

Команда отрицания заставляет процессор добавить знак операнда. Если определен нулевой режим адресации, знак числа в аккумуляторе приемника плавающей запятой добавляется. Порядок числа проверяется, и если он является 0, нули записываются в аккумулятор. Если порядок не нулевой, аккумулятор остается без изменения.

Если нулевой режим адресации не определен, бит знака определенного слова данных в памяти добавляется. После это слово передается из памяти в аккумулятор плавающей запятой. Порядок этого слова проверяется, и если он является 0, все слово данных устанавливается в 0 и передается обратно в память. Если порядок не нулевой, исходные мантисса и порядок восстанавливаются в памяти.

Команда загрузки порядка

Команда загрузки порядка загружает порядок из источника (аккумулятора плавающей запятой в режиме адресации 0 или ячейки в другом случае) в поле порядка аккумулятора приемника. Чтобы это сделать, 16-битовый порядок в дополнительном коде, находящийся в источнике, должен быть преобразован в 8-битовое число в представ-

ление с избытком 200. Этот процесс описан ниже.

Предположим, что I6-битовый порядок в дополнительном коде поступает из памяти. Возможно допустимая величина I6-битовых чисел в памяти находится в пределах от 000000 до $I77777_8$. С другой стороны, возможно допустимая величина порядков распадается на 2 класса.

I. Положительные порядки (от 0 до I77) - Когда 200 складывается с любым из этих чисел и сумма остается допустимой в пределах 8-битового поля порядка (т.е. от 200 до 377).

2. Отрицательные порядки (от I7760I до I77777) - Когда 200 складывается с любым из этих чисел, и сумма остается допустимой в пределах 8-битового поля порядка (т.е. от I до I77).

Заметим, что все допустимые положительные порядки, поступающие из памяти, имеют нечто общее: в их 9-ти старших битах содержатся все нули. Аналогично, все допустимые отрицательные порядки, поступающие из памяти, имеют 9 старших битов, равных I. Таким образом, для обнаружения допустимых порядков необходимо проверить только 9 старших битов на наличие всех единиц или нулей.

Любое число из памяти, не соответствующее этим пределам, является недопустммым и будет результатом условий прерываний либо по потере значимости, либо по переполнению.

Пример I: LDEXP 000034

Порядок 34

00000000	000III00
+	I0000000
	I00III00
	2 3 4

Все старшие 9 битов равны 0, таким образом, этот положительный порядок является допустимым. Число 234 засылается в 8-битовое поле порядка определенного аккумулятора.

Пример 2: LDEXP 201

Порядок 201	00000000	2 0 1 10000001
	+ 0	10000000
	I	

Переполнение

Это недопустимый положительный порядок. Отметим, что, когда 200 складывается с порядком, происходит переполнение.

Пример 3: LDEXP 100200

		2 0 0
Порядок числа 100200	10000000	10000000
	+	10000000
	I	

Потеря значимости

Это недопустимый отрицательный порядок. Отметим, что, когда 200 складывается с порядком, результат оказывается отрицательным числом большим, чем число, которое может быть размещено в 8-битовом поле порядка. Таким образом, происходит потеря значимости.

Пример 4: Особый случай - Порядок 0: LDEXP 177600

Порядок 177600	1111111	10000000
	+ 0	10000000
	00000000	00000000

Этот случай является единственным, когда все 9 старших битов равны, но порядок является недопустимым. Это происходит потому, что 177600 представляет порядок нуля. Этот порядок вызывает причину наличия переполнения; таким образом, этот случай рассматривает порядок, как недопустимый.

Команда загрузки и преобразования целого числа в плавающий формат

Команда загрузки и преобразования целого числа выбирает из памяти целое число в дополнительном коде и преобразовывает его в число с плавающей запятой в формате знак с величиной. Если определен режим коротких целых чисел, выбираемое из памяти число является 16- битовым и преобразовывается в 24- битовую мантиссу (одинарной точности) или 56-битовую мантиссу (удвоенной точности), в зависимости от того, какой режим установлен – одинарный или удвоенный. Если определен режим длинных целых чисел, выбираемое из памяти число является 32-битовым и преобразовывается в число одинарной или удвоенной точности, в зависимости от того, какой режим установлен – плавающий или удвоенный. Целое число загружается в 55-40 биты, если определен режим коротких целых чисел, или в 55-24 биты, если определен режим длинных целых чисел. После этого число сдвигается влево на восемь разрядов, так что 55 бит перемещается в 63 бит (рис. 14).

После, целому числу присваивается порядок целого короткого числа 217_8. Этот порядок является результатом сложения 200_8 (т.к. порядок является выраженным в представлении с избытком 200) с 17_8, что представляет 15_{10} сдвигов. Это число сдвигов является максимальным числом, необходимым для нормализации числа. Если определен режим длинных целых чисел, целому числу присваивается порядок 237_8, что представляет 31_{10} сдвигов.

Целое число в дополнительном коде проверяется путем анализа 63-го бита, для определения положительным или отрицательным является число. Тогда число нормализовывается, сдвигая его влево, пока 63-й бит станет 1. Если 63-й бит является 1 (число отрицательное), целое число является отрицательным, бит знака устанавливается в 1, число преобразовывается в дополнительный код и тогда нормализовывается.

Для нормализованного числа 63-й бит (старший) мантиссы должен быть равен 0, а 62-й бит должен быть равен I. Чтобы это выполнить, целое число сдвигается влево на необходимое количество разрядов, а значение порядка уменьшается на число сдвинутых разрядов (рис. I5).

$EXP=217_8$	Сдвиг целого числа на I5 разрядов влево для нормализации
$-\ I7_8$	Бит 59=0, бит 58=I
200_8	Уменьшить порядок на $I5_{I0}$, что соответствует $I7_8$

Когда производится загрузка длинного целого числа при FD=0, и длинное целое число содержит больше, чем 24 значащих цифр, последняя значащая цифра будет отсечена, что вызовет некоторую потерю точности.

Команда запоминания порядка

Команда запоминания порядка (STEXP) выбирает число с плавающей запятой, извлекает 8-битовое поле порядка из этого числа и вычитает константу 200 (т.к. порядок выражен представлением с избытком 200). После чего порядок запоминается в приемнике как I6-битовое число в дополнительном коде, расположенное в младших (правых) разрядах, со знаком порядка (бит 07), распространяемым на 8 старших битов.

Допустимые значения порядков лежат в пределах от 0 до 377, в представлении с избытком 200. Это означает, что число, которое запоминается, имеет величину от -200 до I77 после вычитания константы 200. Вычитание числа 200 выполняется преобразованием числа 200 в дополнительный код и сложением его с полем порядка (рис. I6 и I7).

Последующие два примера иллюстрируют этот процесс: один с использованием порядка больше 200, а другой с использованием порядка меньше 200.

Пример сдвига влево целого числа

63	62	6I	60	59	58	57	56	55	54	53	52	5I	50	49	48	47	46	45	44	43	42	4I	40
0	0	0	0	0	0	0	0	I	I	I	I	I	0	0	I	I	I	I	I	0	0	0	I
I	I	I	I	I	0	0	I	I	I	I	I	0	0	0	I	0	0	0	0	0	0	0	0

Рис. I4

Пример нормализации целого числа

63	62	6I	60	59	58	57	56	55	54	53	52	5I	50	49	48	47	46	45	44	43	42	4I	40
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	I	0	0	0	0	0	0	0	0

Рис. I5

Пример I: Порядок =207

Порядок 207
Дополнительный код 200
Результат = 7

```
  I0000III
+ I0000000
  00000III
         7
```

Бит знака

Пример 2: Порядок =42

Порядок 42
Дополнительный код 200
Результат = 42

```
  00I000I0
+ I0000000
  I0I000I0
     4  2
```

Бит знака

Пример № I запоминания порядка

Рис. I6

Пример № 2 запоминания порядка

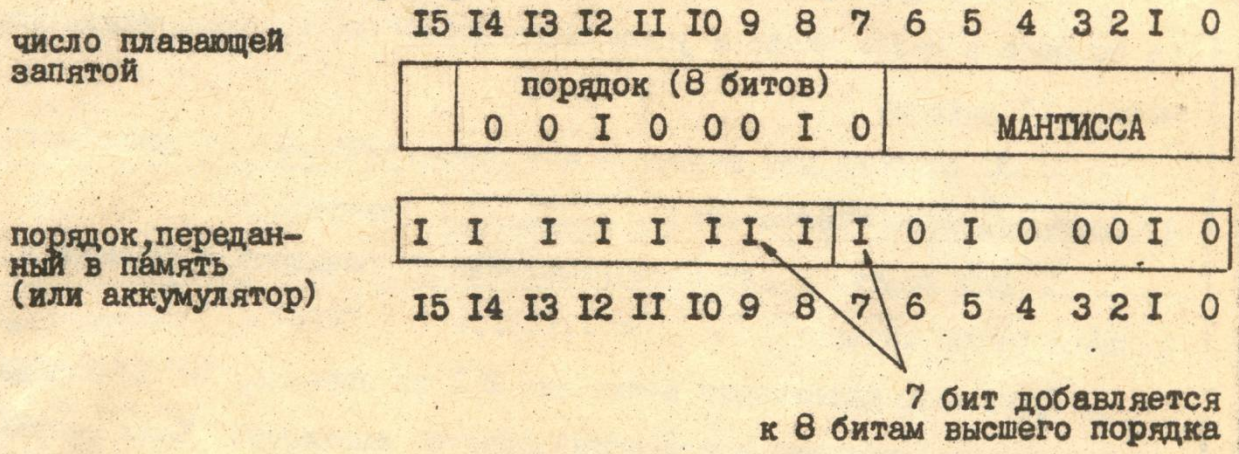


Рис. I7

Команда запоминания и преобразования плавающего формата в целое число

Команда запоминания и преобразования плавающего формата в целое число заставляет процессор плавающей запятой выбрать число с плавающей запятой и преобразовать его целое число для последующей передачи в приемник.

Имеются следующие четыре класса этих команд.

I. STCFI - Преобразовать 24-битовую мантиссу одинарной точности в I6-битовое целое число (режим коротких целых чисел).

2. STCFL - Преобразовать 24-битовую мантиссу одинарной точности в 32-битовое целое число (режим длинных целых чисел).

3. STCDI - Преобразовать 56-битовую мантиссу удвоенной точности в I6-битовое целое число (режим коротких чисел).

4. STCDL - Преобразовать 56-битовую мантиссу удвоенной точности в 32-битовое целое число (режим длинных чисел).

Число с плавающей запятой (нормализованное), которое должно быть преобразовано, передается в процессор плавающей запятой. Процессор производит обработку бита знака и выделяет одну из следующих частей.

I. I5 старших битов мантиссы для преобразования из плавающего формата в целое число и удвоенного целого числа в плавающий формат.

2. 3I старший бит мантиссы для преобразования из удвоенного формата в длинное целое число.

3. Всю мантиссу для преобразования из плавающего формата в длинное целое число.

Процессор производит вычитание 20I из порядка, чтобы определить, является ли число с плавающей запятой мантиссой. Если результат вычитания является отрицательным, т.е. порядок меньше чем 20I, абсолютное значение числа с плавающей запятой меньше единицы. При

преобразовании этого числа в целое, значение его устанавливается в 0; происходит ошибка преобразования, бит FZ устанавливается и 0 засылается в приемник. Если результат вычитания является положительным (или нулевым), это показывает, что порядок больше (и равен) 20I, и число с плавающей запятой может быть преобразовано в ненулевое целое число (рис. I8).

Следующей проверкой, которую выполняет процессор , является определение пределов значений числа с плавающей запятой с целью определения в каком формате целых чисел оно может быть представлено - I6-битовом (I-формате) или 32-битовом (L - формате).

В табл. II рассмотрим значения целых чисел, которые могут быть представлены в I и L форматах и их эквиваленты в формате с плавающей запятой.

Пример запоминания и преобразования целого числа

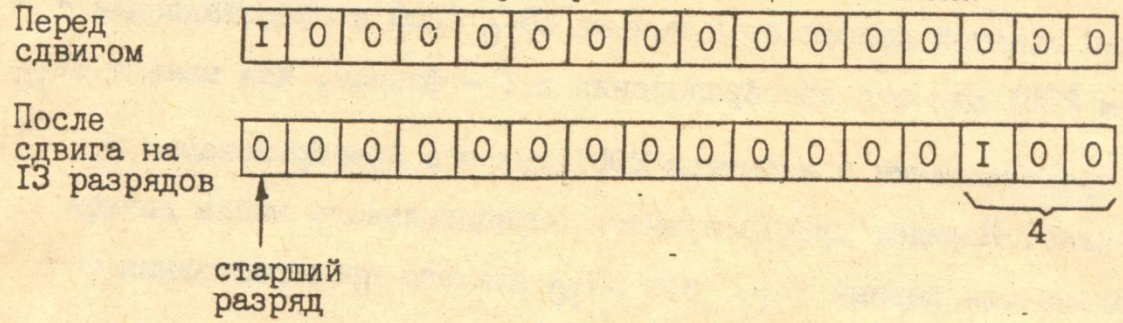


Рис. I8

Таблица II

	I-формат	Эквивалент с плавающей запятой	L- формат	Эквивалент с плавающей запятой
Наибольшее положительное целое число	077777	$+,IIII...x2^{I5}$	I7777777777	$+,III...x2^{3I}$
Наименьшее положительное целое число	00000I	$+,I00...\ 2^{I}$	0000000000I	$+,I00...x2^{I}$

Продолжение табл. II

	I-формат	Эквивалент с плавающей запятой	L-формат	Эквивалент с плавающей запятой
Наименьшее отрицательное целое число	I77777	$-,IIII...x2^{I6}$	37777777777	$-,IIII...2^{32}$
Наибольшее отрицательное целое число	I00000	$-,I000...x2^{I6}$	20000000000	$-,I00...2^{32}$

Примечание. Старший разряд целого числа - знак целого числа.

Таким образом, порядок преобразуемого положительного числа с плавающей запятой должен быть меньше $I6_{IO}$ (220 в представлении с избытком 200) для его преобразования в I - формат, или меньше 32_{IO} (240 в представлении с избытком 200) для его преобразования в L - формат. Порядок преобразуемого отрицательного числа должен быть меньше или равный $I6_{IO}$ или 32_{IO} для его преобразования в I - или L - формат, соответственно.

Процессор плавающей запятой проверяет, находится ли преобразуемое число в пределах целых чисел, которые могут быть представлены в I- или L -форматах, путем вычитания константы 20_8 (для коротких целых чисел) или 40_8 (для длинных целых чисел) из результата первой проверки (результат первой проверки - смещенный порядок $20I_8$ = не смещенный порядок - I). Если результат вычитания положительный или равен 0, это показывает, что число с плавающей запятой является слишком большим, чтобы представить его в форме целого. В этом случае имеет место ошибка преобразования и в приемник засылаются нули. Если результатом вычитания является отрицательное число

отличное от - I, число с плавающей запятой может быть представлено как целое, не вызывая условия переполнения. Если результат вычитания равен -I, порядок числа с плавающей запятой равен или 220 (короткое целое число), или 240 (длинное целое число), и преобразование происходит. Однако, число с плавающей запятой находится внутри допустимых значений только в том случае, если его знак отрицательный, а его мантисса равна, I00... (т.е., если оно является наибольшим отрицательным целым числом; смотри таблицу выше). Если, в этом случае, число не является наибольшим отрицательным целым числом, оно будет обнаружено третьей проверкой ошибки преобразования (смотри ниже) после преобразования.

Для преобразования мантиссы в целое число, мантисса сдвигается вправо на число разрядов, определенных следующими алгоритмами.

Для коротких целых чисел: Число сдвигов вправо =

$$20_8 + 20I_8 - \text{смещенный порядок} - I$$

Для длинных целых чисел: Число сдвигов вправо =

$$40_8 + 20I_8 - \text{смещенный порядок} - I$$

Не обращая внимания на условие бита FT, дробная часть числа всегда отсекается во время выполнения этих сдвигов.

Если число с плавающей запятой является положительным, преобразование в целое число завершается после сдвига, и число передается в соответствующий приемник. Если, однако, число с плавающей запятой является отрицательным, целое число должно быть преобразовано в дополнительный код, до пересылки его в приемник.

После преобразования, выполняется третья проверка на ошибку преобразования путем сравнения старшего бита (преобразованного) целого числа с битом знака первоначального (непреобразованного) числа. Если эти знаки не совпадают, это указывает на ошибку преобразования и происходит прерывание программы, если бит FIC

установлен. Эта проверка выполняется для обнаружения числа с плавающей запятой с порядком 220 (короткое число) или 240 (длинное число), которое не может быть преобразовано в наибольшее отрицательное целое число.

Пример I: Запомнить и преобразовать плавающий формат в целое число (STCFI)

Порядок 203

Знак = 0

Мантисса (24 бита) =, I00000000000000000000000

I5 ст.битов мантиссы = ,I00000000000000

203 (с избытком 200)= 2

Мантисса = I/2 Целое число для запоминания =I/2x2=4

I. Проверка I: Является ли число, подлежащее преобразованию, мантиссой ?

Порядок: 203_8

$20I$

Нет 2

Таким образом этот результат является положительным, данное число с плавающей запятой не является мантиссой и преобразование может быть выполнено без ошибки.

2. Проверка 2: Находится ли преобразовываемое число с плавающей запятой внутри допустимых значений ? (Мы работаем с положительным коротким целым числом).

Результат проверки I : 2

-20

Да $-I6$

Показывает, что число, подлежащее преобразованию, находится внутри допустимых значений и может быть представлено как I6-битовое целое число. Ошибки преобразования не происходят.

Сколько требуется сдвигов вправо? Применяем алгоритм:

$20_8 + 20I_8 - 203_8 - I = 20_8 - 3_8 = I5_8 = I3_{I0}$

= I3 сдвигов вправо

Этот пример включает в себя положительное число, так что он завершается после I3 сдвигов вправо. Если число было бы отрицательным, преобразованное целое число должно было быть переведено в дополнительный код.

3. Проверка 3

Старший бит преобразованного целого числа и бит знака первоначального числа с плавающей запятой сравниваются. Так как они равны, ошибки преобразования не происходит.

Команда загрузки состояния процессора плавающей запятой

Эта команда передает I6 бит из ячейки, заданной источником, в регистр состояния плавающей запятой (FPS). Эти I6 битов содержат информацию состояний для ее использования, чтобы разрешать или блокировать прерывания, устанавливать и гасить биты режимов и устанавливать коды состояний (см.п. 5.I2.I).

Команда запоминания слова состояния процессора плавающей запятой

Эта команда передает I6 битов регистра FPS в заданный приемник.

Команда запоминания состояния плавающей запятой (STST)

Эта команда считывает содержимое регистров кода особых случаев плавающей запятой (FEC) и адреса особых случаев плавающей запятой (FEA), когда особый случай плавающей запятой (ошибка) происходит.

Если установлен режим адресации 0, в аккумулятор приемника засылается только FEC. Если режим адресации 0 не установлен, FEC запоминается в памяти следом за FEA. В памяти данные FEA занимают все I6 битов своей ячейки памяти, тогда как данные FEC занимают только 4 младших бита своей ячейки.

Когда происходит ошибка и прерывание не заблокировано, управление передается вектору прерывания 244, где пускается команда STST для определения типа ошибки.

Команда STST может быть использована только после появления ошибки, так как во всех других случаях команда содержит неопределенные данные или содержит условия, которые возникли после последней ошибки.

Команда перезаписи кодов условий плавающей запятой CFCC

Эта команда переписывает четыре кода условий плавающей запятой из FPS(FC, FZ, FV, FN) в коды условий PS(C, Z, V, N).

Команда установки плавающего режима SETF

По этой команде сбрасывается бит FD(07-ой бит регистра FPS); это указывает на работу с одинарной точностью.

Команда установки режима удвоенной точности SETD

Эта команда устанавливает бит FD(07-ой бит регистра FPS); это указывает на работу с удвоенной точностью.

Команда установки режима целых чисел SETI

Эта команда сбрасывает бит IL (06-ой бит регистра FPS). Это указывает, что установлен режим коротких целых чисел (16 битов).

Команда установки режима длинных целых чисел SETL

Эта команда устанавливает бит IL (06-ой бит регистра FPS). Это указывает, что установлен режим длинных целых чисел (32 бита).

5.12.4. Примеры программирования

Приведены два примера программирования с использованием команд плавающей запятой. В примере 1, A складывается с B, E вычитается и C, величина (A+B) умножается на (C-E), произведение этого умножения делится на X, а результат запоминается в стек.

Пример 2 вычисляет выражение $EX^3 + CX^2 + BX+A$, и включает в себя 3-кратный цикл, а результат помещается в стек.

Пример I: $(A+B)\times(C-E)/X$

```
SETF
LDF  A,AC0         ;ЗАГРУЗИТЬ AC0 ИЗ A
ADDF B,AC0         ;AC0 СОДЕРЖИТ (A+B)
LDF  C,AC1         ;ЗАГРУЗИТЬ ACI ИЗ C
SUBF E,ACI         ;ACI СОДЕРЖИТ (C-E)
MULF ACI,AC0       ;AC0 СОДЕРЖИТ (A+B)×(C-E)
DIVF X,AC0         ;AC0 СОДЕРЖИТ (A+B)×(C-E) /X
STF  AC0,-(6)      ;ЗАПОМНИТЬ (A+B)×(C-E)/X В СТЕК
```

Пример 2: $EX^3 + CX^3 + BX+A$

$$AC0 = \underbrace{\left[\overbrace{\underbrace{(E\times X+C)}_{\text{цикл I}}\times X+B}^{\text{цикл 2}}\right]\times X+A}_{\text{Цикл 3}}$$

$$AC0 = \left[EX^2 + CX+B\right]\times X + A$$

$$AC0 = EX^3 + CX^2 + BX+A$$

```
        SETF
        MOV  #3,%0        ;УСТАНОВИТЬ СЧЕТЧИК ЦИКЛОВ
        MOV  #E+4,%I      :УСТАНОВИТЬ УКАЗАТЕЛЬ РАВНЫЙ КОЭФФИЦИЕНТАМ
        LDF  (6)+,ACI     ;ИЗВЛЕЧЬ X ИЗ СТЕКА
        CLRF AC0          ;ОЧИСТИТЬ AC0
MOOP;   ADDF -(4),AC0     ;ПРИБАВИТЬ СЛЕДУЮЩИЙ КОЭФФИЦИЕНТ
                          ;К ЧАСТНОМУ РЕЗУЛЬТАТУ
        MULF ACI,AC0      ;УМНОЖИТЬ ЧАСТНЫЙ РЕЗУЛЬТАТ НА X
        SOB  %0,MOOP      ;ВЫПОЛНИТЬ ЦИКЛ 3 РАЗА
        ADDF -(4),AC0     ;ПРИБАВИТЬ X К ВЫЧИСЛЕННОМУ РЕЗУЛЬТАТУ
        STF  AC0,-(6)     ;ПОМЕСТИТЬ РЕЗУЛЬТАТ В СТЕК
```

ПРИЛОЖЕНИЕ I

ОБОЗНАЧЕНИЯ, ПРИМЕНЯЕМЫЕ ПРИ ОПИСАНИИ КОМАНД

(XXX)	- содержимое XXX
ист	- адрес источника
прм	- адрес приемника
$\wedge$	- функция "И" (логическое умножение)
$\vee$	- функция "ИЛИ" (логическое сложение)
$\veebar$	- функция "Исключающее ИЛИ" (сложение по модулю 2)
$\overline{\quad}$	- черта сверху - функция "НЕ" (инверсия)
$\rightarrow$	- "становится"
Z	- разряд признака нуля в слове состояния
N	- разряд признака минус в слове состояния
C	- разряд признака переноса в слове состояния
V	- разряд признака переполнения в слове состояния
PC	- счетчик команд
SP	- указатель стека
PS	- слово состояния процессора
R	- универсальный регистр
ВР	- внутренний регистр процессора
АС	- аккумулятор плавающей запятой
FPS	- слово состояния процессора плавающей запятой

Лист регистрации изменений

Изм.	Номера листов (страниц) изменен-ных	заменен-ных	новых	аннули-рован-ных	Всего листов (страниц) в докум.	№ докум.	Входящий № сопрово-дительного докум. и дата	Подпись	Дата

Инв. № подл. 16-2978 | Подп. и дата Абтю - 83.4.6. | Взам. инв. № | Инв. № дубл. | Подп. и дата

Изм.	Стр.	№ докум.	Подп.	Дата	I.700.010T01	Стр. 119