КМ1801ВМ2

16-РАЗРЯДНЫЙ МИКРОПРОЦЕССОР

Предисловие.

В данном документе я постарался собрать все сведения, известные мне о процессоре КМ1801ВМ2. Эта документация является компиляцией сведений о процессоре из различных источников, а также сведений, полученных при изучении процессора на компьютере «Электроника МС-0511» УКНЦ. Так как сведения о процессоре, находящиеся в сети Internet, весьма скудные, поэтому я постарался собрать воедино все информацию о процессоре и акцентировать внимание на нестандартных ситуациях в его работе. Информация о назначении выводов и реализации циклов обмена с системной шиной взята из стандартной документации и описана поверхностно. Более глубоко рассмотрена работа процессора по исполнению команд и работе блока обработки прерываний, и их реакции на нестандартные ситуации (зависание, работа в режиме HALT, переход между режимами). Поэтому в данном случае этот документ будет полезен программистам и эмуляторописателям. Исследование процессора проводились на компьютере УКНЦ, поэтому при описании процессора будут описаны особенности его работы на данной машине.

Использованные источники информации:

1. Мячев А.А., Степанов В.Н. «Персональные ЭВМ и микроЭВМ. Основы организации». М., «Радио и связь», 1991. С. 60–74: Глава 2.5. «Микропроцессорные комплекты серий К1801/К1809».
2. «1806ВМ2, Н1806ВМ2», описание микропроцессора, з-д «Ангстрем» – 1806vm2.pdf
3. «Описание процессоров серии 1801/1806» – http://www.vak.ru/doku.php/proj/bk/1801vm-series
4. «DCJ11 Microprocessor User's Guide», DEC, 1983 – dcj11.pdf
5. Захаров И.В. «Техническое обслуживание и эксплуатация микроЭВМ «Электроника 60М». М., «Машиностроение», 1989.
6. ЭВМ «Электроника МС 0511». Техническое описание. У11.700.016 ТО.

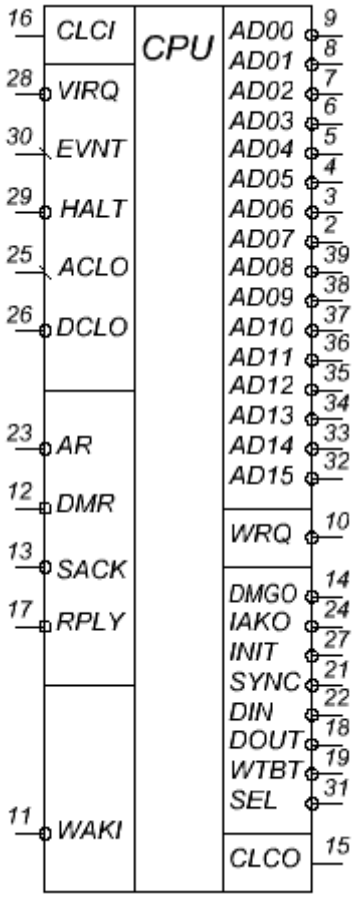
Основные характеристики.

КМ1801ВМ2 – БИС 16-разрядного микропроцессора (МП) с архитектурой, совместимой с микроЭВМ LSI-11 фирмы DEC. Микропроцессор выполнен по технологии n-MOS и выпускается в 40-контактном металлокерамическом или пластмассовом корпусах.

- 16-разрядный микропроцессор
- технология n-MOS, быстродействие до 1 млн. регистровых сложений/сек
- тактовая частота до 10 МГц
- напряжение питания +5В±10%
- системная магистраль МПИ (QBUS)
- возможность работы в двух режимах: USER и HALT
- 4 линии запроса на прерывания: две по уровню, две по обратному фронту сигнала
- максимальный размер адресного пространства: 128 Кбайт (64 Кбайт в режиме USER и 64 Кбайт в режиме HALT)
- 11 программно доступных регистров, из них 2 доступны только в режиме HALT
- набор из 77 команд:
  1. Базовый набор команд (+SOB, XOR, MTPS, MFPS)
  2. Команды расширенной арифметики EIS
  3. Спецгруппа команд режима HALT
  4. Возможность эмуляции команд арифметики с плавающей запятой (FIS)

Особенностью данного микропроцессора является возможность работы в двух режимах: USER и HALT, каждому из которых соответствует своё адресное пространство. Режим USER предназначен для работы программ пользователя. Режим HALT предназначен для реализации программ начального пуска системы, пультового отладчика, эмуляции команд FIS и обработки фатальных ситуаций. Так как адресное пространство HALT может не пересекаться с адресным пространством USER, то это даёт возможность реализовать выше перечисленные программы не на микропрограммном уровне, а на программном, в результате чего достигается большая гибкость. Из режима работы HALT можно получить доступ к адресному пространству USER, а в режиме работы USER получить доступ к адресному пространству HALT средствами микропроцессора невозможно.

Описание выводов и структуры микропроцессора.



Функциональное назначение выводов:

Вывод	Обозначение	Функциональное назначение
1	0V	Общий усилителей шины адреса–данных
2	AD07 (АД07)	7-й разряд шины адреса–данных
3	AD06 (АД06)	6-й разряд шины адреса–данных
4	AD05 (АД05)	5-й разряд шины адреса–данных
5	AD04 (АД04)	4-й разряд шины адреса–данных
6	AD03 (АД03)	3-й разряд шины адреса–данных
7	AD02 (АД02)	2-й разряд шины адреса–данных
8	AD01 (АД01)	1-й разряд шины адреса–данных
9	AD00 (АД00)	0-й разряд шины адреса–данных
10	WRQ	Запрос обмена с окном межпроцессорного обмена
11	WAKI	Разрешение обмена с окном
12	DMR (ТПД)	Запрос ПДП

Вывод	Обозначение	Функциональное назначение
13	SACK (ПВ)	Подтверждение захвата шины ПДП
14	DMGO (ППД)	Разрешение ПДП
15	CLCO	Выход тактовой частоты
16	CLCI	Вход тактовой частоты
17	RPLY (СИП)	Ответ пассивного устройства
18	DOUT (ВЫВОД)	Сопровождение записи
19	WTBT (БАЙТ)	Запись байта
20	0V	Общий внутренних схем процессора
21	SYNC (СИА)	Синхросигнал активного устройства
22	DIN (ВВОД)	Сопровождение чтения
23	AR	Адрес принят
24	IAKO (ППР)	Разрешение прерывания
25	ACLO (ПИТ)	Авария сетевого питания
26	DCLO (ПОСТ)	Авария источника питания
27	INIT (СБРОС)	Сброс внешних устройств
28	VIRQ (ТПР)	Запрос векторного прерывания
29	HALT (ОСТ)	Запрос радиального прерывания HALT
30	EVNT (ПРТ)	Прерывание по событию
31	SEL	Выбор безадресного регистра / обращение к адресному пространству режима HALT
32	AD15 (АД15)	15-й разряд шины адреса–данных
33	AD14 (АД14)	14-й разряд шины адреса–данных
34	AD13 (АД13)	13-й разряд шины адреса–данных
35	AD12 (АД12)	12-й разряд шины адреса–данных
36	AD11 (АД11)	11-й разряд шины адреса–данных
37	AD10 (АД10)	10-й разряд шины адреса–данных
38	AD09 (АД09)	9-й разряд шины адреса–данных
39	AD08 (АД08)	8-й разряд шины адреса–данных
40	+5V	Питание

Описание выводов:

AD00 – AD15 – 16 выводов сигналов адреса и данных, которые передаются по магистрали. У МП совмещённая шина адреса и данных, при адресном обмене сначала на шине выставляется адрес, а затем выставляются данные, если производится запись, или выводы AD00-AD15 переключаются в режим входов для чтения данных с магистрали. Активный уровень – низкий, т.е. электрический ноль соответствует логической единице.

SYNC – синхросигнал начала адресного обмена по магистрали. После выставления адреса устройства на выводах AD00-AD15 обратным фронтом этого сигнала устройству (ОЗУ или контроллеру периферийного устройства) даётся команда сохранить адрес в своем буферном регистре адреса. Активный уровень – низкий.

AR – адрес принят устройством. Устанавливается устройством в ответ на сигнал SYNC. Используется для асинхронного обмена по магистрали. В случае использования синхронного обмена и для совместимости с шиной QBUS (МПИ) выход SYNC необходимо соединить с входом AR через буферный усилитель. Активный уровень – низкий.

DIN – сигнал, вырабатываемый МП, и свидетельствующий о том, что МП производит чтение данных с устройства, приём вектора прерывания, или чтение безадресного регистра. При активном низком уровне SYNC производится чтение данных с устройства. При пассивном высоком уровне SYNC и активном сигнале IAKO производится приём вектора прерывания. При пассивном сигнале SYNC и активном SEL производится чтение безадресного регистра. Активный уровень – низкий.

DOUT – сигнал, вырабатываемый МП, и сообщающий устройству, что на шине AD00-AD15 выставлены данные и производится запись. Активный уровень – низкий.

WTBT – сигнал, сообщающий устройству, что производится запись байта. Активный уровень – низкий. Имеет смысл только при активном низком уровне сигнала DOUT.

RPLY – синхросигнал пассивного устройства, сообщающий МП, что чтение/запись данных произведены, или устройство передало вектор прерывания. Активный уровень – низкий.

SEL – выбор адресного пространства HALT или чтение безадресного регистра. Во время адресного обмена выбирается адресное пространство HALT, а при неактивном уровне SYNC и активном DIN – безадресный регистр. Активный уровень – низкий.

VIRQ – сигнал, вырабатываемый внешним устройством, если оно готово принять или передать данные в режиме прерывания. Активный уровень – низкий.

IAKO – сигнал, вырабатываемый МП, в момент входа в режим обработки векторного прерывания, затребованного по запросу VIRQ. Сигнал транслируется по цепочке через все внешние устройства. Устройство, которое затребовало прерывание, прекращает трансляцию этого сигнала. Активный уровень – низкий.

DMR – сигнал, вырабатываемый устройством прямого доступа, при необходимости захватить магистраль. Активный уровень – низкий.

DMGO – сигнал, вырабатываемый МП, сообщающий устройству ПДП о том, что оно может захватить магистраль. Активный уровень – низкий.

SACK – сигнал, вырабатываемый устройством ПДП для удержания МП в пассивном состоянии. Активный уровень – низкий.

WRQ – запрос обмена с окном. Появляется в случае адресного обмена, если адрес содержится в диапазоне 160000_8-163777_8. Обмен начнётся только после выставления сигнала WACK активным низким уровнем. Активный уровень – низкий.

WACK – подтверждение обмена с окном. Активный уровень – низкий. В случае использования диапазона адресов 160000_8-163777_8, как обычного адресного пространства, вход WACK должен быть соединён с землей.

EVNT – сигнал требования прерывания по внешнему событию. Активным является обратный фронт сигнала, по которому в МП устанавливается триггер. В УКНЦ используется для этого кадровый синхросигнал видеоадаптера, поступающий на вход EVNT с частотой 50 Гц. В качестве вектора используется значение 100_8.

DCLO – авария постоянного питания. Активный низкий уровень используется для сброса МП в исходное состояние. Значение сигнала на входе DCLO также транслируется на выход INIT для приведения внешних устройств в исходное состояние. После снятия сигнала DCLO процессор должен быть запущен положительным фронтом сигнала ACLO.

ACLO – авария источника питания. Отрицательный фронт вызывает установку триггера прерывания по вектору 24_8. Положительный фронт инициирует микропрограмму пуска МП по вектору начального пуска $SEL000_8$ после снятия сигнала DCLO.

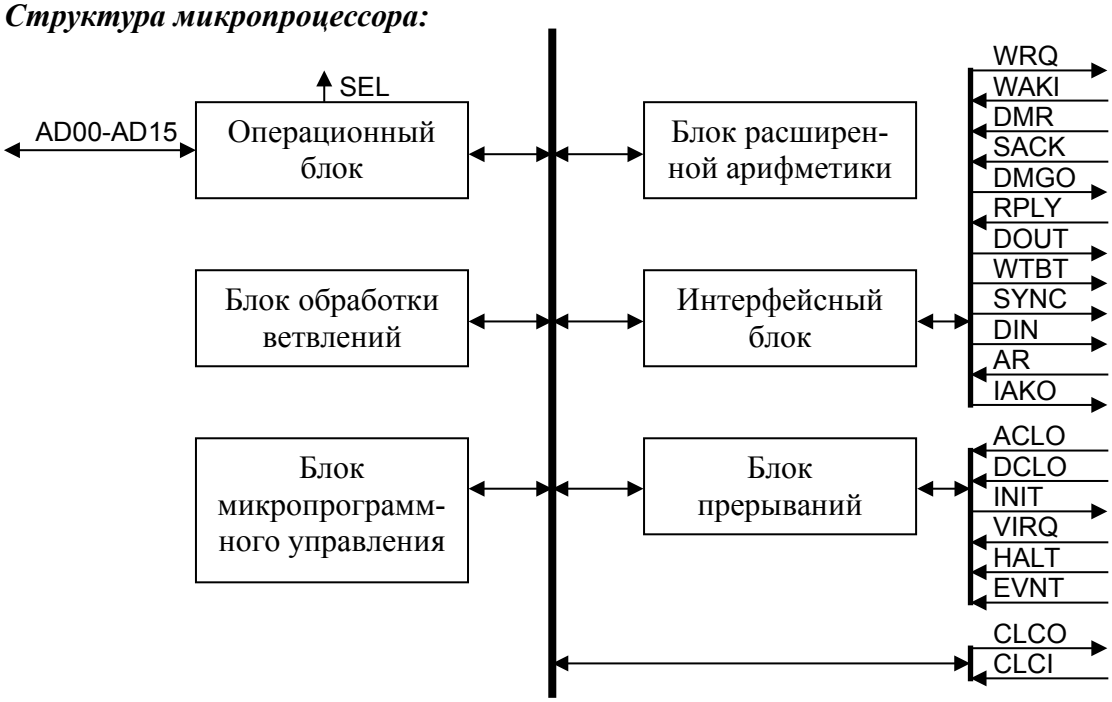
INIT – сигнал, формируемый МП для приведения внешних устройств в исходное состояние. Активный уровень – низкий. Вырабатывается в случае активного низкого уровня сигнала DCLO, а также по команде МП RESET.

HALT – сигнал радиального прерывания для перевода МП в режим HALT и запуска программы пультового отладчика. Активный низкий уровень вызывает прерывание по вектору $SEL170_8$.

CLCI – входная тактовая частота.

CLCO – выходная тактовая частота. Равна половине входной тактовой частоты CLCI.

Структура микропроцессора:



В состав операционного блока входят: буферный регистр данных (БРД), регистр адреса (РА), регистр копии счётчика команд (СРС), компаратор адресов (КОМП), буферный регистр команд ОБ (БРК ОБ), регистр команд ОБ (РК ОБ), блок констант, регистр состояния процессора (PSW), копия регистра состояния процессора (CPSW), регистры общего назначения (R0 – R7), аккумулятор (АК), регистр источника (РИ), 16-разрядное АЛУ, сдвигатель (СДВ), блок обмена байтов (БОБ) и схема записи (СхЗп). ОБ обеспечивает вычисление адреса и его временное хранение в РА, прием данных и хранение в регистрах, выполнение арифметических и логических операций между регистрами или между регистрами и константами, выдачу данных в системную магистраль, формирование адресов векторов прерывания, формирование состояний. Программно доступны только регистры R0 – R7, PSW, СРС и CPSW. Регистры СРС и CPSW программно доступны только в режиме процессора HALT.

Блок микропрограммного управления вырабатывает последовательности микрокоманд на основе принятого кода команды.

Блок прерываний служит для приема и предварительной обработки сигналов прерываний, вырабатывает также адрес вектора прерывания для выборки его из блока констант операционного блока и код прерывания для переключения блока МПУ на микропрограмму обработки прерываний. В блоке находится логика аппаратной поддержки выполнения команды RESET.

Блок расширенной арифметики предназначен для аппаратной поддержки выполнения команд умножения, деления и параметрического сдвига MUL, DIV, ASH, ASHC. Для выполнения содержательной части команды используется одна микрокоманда, которая модифицируется по определённому алгоритму схемой управления блоком расширенной арифметики в процессе выполнения операции. Каждая команда РА выполняется за определённое число тактов. Внутренние регистры блока расширенной арифметики доступны по адресу в микрокоманде.

Блок обработки ветвлений обеспечивает подачу управляющего сигнала ветвления на основе кода команды и признаков ветвления N, Z, V и C, вырабатываемых в операционном блоке при выполнении команд.

Интерфейсный блок служит для организации обменов между процессором и устройствами на системной магистрали. В интерфейсном блоке содержится арбитр прямого доступа к памяти, совмещённый со схемой запроса окон. Арбитр прямого доступа отслеживает поступление запроса на прямой доступ. После поступления запроса процессор заканчивает текущий цикл обмена и выдает разрешение на прямой доступ. Во время прямого доступа процессор останавливается. После вычисления адреса и записи его в регистр адреса в интерфейсный блок поступает запрос на обмен. Если системная магистраль, или дополнительная магистраль, захваченная через окно, свободна, то начинается выдача адреса на магистраль.

Процессор принимает команды с опережением. Алгоритм приёма команд построен так, что к концу выполнения команды следующая команда уже принята на буферный регистр команд, и

начинается прием ещё одной команды. При выполнении любых команд, загружающих PC, аппаратно осуществляется повторный приём следующей команды. Для восстановления опережения в этой же команде подготавливается приём еще одной команды. Нарушение опережения происходит и тогда, когда по счётчику команд читается не команда, а данные. В конце команды с адресацией данных по счётчику команд также производится восстановление опережения путем организации чтения двух последующих команд. Дешифратор команд запускается, если команда принята на буферный регистр команд и установлен признак того, что предыдущая команда окончилась. В последней микрокоманде команды запускается блок прерываний, и если есть незамаскированное прерывание, процессор запускает микропрограмму обработки прерывания. Если в поле обмена микрокоманды есть признак обращения к системной магистрали, операционный блок вырабатывает сигнал запроса на обмен, поступающий в интерфейсный блок. Если системная магистраль свободна, интерфейсный блок выполняет цикл обмена по магистрали, а операционный блок в это время может выполнять микрокоманды, не связанные с обменом или связанные с подготовкой к следующему обмену.

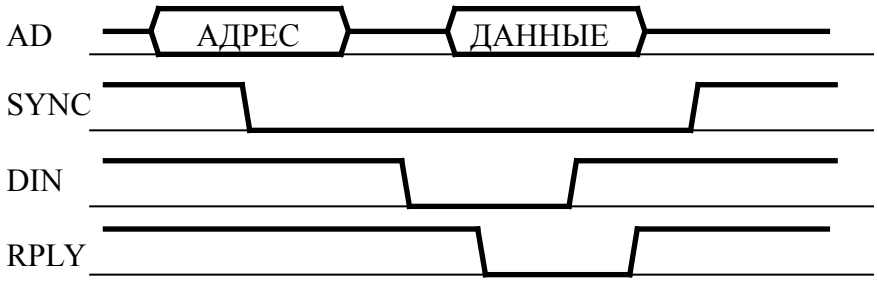
Основные циклы обращения к каналу.

Для функционирования компьютера микропроцессор должен выбирать коды данных и команд из ОЗУ и загружать их, то есть выполнять элементарные циклы ВВОД и ВЫВОД. Если выбираемые из внешней памяти данные вновь загружаются в ту же ячейку, то рациональнее вместо двух циклов ВВОД и ВЫВОД использовать совместный цикл ВВОД-ПАУЗА-ВЫВОД.

Выполнение любой команды процессор начинает с цикла ВВОД. По значению счётчика команд из памяти выбирается код команды. Затем в зависимости от команды и применяемых в ней методов адресации, либо осуществляется конечное число цикло обращения к магистрали, либо проводится выборка следующей команды.

В УКНЦ сигнал асинхронного обмена AR не используется, поэтому временные диаграммы будут показаны без использования этого сигнала.

Цикл ВВОД (адресное чтение).

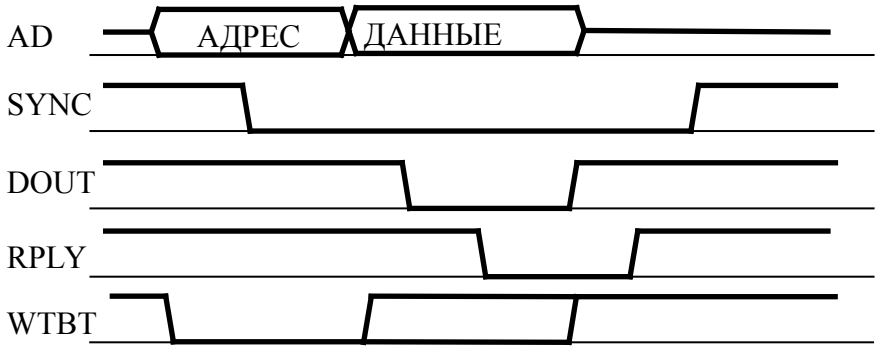


Сначала процессор выставляет на шине адреса–данных адрес считываемой ячейки. Если производится чтение памяти адресного пространства HALT, во время выдачи адреса также устанавливается в активное состояние сигнал SEL. Через один такт CLCO процессор устанавливает сигнал SYNC в активное состояние. Отрицательный фронт сигнала используется для записи адреса (или его части) в буферный регистр адреса устройства. После формирования сигнала SYNC МП снимает адрес с магистрали. Затем вырабатывается сигнал DIN. В ответ на этот сигнал внешнее устройство выставляет на шину адреса–данных считываемые данные и формирует сигнал ответа RPLY. В ответ на RPLY процессор считывает данные с шины адреса–данных и снимает сигналы DIN и SYNC. После снятия сигнала DIN внешнее устройство снимает сигнал RPLY.

Во время цикла ВВОД всегда происходит выборка полного 16-разрядного слова, поэтому внешние устройства обычно игнорируют младший бит адреса AD00 во время выставления адреса устройства на магистраль. Если использовалась команда байтового чтения, то во время цикла ВВОД прочитывается слово, а выделение нужного байта уже происходит в операционном блоке МП.

Если за 54 такта после выставления сигнала DIN устройство не ответило сигналом RPLY, то процессор снимает сигналы DIN и SYNC и прерывается по зависанию при передаче данных.

Цикл ВЫВОД.

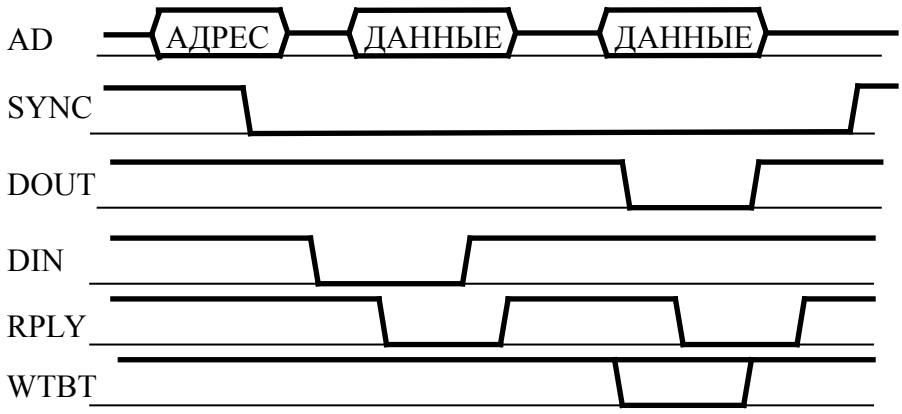


Сначала процессор выставляет на шине адреса–данных адрес записываемой ячейки. Если производится запись в память адресного пространства HALT, во время выдачи адреса также устанавливается в активное состояние сигнал SEL. Через один такт CLCO процессор устанавливает сигнал SYNC в активное состояние. Отрицательный фронт сигнала используется для записи адреса (или его части) в буферный регистр адреса устройства. После формирования сигнала SYNC МП снимает адрес с магистрали. Затем на шину адреса–данных выставляются записываемые данные и вырабатывается сигнал DOUT. Если производится запись байта, одновременно с выставлением данных переходит в активное состояние и сигнал WTBT. В ответ на сигнал DOUT внешнее устройство записывает данные и формирует сигнал ответа RPLY. В ответ на RPLY процессор снимает данные с шины адреса–данных и сигналы DOUT, SYNC и WTBT переводит в неактивное состояние. После снятия сигнала DOUT внешнее устройство снимает сигнал RPLY.

Во время цикла ВЫВОД, если производится запись слова, то обычно внешнее устройство игнорирует младший бит адреса AD00. Во время записи байта младший разряд адреса AD00 показывает запись какого байта – старшего или младшего, выполняется в данный момент. Если в фазе выдачи адреса AD00 был равен нулю, то байт содержится в младшей части (разряды 00-07), а при равенстве AD00 единице, записываемый байт содержится в старшей части (разряды 08-15).

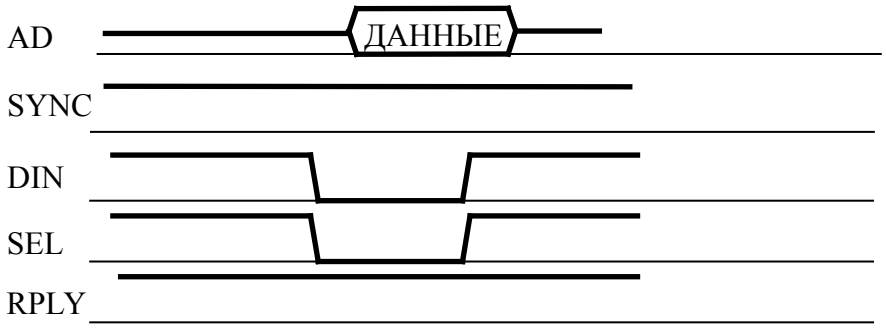
Если за 54 такта после выставления сигнала DOUT устройство не ответило сигналом RPLY, то процессор снимает сигналы DOUT, SYNC и WTBT и прерывается по зависанию при передаче данных.

Цикл ВВОД-ПАУЗА-ВЫВОД.



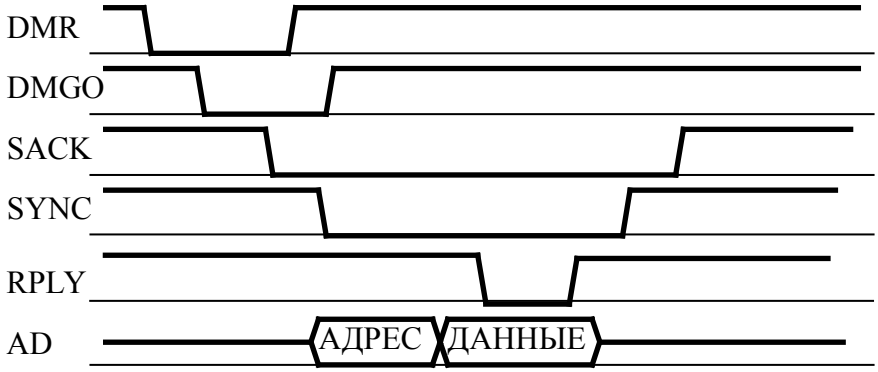
При выполнении ряда команд после обработки операндов результат засылается по адресу последнего выбранного операнда. А поскольку его адрес обычно хранится в буферном регистре адреса устройства, то имеется возможность пропустить адресную часть цикла ВЫВОД. Особенностью цикла ВВОД-ПАУЗА-ВЫВОД является сохранение сигнала SYNC в активном состоянии после снятия сигнала RPLY. Сигнал WTBT формируется при необходимости одновременно с данными, загружаемыми в ячейку (совместно с сигналом DOUT).

Безадресное чтение.



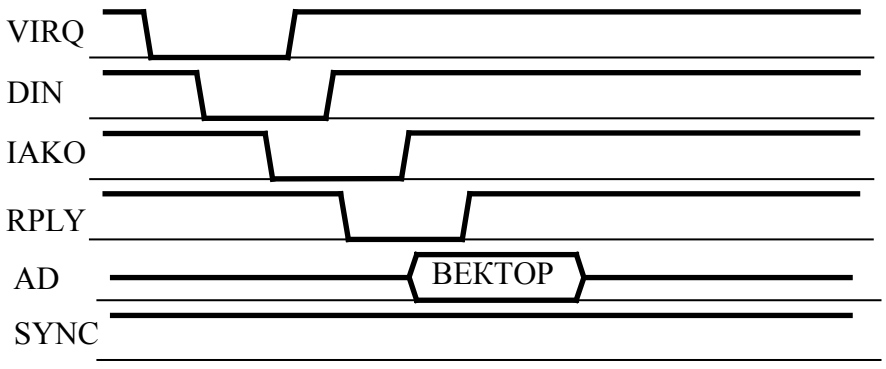
МП имеет специальную процедуру синхронного обмена – безадресное чтение. Эта процедура выполняется во время начального пуска процессора, при обработке фатальных прерываний (зависание в режиме HALT, двойное зависание, зависание при приёме адреса вектора прерывания), прерывания по сигналу HALT, выполнении команды HALT и группы команд FIS, а также при исполнении команды чтения безадресного регистра. Особенностью этого цикла является то, что сигнал SYNC остаётся в неактивном уровне и ответа сигналом RPLY не требуется.

Предоставление прямого доступа к памяти.



Устройство ПДП, требующее прямого доступа к памяти сначала выставляет сигнал запроса ПДП DMR. МП в ответ на сигнал DMR выставляет сигнал разрешения захвата шины DMGO. По приходу сигнала DMGO устройство ПДП снимает сигнал DMR и выставляет сигнал подтверждения захвата шины SACK. По сигналу SACK процессор снимает сигнал DMGO и переводит все выходы в неактивное состояние (кроме CLCO, DMGO, SEL, IAKO). После этого управление магистралью осуществляется устройством ПДП с использованием штатных циклов обмена (ВВОД, ВЫВОД). После завершения обмена устройство ПДП снимает сигнал SACK и МП продолжает работу с точки останова.

Приём вектора прерывания.

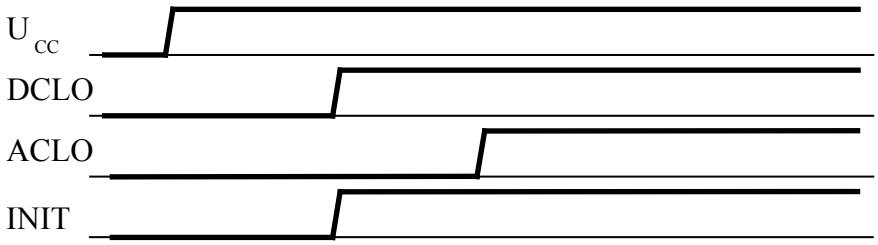


Устройство, которому необходимо обслуживание, выдаёт в канал сигнал запроса векторного прерывания VIRQ. По окончании обработки команды в случае, если запрос VIRQ не замаскирован, МП переходит в режим обработки прерывания. Микропрограмма обработки прерывания двумя циклами ВЫВОД сохраняет в стеке (адресация по регистру R6(SP)) регистр

копии состояния процессора CPSW и регистр копии счетчика команд CPC. Затем процессор последовательно выдает на шину сигналы DIN и IAKO. Сигнал DIN, поступая на интерфейсы всех устройств, подключенных к каналу, обеспечивает в устройстве, требующем прерывания, выработку условия запрета трансляции через него сигнала IAKO. После выработки сигнала DIN вырабатывается сигнал IAKO, который, распространяясь по приоритетной цепочке и достигая первого устройства, требующего прерывания, обеспечивает выдачу им на шину адреса-данных кода вектора прерывания и сигнала RPLY. Сигнал SYNC в этом цикле не вырабатывается. По сигналу IAKO устройством снимается сигнал VIRQ. Процессор, принимая сигнал RPLY, снимает сигналы DIN и IAKO, после этого устройство снимает с шины код вектора прерывания и сигнал RPLY. После завершения цикла приёма вектора прерывания, МП считывает из двух последовательно расположенных ячеек, адрес первой из которых задаётся кодом вектора прерывания, новые значения счетчика команд R7 (PC) и слова состояния процессора PSW.

Если через 54 такта после выставления сигнала IAKO ни одно устройство не ответило сигналом RPLY, то процессор прерывается по вектору зависания при приёме адреса вектора прерывания со значением $SEL274_8$.

Процедура начального пуска.



После подачи питающего напряжения сигналы DCLO и ACLO должны находится в низком уровне. Низкий уровень сигнала DCLO обеспечивает сброс внутренних схем МП и трансляцию сигнала DCLO на выход INIT, в результате чего обеспечивается приведение всех внешних устройств в исходное состояние. После стабилизации питающего напряжения не ранее чем через 40 мс снимается сигнал DCLO, что в свою очередь обеспечивает снятие сигнала INIT. После не ранее чем через 70 мс снимается сигнал ACLO. Положительный фронт сигнала ACLO после снятия сигнала DCLO обеспечивает запуск процессора по вектору начального пуска с адресом $SEL000_8$. Для этого МП переходит в режим HALT и по процедуре безадресного чтения считывает безадресный регистр SEL и формирует адрес вектора начального пуска АВП[15:8]=SEL[15:8], АВП[7:0]=0. Затем из вектора начального пуска загружаются счётчик команд R7=(АВП) и слово состояния процессора PSW=(АВП+2), проверяются источники прерывания, и если есть незамаскированные, то процессор прерывается, иначе грузится очередная команда и начинается выполнение.

Описание алгоритма работы микропроцессора.

Программно доступные регистры микропроцессора.

Для программиста микропроцессор представлен 11-ю программно доступными регистрами. Это восемь 16-разрядных регистров общего назначения, именуемых с R0 по R7, при этом регистр R6 является указателем стека и имеет еще имя SP (Stack Pointer), а регистр R7 является счётчиком команд с именем PC (Program Counter). Слово состояния процессора PSW (Processor Status Word) – 9-разрядный регистр, в котором хранятся признаки выполнения команд, бит трассировки, бит запрета прерываний и бит режима работы процессора (USER или HALT). Кроме этого имеются ещё два регистра, которые программно доступны только в режиме работы процессора HALT – это 16-разрядный регистр копии счётчика команд CPC и 9-разрядный регистр копии слова состояния процессора CPSW.

Формат регистров общего назначения (РОН):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

РОН представляют собой 16-разрядное регистры, которые доступны для команд работы с 16-разрядными словами. При работе с байтами доступны только 8 младших разрядов (с 7 по 0).

Регистр R6 (SP) кроме этого является указателем стека. Он используется при обработке прерываний режима USER, возврата из прерываний, вызова подпрограмм.

Регистр R7 (PC) является счётчиком команд. По его значению микропроцессором прочитывается команда из памяти. После чтения команды значение PC увеличивается на 2, и указывает на следующее слово после исполняемой команды.

Формат регистра слова состояния процессора:

8 H/U	7 P	6	5	4 T	3 N	2 Z	1 V	0 C

Разряд 8 H/U – режим работы микропроцессора - HALT (1) или USER (0). В режиме работы HALT во время адресного обмена на магистраль выдаётся сигнал SEL, что дает возможность использовать отдельное адресное пространство для этого режима. Программы пользователя выполняются в режиме USER. Режим HALT предназначен для реализации программ пультового отладчика, для программной реализации команд FIS, а также для обработки фатальных ситуаций.

Разряд 7 P – приоритет работы процессора. При установленном разряде маскируются прерывания EVNT и VIRQ.

Разряды 6 и 5 могут быть установлены и сброшены, но не оказывают никакого влияния на работу процессора. Введены для совместимости с другими процессорами, где приоритет работы выставляется разрядами 7-5.

Разряд 4 T – бит трассировки. Если этот разряд установлен, то после исполнения команды будет вызвано прерывание программы по вектору 14_8. Это даёт возможность организовать пошаговую отладку программы.

Разряды с 0 по 3 определяют коды условий ветвления и содержат информацию о результатах выполнения последней команды.

Разряд 3 N – после исполнения команды получился отрицательный результат.

Разряд 2 Z – после исполнения команды результат равен нулю.

Разряд 1 V – после исполнения команды произошло арифметическое переполнение.

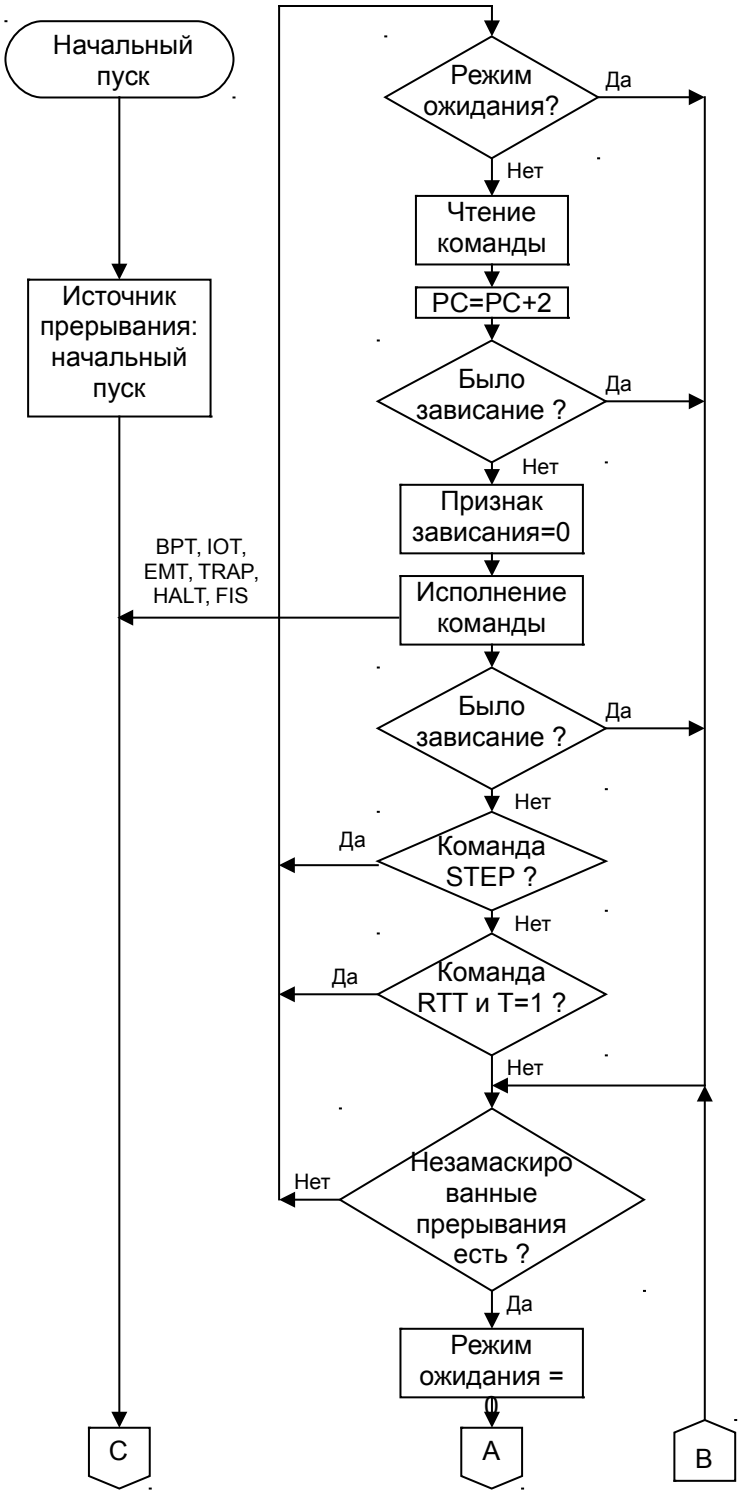
Разряд 0 C – после исполнения команды произошёл перенос из самого старшего разряда, а при сдвиге вправо – из самого младшего.

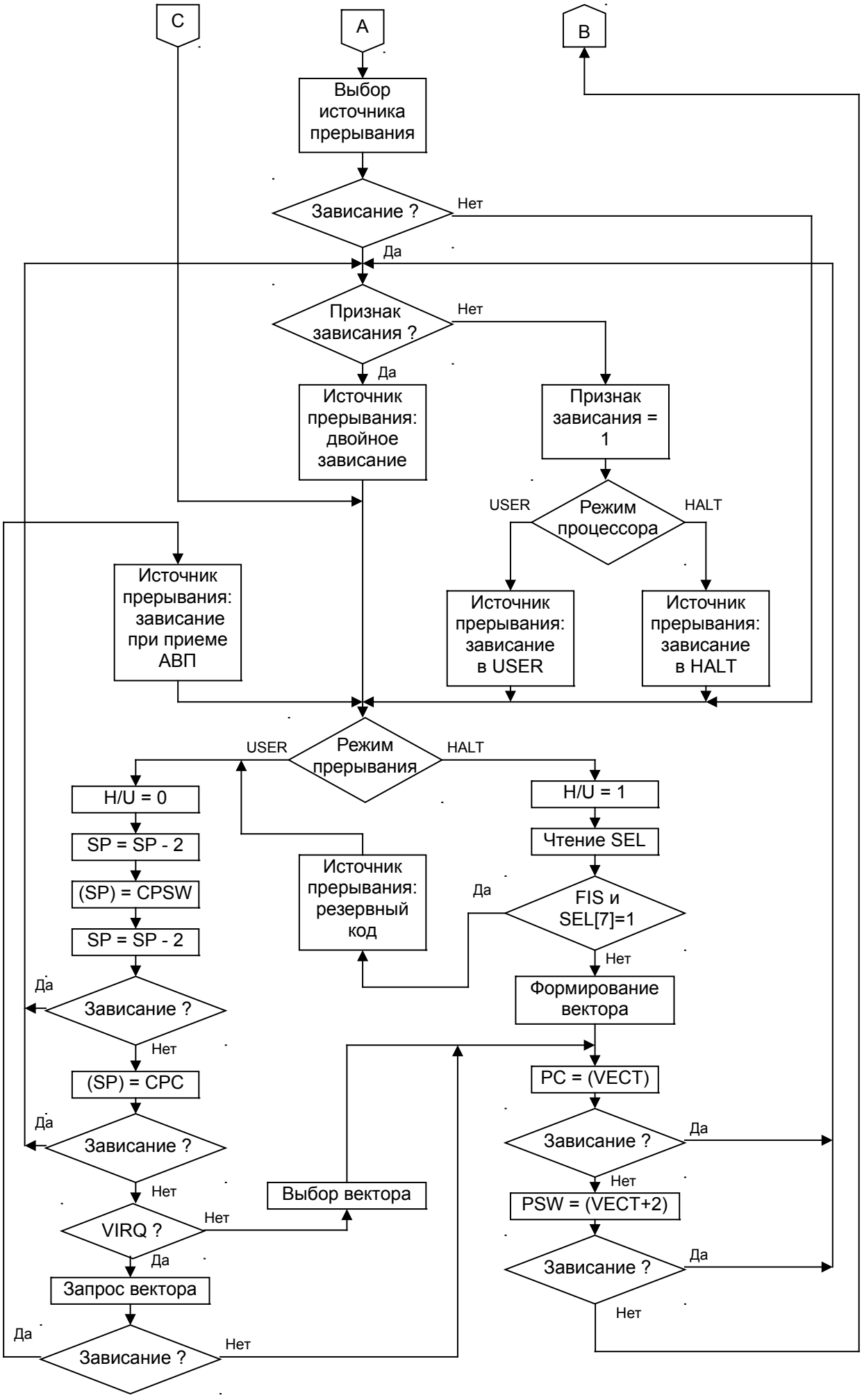
Переключение режимов работы процессора во время загрузки PSW:

Команда, прерывание	PSW
MTPS	7:5, 3:0 – загружаются 4 – не изменяется 8 – не изменяется
RTI, RTT	7:0 – загружаются 8 – если значение нового PC $\geq$ 160000, то загружается, иначе не изменяется
Прерывания режима USER, BPT, IOT, TRAP, EMT	7:0 – загружаются 8 – обнуляется
Прерывания режима HALT, команда HALT, START, STEP	8:0 – загружаются

Регистры копии счётчика команд CPC и копии слова состояния процессора CPSW повторяют значения счётчика команд PC и слова состояния процессора PSW, если какой-нибудь из битов 7 (P) или 8 (H/U) PSW находится в сброшенном состоянии (т.е. в режиме USER или в режиме HALT с разрешёнными прерываниями). При равенстве обеих битов P и H/U в PSW единице регистры копий блокируются, что даёт возможность установить их значение с помощью специальных команд. Регистр CPSW изменяется только тогда, когда происходит изменение регистра PSW, таким образом, в режиме HALT с разрешёнными прерываниями можно установить новое значение CPSW и он будет сохранять установленное значение до тех пор, пока не исполнится команда, изменяющая PSW (загрузка PSW, изменение признаков после исполнения арифметических и логических команд).

Алгоритм работы микропроцессора.




C
A
B
Выбор источника прерывания
Зависание ?
Нет
Да
Признак зависания ?
Нет
Да
Источник прерывания: двойное зависание
Признак зависания = 1
USER
Режим процессора
HALT
Источник прерывания: зависание при приеме АВП
Источник прерывания: зависание в USER
Источник прерывания: зависание в HALT
USER
Режим прерывания
HALT
H/U = 0
SP = SP - 2
(SP) = CPSW
SP = SP - 2
Да
Зависание ?
Нет
(SP) = CPC
Да
Зависание ?
Нет
VIRQ ?
Нет
Да
Запрос вектора
Да
Зависание ?
Нет
Источник прерывания: резервный код
H/U = 1
Чтение SEL
Да
FIS и SEL[7]=1
Нет
Формирование вектора
PC = (VECT)
Зависание ?
Да
Нет
PSW = (VECT+2)
Зависание ?
Да
Нет
Выбор вектора

На блок-схемах показан алгоритм функционирования МП при выполнении команд и обработке прерываний. Поведение процессора при запросе на прямой доступ к памяти здесь не показано, подробно об этом можно прочесть в заводской документации.

В своём составе микропроцессор включает триггеры режима ожидания и признака зависания. Режим ожидания позволяет процессору не выбирать команды из памяти и ожидать сигнала прерывания, после исполнения которого работа процессора может быть продолжена с точки останова. Признак зависания служит для определения двойного зависания, т.е. если при обработке микропрограммы зависания происходит зависание, то это переводит процессор в режим обработки фатальной ситуации.

После снятия сигналов DCLO и ACLO по положительному фронту ACLO микропроцессор формирует в качестве источника прерывания событие начального запуска системы и переходит на обработку этого прерывания. Если после загрузки вектора начального пуска отсутствуют незамаскированные прерывания, то процессор переходит к обработке команды, иначе производится обработка прерываний.

Обработка команды.

Если процессор находится не в режиме ожидания, то по значению счётчика команд производится выборка очередной команды из памяти. После выборки команды значение PC увеличивается на 2. Если во время выборки команды произошло зависание, то МП переходит на микропрограмму обработки прерываний, в противном случае очищается триггер признака зависания. Если в качестве прочитанных команд были команды программных прерываний (BPT, IOT, EMT, TRAP, HALT, группа команд FIS), то процессор формирует источник прерывания в зависимости от команды и переходит на микропрограмму исполнения прерываний. В другом случае, в зависимости от исполняемой команды и используемых в ней способов адресации, МП может прочитывать аргументы из памяти и записывать результат в память. Если во время исполнения команды зафиксировано зависание, то обработка команды прекращается и процессор переходит к обработке зависания. После успешной обработки команды производится переход к микропрограмме обработки прерываний, однако если в качестве исполняемых команд были команды STEP и RTT (если она производит установку разряда T в PSW), то запуск микропрограммы обработки прерываний не производится и микропроцессор переходит к обработке следующей команды.

Обработка прерываний.

Микропрограммой обработки прерываний сначала анализируются все незамаскированные прерывания и выбирается из них самое приоритетное. После выбора прерывания сбрасывается признак режима ожидания. В качестве источников прерываний используются:

1. Прерывание по зависанию. Подразделяется на:
    1.1. Зависание в режиме USER. Возникает при работе процессора в режиме USER, если в результате исполнения циклов ВВОД или ВЫВОД от устройства не поступил сигнал ответа RPLY. В качестве вектора используется значение 4_8. При его обработке устанавливается признак зависания.
    1.2. Зависание в режиме HALT. Возникает при работе процессора в режиме HALT, если в результате исполнения циклов ВВОД или ВЫВОД от устройства не поступил сигнал ответа RPLY. В качестве вектора используется значение $SEL004_8$. При его работе устанавливается признак зависания.
    1.3. Двойное зависание. Возникает при работе процессора, если в результате исполнения циклов ВВОД или ВЫВОД от устройства не поступил сигнал ответа RPLY и при этом был установлен признак зависания. Оно обычно возникает в момент возникновения зависания во время работы микропрограммы обработки зависания, а также, если при чтении первой команды из памяти после успешного завершения микропрограммы обработки прерывания произошло зависание. В качестве вектора используется значение $SEL174_8$.

2. Прерывание по запрещённому коду или резервной команде:
    2.1. Прерывание по запрещённому коду. Возникает, если в команде используется недопустимый способ адресации. Такими случаями являются использование в командах JMP и JSR регистрового способа адресации. В качестве вектора используется значение 4_8.
    2.2. Прерывание по резервному коду. Возникает в трех случаях:
        а) в системе команд отсутствует команда с заданным кодом;
        б) группа команд режима HALT (коды 10 – 37) исполняется в режиме USER;
        в) если при исполнении обработки команд группы FIS (коды 075000 – 075037) в безадресном регистре SEL 7-й бит находится в установленном состоянии.

    В качестве вектора используется значение 10_8.
3. Прерывание по Т-разряду в PSW. Возникает при установлении разряда Т в PSW. В качестве вектора используется 14_8. Прерывание маскируется установкой триггера режима ожидания при исполнении команды WAIT. Установка и очистка бита Т возможна только при выполнении команд RTT, RTI, STEP, START и при загрузке вектора прерывания. Если при загрузке вектора прерывания, исполнения команд START и RTI устанавливается бит Т, то первое прерывание произойдет до исполнения первой команды нового процесса. Если при загрузке нового вектора, исполнении команд START и STEP очищается бит Т, то прерывания не будет. При исполнении команд STEP и RTT устанавливается бит Т – прерывания нет, т.к. после этих команд не запускается микропрограмма обработки прерываний. Команды RTI и RTT очищают бит Т – после их исполнения происходит прерывание, но в стеке сохранится PSW с очищенным битом Т.
4. Отрицательный фронт ACLO. При отрицательном фронте производится установка триггера сигнала ACLO. Сброс этого триггера возможен только при удовлетворении запроса на прерывание. В качестве вектора используется 24_8. Прерывание маскируется одновременной установкой разрядов 7 (P) и 8 (H/U) в PSW.
5. Радиальное прерывание HALT. Прерывание возникает при активном низком уровне сигнала на входе HALT микропроцессора (вывод 29). В качестве вектора используется $SEL170_8$. Прерывание маскируется установкой разряда 8 (H/U) в PSW.
6. Отрицательный фронт EVNT. При отрицательном фронте производится установка триггера сигнала EVNT. Сброс этого триггера возможен при удовлетворении запроса на прерывание или по команде RESET. В качестве вектора используется 100_8. Прерывание маскируется установкой разряда 7 (P) в PSW.
7. Векторное прерывание VIRQ. Прерывание возникает при активном низком уровне сигнала на входе VIRQ микропроцессора (вывод 28). Вектор прерывания передаётся устройством по процедуре приема вектора прерывания. Прерывание маскируется установкой разряда 7 (P) в PSW.

После выбора источника анализируется, обрабатывается в данный момент зависание или нет. При обработке зависания детализируется, какой тип зависания обрабатывается. Если уже установлен триггер признака обработки зависания, то в качестве источника прерывания выбирается двойное зависание. Если же признак зависания не установлен, то производится установка этого признака, а в качестве источника выбирается либо зависание в режиме USER, либо зависание в режиме HALT, в зависимости от режима работы процессора.

Дальнейшим шагом является проверка типа прерывания. Все прерывания делятся на два типа: прерывания, обрабатываемые в режиме USER и прерывания, обрабатываемые в режиме HALT, или более просто: прерывания USER и прерывания HALT. Каждый тип прерываний принудительно переключает процессор в тот режим, в котором он производит свою обработку.

Прерывания USER переключают МП в режим USER сбросом бита 8 (H/U) в PSW. Это необходимо, т.к. сохранение состояния текущего процесса производится в стеке, вектора прерываний расположены в адресном пространстве USER и обработка этих прерываний производится всегда в режиме USER. Далее в стеке последовательно сохраняются значения копий слова состояния процессора CPSW и счётчика команд CPC. Если в процессе сохранения происходит зависание, то процессор прерывает обработку прерывания и переходит сразу к микропрограмме обработки зависания. Следует заметить, что если происходит зависание при сохранении состояния текущего процесса, то значение регистра R6(SP) всегда уменьшается на 4. Если производится обработка прерывания VIRQ, то МП по процедуре приема вектора прерывания

запрашивает у внешнего устройства вектор. При зависании при приёме вектора, процессор прерывается по зависанию при приёме адреса вектора прерывания со значением $SEL274_8$. При обработке других источников (командные прерывания (BPT, IOT, EMT, TRAP), зависание в USER, запрещённый код, резервный код, T-разряд, ACLO, EVNT) вектор прерывания выбирается из блока констант операционного блока микропроцессора. На этом шаге и происходит сброс триггеров прерываний ACLO и EVNT при их обработке.

Прерывания HALT переключают МП в режим HALT установкой бита 8 (H/U) в PSW. Это необходимо, т.к. загрузка вектора производится из области памяти HALT. Дальнейший режим работы процессора подпрограммы обработки прерывания определяется загруженным PSW из вектора. Для определения значения вектора МП считывает безадресный регистр SEL. Старший байт вектора равен старшему байту регистра SEL, а младший считывается из блока констант операционного блока. Если производится обработка группы команд FIS и в считанном регистре SEL установлен бит 7, то производится переход на обработку по коду резервной команды, в ином случае формируется вектор $SEL010_8$. При обработке прерываний HALT явного сохранения состояния текущего процесса не производится. Если прерываемый процесс исполнялся в режиме USER или в режиме HALT с разрешёнными прерываниями, а новый процесс будет исполняться в режиме HALT с запрещёнными прерываниями, то значения PC и PSW прерываемого процесса можно прочесть из копий CPC и CPSW, в ином случае значения PC и PSW прерываемого процесса узнать невозможно.

Сформированный вектор прерывания указывает на пару смежных 16-разрядных ячеек в памяти, которые содержат новое значение PC и PSW прерываемого процесса. Если обрабатывается прерывание USER, то загрузить 8-й бит PSW невозможно, он останется в сброшенном состоянии. При обработке прерываний HALT загружаются все 9 битов.

После загрузки новых значений PC и PSW микропрограмма обработки прерываний переходит на своё начало и анализирует источники незамаскированных прерываний с учетом новых значений битов H/U(8) и P(7) в PSW. При отсутствии незамаскированных прерываний осуществляется переход на обработку команды.

Сводная таблица прерываний микропроцессора:

Источник	Вектор (восьмер.)	Режим	Приоритет	Хранение состояния предыдущего процесса
Зависание в USER	004	USER	1	Стек
Запрещённый код	004	USER	2	Стек
Резервный код	010	USER	2	Стек
T-разряд	014	USER	3	Стек
Команда BPT	014	USER	-	Стек
Команда IOT	020	USER	-	Стек
ACLO	024	USER	4	Стек
Команда EMT	030	USER	-	Стек
Команда TRAP	034	USER	-	Стек
EVNT	100	USER	6	Стек
VIRQ	Передаётся	USER	7	Стек
Начальный пуск	SEL000	HALT	-	-
Зависание в HALT	SEL004	HALT	1	CPC, CPSW
Команды FIS	SEL010	HALT	-	CPC, CPSW
Сигнал HALT	SEL170	HALT	5	CPC, CPSW
Команда HALT	SEL170	HALT	-	CPC, CPSW
Двойное зависание	SEL174	HALT	1	CPC, CPSW
Зависание при приёме АВП	SEL274	HALT	-	CPC, CPSW

Примечание: процессор не обрезает старшие биты АВП по VIRQ.

Использование очистки разряда разрешения прерывания в регистре состояния устройства может вызвать зависание при приеме АВП, если при выполнении этой команды произошло прерывание от этого устройства. Во избежании этой ситуации необходимо очистку разряда разрешения прерывания производить при установленном разряде 7 (P) в PSW (при замаскированном VIRQ):

```
MTPS #200
CLR  @#CSR
MTPS #0
```

Описание системы команд микропроцессора.

Форматы команд.

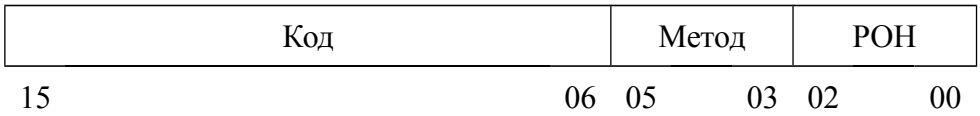
В микропроцессоре используются три типа команд: безадресные, одноадресные и двухадресные.

В безадресных командах код команды содержит только код операции.

В кодах одноадресных и двухадресных команд обычно содержится информация, которая представляет:

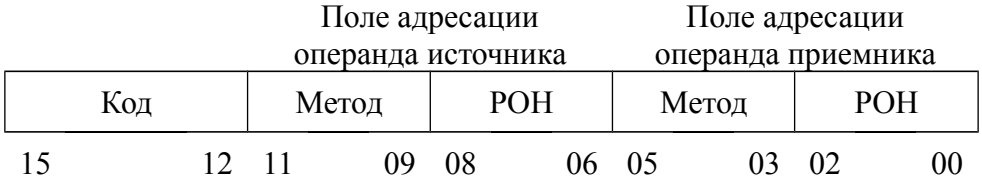
- выполняемую функцию (код операции);
- метод адресации;
- регистры общего назначения, используемые при выборе операндов.

Формат одноадресных команд:



При исполнении одноадресных команд рассчитывается адрес операнда, производится его чтение из памяти, изменение в операционном блоке МП, и запись в память по вычисленному заранее адресу. После выполнения команды изменяются признаки N, Z, V и C в PSW. Необходимость чтения из памяти и запись в память определяются используемой командой и применяемым методом адресации. Если во время выполнения команды происходит зависание, то выполнение команды прекращается и признаки в PSW не изменяются.

Формат двухадресных команд:



При исполнении двухадресных команд сперва рассчитывается адрес операнда источника, производится его чтение из памяти. Затем рассчитывается адрес операнда приёмника и производится его чтение из памяти. Далее в операционном блоке микропроцессора производится операция между операндами, которая определяется используемым кодом операции. После выполнения операции результат записывается в память по адресу операнда приёмника. Также как и в случае с одноадресными командами необходимость чтения из памяти и запись в память операндов определяются используемым кодом операции и методами адресации. Если во время вычисления адреса источника или его чтения из памяти происходит зависание, то команда прерывается и вычисления адреса приёмника не производится, также как и запись операнда приёмника в память. Признаки N, Z, V и C устанавливаются только после успешного завершения операции.

Методы адресации.

В микропроцессоре используются восемь методов адресации:

Код	Наименование	Мнемоника	Описание
0	Регистровый	Rn	Регистр содержит операнд
1	Косвенно-регистровый	@Rn или (Rn)	Регистр содержит адрес операнда
2	Автоинкрементный	(Rn)+	Регистр содержит адрес операнда, который после выборки увеличивается на 2 для команд с операциями над полными словами и на 1 для команд с операциями над байтами
3	Косвенно-автоинкрементный	@(Rn)+	Регистр содержит адрес ячейки, в которой хранится адрес операнда. После выборки содержимое регистра увеличивается на 2.
4	Автодекрементный	-(Rn)	Содержимое регистра автоматически уменьшается на 2 для команд с операциями над полными словами и на единицу для команд с операциями над байтами. Затем содержимое регистра используется как адрес операнда.
5	Косвенно-автодекрементный	@-(Rn)	Содержимое регистра уменьшается на 2 и затем используется как адрес ячейки, в которой хранится адрес операнда.
6	Индексный	X(Rn)	Содержимое регистра складывается с индексным словом, которое следует за командой и адресуется счётчиком команд PC(R7), и их сумма используется как адрес операнда. После чтения индексного слова значение счётчика команд увеличивается на 2.
7	Косвенно-индексный	@X(Rn)	Содержимое регистра складывается с индексным словом, которое следует за командой и адресуется счетчиком команд PC(R7), и их сумма используется как адрес ячейки в которой хранится адрес операнда. После чтения индексного слова значение счётчика команд увеличивается на 2.

Если в команде при автодекрементном или автоинкрементном способах адресации применяются регистры SP(R6) или PC(R7), то они всегда уменьшаются или увеличиваются на 2, независимо от того, работает выполняемая команда с полными словами или с байтами.

Содержимое регистров всегда увеличивается или уменьшается при использовании методов адресации 2, 3, 4 и 5, даже если при вычислении адреса операнда или чтении операнда из памяти произошло зависание.

Регистр PC(R7) может использоваться при всех способах адресации, но наиболее эффективно он используется с четырьмя методами:

Код	Наименование	Мнемоника	Описание
2	Непосредственный	#X	Операнд следует за командой. После выборки операнда значение PC увеличивается на 2.

Код	Наименование	Мнемоника	Описание
3	Абсолютный	@#X	Адрес операнда следует за командой. После выборки значение PC увеличивается на 2.
6	Относительный	X	Индексное слово, которое следует за командой складывается с содержимым PC, который указывает на ячейку памяти, следующую за индексным словом, и их сумма используется как адрес операнда.
7	Косвенно-относительный	@X	Индексное слово, которое следует за командой складывается с содержимым PC, который указывает на ячейку памяти, следующую за индексным словом, и их сумма используется как адрес ячейки, которая содержит адрес операнда.

Описание команд.

Описание каждой команды включает: мнемонику, восьмеричный код, формат команды в двоичном коде, алгоритм и описание выполнения команды и выработки признаков.

При описании команд используются следующие обозначения:

R – регистр общего назначения;
PC – счётчик команд R7;
SP – указатель стека R6;
PSW – слово состояния процессора;
SS – поле адресации операнда источника (включает метод и регистр);
src – адрес источника;
(src) – операнд источника;
DD – поле адресации операнда приёмника (включает метод и регистр);
dst – адрес приёмника;
(dst) – операнд приёмника;
XXX – смещение (8 разрядов);
NN – смещение (6 разрядов);
( ) – содержимое ячейки;
and – логическое умножение (И);
or – логическое сложение (ИЛИ);
xor – исключающее ИЛИ;
not – логическое отрицание (инверсия);
:= – становится равным;
–(SP) – записать в стек;
(SP)+ – выбрать из стека;
B – байтовая команда.

Одноадресные команды.

CLR 0050DD (005000–005077)
CLRB 1050DD (105000–105077)

CLEAR / ОЧИСТКА

0 / 1	0 0 0	1 0 1	0 0 0	d d d	d d d

Алгоритм: (dst) := 0
Описание: В указанную ячейку или байт записываются нули.
Признаки: *N* – очищается; *Z* – устанавливается; *V* – очищается; *C* – очищается.

COM 0051DD (005100–005177)
COMB 1051DD (105100–105177)

COMPLEMENT / ИНВЕРТИРОВАНИЕ

0 / 1	0 0 0	1 0 1	0 0 1	d d d	d d d

Алгоритм: (dst) := not (dst)
Описание: Заменяет содержимое указанной ячейки или байта его двоичным обратным кодом (каждый разряд, содержащий «0», устанавливается, а каждый разряд, содержащий «1», очищается).
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – очищается; *C* – устанавливается.

INC 0052DD (005200–005277)
INCB 1052DD (105200–105277)

INCREMENT / ПРИБАВЛЕНИЕ ЕДИНИЦЫ

0 / 1	0 0 0	1 0 1	0 1 0	d d d	d d d

Алгоритм: (dst) := (dst) + 1
Описание: Прибавляет единицу к содержимому указанной ячейки или байта.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается, если операнд равен 077777_8 (177_8 для байтовых команд), в противном случае очищается; *C* – не изменяется.

DEC 0053DD (005300–005377)
DECB 1053DD (105300–105377)

DECREMENT / ВЫЧИТАНИЕ ЕДИНИЦЫ

0 / 1	0 0 0	1 0 1	0 1 1	d d d	d d d

Алгоритм: (dst) := (dst) – 1
Описание: Из содержимого указанной ячейки или байта вычитается единица.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается, если операнд равен 100000_8 (200_8 для байтовых команд), в противном случае очищается; *C* – не изменяется.

NEG 0054DD (005400–005477)
NEGB 1054DD (105400–105477)

NEGATE / ИЗМЕНЕНИЕ ЗНАКА

0 / 1	0 0 0	1 0 1	1 0 0	d d d	d d d

Алгоритм: (dst) := – (dst)
Описание: Содержимое указанной ячейки или байта заменяется двоичным дополнением операнда. Следует заметить, что число 100000_8 (200_8 для байтовых команд) заменяется самим собой, так как не существует соответствующего ему положительного числа.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается, если результат равен 100000_8 (200_8 для байтовых команд), в противном случае очищается; *C* – очищается, если результат равен нулю, в противном случае устанавливается.

TST 0057DD (005700–005777)
TSTB 1057DD (105700–105777)

TEST / ПРОВЕРКА

0 / 1	0 0 0	1 0 1	1 1 1	d d d	d d d

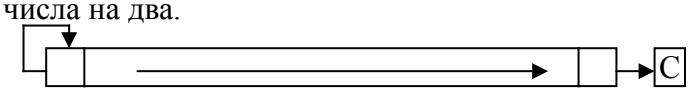
Алгоритм: (dst)
Описание: В зависимости от содержимого указанной ячейки или байта устанавливаются или очищаются признаки *N* и *Z*. Содержимое ячейки не изменяется.
Признаки: *N* – устанавливается, если операнд меньше нуля, в противном случае очищается; *Z* – устанавливается, если операнд равен нулю, в противном случае очищается; *V* – очищается; *C* – очищается.

ASR 0062DD (006200–006277)
ASRB 1062DD (106200–106277)

ARITHMETIC SHIFT RIGHT / АРИФМЕТИЧЕСКИЙ СДВИГ ВПРАВО

0 / 1	0 0 0	1 1 0	0 1 0	d d d	d d d

Алгоритм: (dst) := (dst) >> 1.
Описание: Операнд (слово или байт) сдвигается на одну позицию вправо. Содержимое старшего (знакового) разряда не изменяется (копируется сам в себя), разряд *C* загружается содержимым младшего выдвинутого разряда. Следовательно команды ASR и ASRB выполняют деление числа на два.



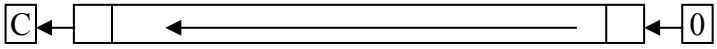
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается результатом операции исключающее ИЛИ над содержимым разрядов *N* и *C*, которое они имеют после операции сдвига; *C* – загружается содержимым младшего выдвинутого разряда.

ASL 0063DD (006300–006377)
ASLB 1063DD (106300–106377)

ARITHMETIC SHIFT LEFT / АРИФМЕТИЧЕСКИЙ СДВИГ ВЛЕВО

0 / 1	0 0 0	1 1 0	0 1 1	d d d	d d d

Алгоритм: (dst) := (dst) << 1.
Описание: Операнд (слово или байт) сдвигается на одну позицию влево. Содержимое старшего (знакового) выдвинутого разряда загружается в разряд *C*, в младший разряд задвигается нуль. Следовательно команды ASL и ASLB выполняют умножение числа на два.



Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается результатом операции исключающее ИЛИ над содержимым разрядов *N* и *C*, которое они имеют после операции сдвига; *C* – загружается содержимым старшего выдвинутого разряда.

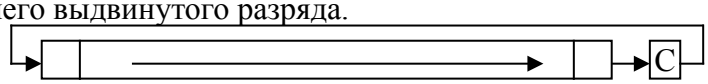
ROR 0060DD (006000–006077)
RORB 1060DD (106000–106077)

ROTATE RIGHT / ЦИКЛИЧЕСКИЙ СДВИГ ВПРАВО

0 / 1	0 0 0	1 1 0	0 0 0	d d d	d d d

Алгоритм: (dst) := (dst) >>> 1.

Описание: Операнд (слово или байт) сдвигается на одну позицию вправо. Содержимое старшего (знакового) разряда загружается содержимым разряда *C*, разряд *C* загружается содержимым младшего выдвинутого разряда.



Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается результатом операции исключающее ИЛИ над содержимым разрядов *N* и *C*, которое они имеют после операции сдвига; *C* – загружается содержимым младшего выдвинутого разряда.

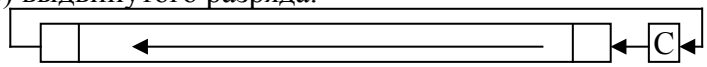
ROL 0061DD (006100–006177)
ROLB 1061DD (106100–106177)

ROTATE LEFT / ЦИКЛИЧЕСКИЙ СДВИГ ВЛЕВО

0 / 1	0 0 0	1 1 0	0 0 1	d d d	d d d

Алгоритм: (dst) := (dst) <<< 1.

Описание: Операнд (слово или байт) сдвигается на одну позицию влево. Содержимое младшего разряда загружается содержимым разряда *C*, разряд *C* загружается содержимым старшего (знакового) выдвинутого разряда.



Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается результатом операции исключающее ИЛИ над содержимым разрядов *N* и *C*, которое они имеют после операции сдвига; *C* – загружается содержимым младшего выдвинутого разряда.

ADC 0055DD (005500–005577)
ADCB 1055DD (105500–105577)

ADD CARRY / ПРИБАВЛЕНИЕ ПЕРЕНОСА

0 / 1	0 0 0	1 0 1	1 0 1	d d d	d d d

Алгоритм: (dst) := (dst) + *C*.

Описание: Операнд (слово или байт) складывается с содержимым разряда *C*. Эта команда даёт возможность оперировать со значениями, превышающими по размеру 16 бит, прибавляя перенос к старшей части.

Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается, если перед выполнением операции операнд был равен 077777_8 (177_8 для байтовых команд) и разряд *C* был установлен, в противном случае очищается; *C* – устанавливается, если перед выполнением операции операнд был равен 177777_8 (377_8 для байтовых команд) и разряд *C* был установлен, в противном случае очищается.

SBC 0056DD (005600–005677)
SBCB 1056DD (105600–105677)

SUBTRACT CARRY / ВЫЧИТАНИЕ ПЕРЕНОСА

0 / 1	0 0 0	1 0 1	1 1 0	d d d	d d d

Алгоритм: (dst) := (dst) – *C*.
Описание: Вычитает содержимое разряда *C* из операнда (слова или байта). Эта команда даёт возможность оперировать со значениями, превышающими по размеру 16 бит, вычитая перенос из старшей части.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается, если перед выполнением операции операнд был равен 100000_8 (200_8 для байтовых команд) и разряд *C* был установлен, в противном случае очищается; *C* – устанавливается, если перед выполнением операции операнд был равен нулю и разряд *C* был установлен, в противном случае очищается.

SXT 0067DD (006700–006777)

SIGN EXTEND / РАСШИРЕНИЕ ЗНАКА

0	0 0 0	1 1 0	1 1 1	d d d	d d d

Алгоритм: (dst) := (*N* == 1) ? 177777_8 , 0.
Описание: Если *N* равен единице, то операнд заменяется значением 177777_8, в противном случае заменяется нулем. Эта команда даёт возможность оперировать со значениями, превышающими по размеру 16 бит, расширяя знаковую часть.
Признаки: *N* – не изменяется; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – очищается; *C* – не изменяется.

SWAB 0003DD (000300–000377)

SWAP BYTES / ПЕРЕСТАНОВКА БАЙТОВ

0	0 0 0	0 0 0	0 1 1	d d d	d d d

Алгоритм: Byte 1 / Byte 0 := Byte 0 / Byte 1.
Описание: В операнде меняются местами старший и младший байт. Адресация всегда происходит по полному слову.
Признаки: *N* – устанавливается, если старший разряд младшего байта результата установлен (разряд 7), в противном случае очищается; *Z* – устанавливается, если младший байт результата равен нулю, в противном случае очищается; *V* – очищается; *C* – очищается.

MFPS 1067DD (106700–106777)

MOVE BYTE FROM PROCESSOR STATUS WORD / ЧТЕНИЕ PSW

1	0 0 0	1 1 0	1 1 1	d d d	d d d

Алгоритм: (dst) := (byte) PSW[7:0].
Описание: Восемь разрядов слова состояния процессора PSW (разряды 7–0) пересылаются в указанный операнд. При выполнении команды всегда используется адресация по байту. Если в качестве приёмника используется регистр общего назначения, то в указанном регистре производится расширение знака (разряды с 15 по 08 становятся равными значению разряда 07). Так как эта команда работает только с байтами, то содержимое разряда H/U (8) в PSW с помощью этой команды узнать невозможно.
Признаки: *N* – устанавливается, если в разряде 07 PSW была единица, в противном случае очищается; *Z* – устанавливается, если восемь разрядов PSW (с 07 по 00) были равны нулю, в противном случае очищается; *V* – очищается; *C* – не изменяется.

MTPS 1064SS (106400–106477)

MOVE BYTE TO PROCESSOR STATUS WORD / ЗАПИСЬ PSW

1	0 0 0	1 1 0	1 0 0	s s s	s s s

Алгоритм: PSW[7:5,3:0] := (byte) (src)[7:5,3:0].
Описание: Семь разрядов указанного операнда замещают значение слова состояния процессора PSW (разряды 7–5 и 3–0). При выполнении команды всегда используется адресация по байту. Изменить содержимое разряда T (4) в PSW с помощью этой команды невозможно. Так как эта команда работает только с байтами, то с помощью этой команды невозможно изменить разряд H/U (8) в PSW. Обычно эта команда используется для изменения разряда P (7) в PSW, установкой которого можно маскировать прерывания EVNT и VIRQ.
Признаки: *N*, *Z*, *V*, *C* – устанавливаются или очищаются в соответствии с разрядами 03–00 операнда источника (src).

Двухадресные команды.

MOV 01SSDD (010000–017777)
MOVB 11SSDD (110000–117777)

MOVE SOURCE TO DESTINATION / ПЕРЕСЫЛКА

0 / 1	0 0 1	s s s	s s s	d d d	d d d

Алгоритм: (dst) := (src)
Описание: Операнд источника (src) пересылается по адресу операнда приёмника. Прежнее значение ячейки dst теряется. Содержимое ячейки src не изменяется. При операциях с байтами команда MOVB, если приёмником является регистр общего назначения (единственная среди байтовых команд), расширяет старший байт указанного регистра значением знакового разряда младшего байта (разряды с 15 по 08 становятся равными разряду 07). В других случаях MOVB оперирует с байтами так же, как MOV со словами.
Признаки: *N* – устанавливается, если (src) меньше нуля, в противном случае очищается; *Z* – устанавливается, если (src) равен нулю, в противном случае очищается; *V* – очищается; *C* – не изменяется.

CMP 02SSDD (020000–027777)
CMPB 12SSDD (120000–127777)

COMPARE SOURCE TO DESTINATION / СРАВНЕНИЕ

0 / 1	0 1 0	s s s	s s s	d d d	d d d

Алгоритм: (src) – (dst).
Описание: Сравнивает операнды источника и приёмника и изменяет признаки, которые потом могут быть использованы для команд условных переходов. Оба операнда не изменяются. В отличии от команды вычитания порядок действий следующий: (src) – (dst), а не (dst) – (src).
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается, если было арифметическое переполнение, в противном случае очищается. Арифметическое переполнение может произойти в том случае, когда операнды были противоположного знака и знак результата совпадает со знаком операнда приёмника (dst). Новое значение *V* можно выразить формулой *V* := (sign(src) xor sign(dst)) and not(sign(res) xor sign(dst)); *C* – очищается, если был перенос из старшего разряда результата, в противном случае устанавливается. Перенос может произойти в том случае, если операнд источника был больше или равен нулю, а операнд приёмника – меньше нуля, или если

операнды источника и приёмника были одного знака, а результат получился меньше нуля. Новое значение *C* можно выразить формулой *C* := (not sign(src) and sign(dst)) or (not(sign(src) xor sign(dst)) and sign(res)).

ADD 06SSDD (060000–067777)

ADD SOURCE TO DESTINATION / СЛОЖЕНИЕ

0	1 1 0	s s s	s s s	d d d	d d d

Алгоритм: (dst) := (src) + (dst).
Описание: Операнды источника (src) складывается с операндом приёмника (dst) и результат записывается по адресу операнда приёмника. Первоначальное значение (dst) теряется. Содержимое (src) не изменяется. Сложение выполняется в двоичном дополнительном коде.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается, если было арифметическое переполнение, в противном случае очищается. Арифметическое переполнение может произойти в том случае, когда операнды были одного знака, а результат получился противоположного знака. Новое значение *V* можно выразить формулой *V* := not(sign(src) xor sign(dst)) and (sign(res) xor sign(dst)); *C* – устанавливается, если был перенос из старшего разряда результата, в противном случае очищается. Перенос может произойти в том случае, если операнды источника и приёмника были меньше нуля, или если операнды источника и приёмника были противоположного знака, а результат получился больше или равен нулю. Новое значение *C* можно выразить формулой *C* := (sign(src) and sign(dst)) or ((sign(src) xor sign(dst)) and not sign(res)).

SUB 16SSDD (160000–167777)

SUBTRACT SOURCE FROM DESTINATION / ВЫЧИТАНИЕ

1	1 1 0	s s s	s s s	d d d	d d d

Алгоритм: (dst) := (dst) – (src).
Описание: Из операнда приёмника (dst) вычитается операнд источника (src) и результат записывается по адресу операнда приёмника. Первоначальное значение (dst) теряется. Содержимое (src) не изменяется. При арифметических операциях с удвоенной точностью установка разряда *C* означает заём единицы из старшей части вычитаемого.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается; *Z* – устанавливается, если результат равен нулю, в противном случае очищается; *V* – устанавливается, если было арифметическое переполнение, в противном случае очищается. Арифметическое переполнение может произойти в том случае, когда операнды были противоположного знака, а результат получился одного знака с операндом источника. Новое значение *V* можно выразить формулой *V* := (sign(src) xor sign(dst)) and not(sign(res) xor sign(src)); *C* – очищается, если был перенос из старшего разряда результата, в противном случае устанавливается. Перенос может произойти в том случае, если операнд источника был меньше нуля и операнд приёмника был больше или равен нулю, или если операнды источника и приёмника были одного знака, а результат получился меньше нуля. Новое значение *C* можно выразить формулой *C* := (sign(src) and not sign(dst)) or (not(sign(src) xor sign(dst)) and sign(res)).

BIT 03SSDD (030000–037777)
BITB 13SSDD (130000–137777)

BIT TEST / ПРОВЕРКА РАЗРЯДОВ

0 / 1	0 1 1	s s s	s s s	d d d	d d d

Алгоритм: (src) and (dst)
Описание: Выполняется логическая функция И над операндами источника (src) и приёмника (dst), изменяя соответствующим образом признаки. Оба операнда не изменяют своего значения. Команда BIT/BITB используется для проверки состояния разрядов операнда (src), для которых установлены соответствующие разряды в операнде (dst).
Признаки: *N* – устанавливается, если старший разряд результата установлен, в противном случае очищается; *Z* – устанавливается, если все разряды результата равны нулю, в противном случае очищается; *V* – очищается; *C* – не изменяется.

BIC 04SSDD (040000–047777)
BICB 14SSDD (140000–147777)

BIT CLEAR / ОЧИСТКА РАЗРЯДОВ

0 / 1	1 0 0	s s s	s s s	d d d	d d d

Алгоритм: (dst) := not(src) and (dst)
Описание: Очищается каждый разряд операнда (dst), соответствующий установленному разряду операнда (src). Первоначальное значение (dst) теряется. Содержимое (src) не изменяется.
Признаки: *N* – устанавливается, если старший разряд результата установлен, в противном случае очищается; *Z* – устанавливается, если все разряды результата равны нулю, в противном случае очищается; *V* – очищается; *C* – не изменяется.

BIS 05SSDD (050000–057777)
BISB 15SSDD (150000–157777)

BIT SET / УСТАНОВКА РАЗРЯДОВ

0 / 1	1 0 1	s s s	s s s	d d d	d d d

Алгоритм: (dst) := (src) or (dst)
Описание: Выполняет логическую операцию ИЛИ между операндами источника и приёмника и заносит результат по адресу приёмника dst. Разряды операнда (dst) устанавливаются в состояние «1», если соответствующие разряды операнда (src) находятся в состоянии «1». Первоначальное значение (dst) теряется. Содержимое (src) не изменяется.
Признаки: *N* – устанавливается, если старший разряд результата установлен, в противном случае очищается; *Z* – устанавливается, если все разряды результата равны нулю, в противном случае очищается; *V* – очищается; *C* – не изменяется.

XOR 074RDD (074000–074777)

EXCLUSIVE OR / ИСКЛЮЧАЮЩЕЕ ИЛИ

0	1 1 1	1 0 0	R R R	d d d	d d d

Алгоритм: (dst) := R xor (dst)
Описание: Выполняется операция ИСКЛЮЧАЮЩЕЕ ИЛИ между содержимым указанного регистра и операнда (dst). Результат записывается в (dst). Содержимое регистра R не изменяется.
Признаки: *N* – устанавливается, если старший разряд результата установлен, в противном случае очищается; *Z* – устанавливается, если все разряды результата равны нулю, в противном случае очищается; *V* – очищается; *C* – не изменяется.

Команды ветвления.

Эти команды вызывают ветвление по адресу, являющемуся суммой смещения (умноженному на 2) и текущего содержимого PC (равного адресу команды ветвления плюс 2), если условие ветвления выполняется.

Смещение показывает, на сколько ячеек нужно перейти относительно текущего содержимого PC в ту или другую сторону. Так как слова имеют чётные адреса, то для получения истинного исполнительного адреса смещения необходимо умножить его на 2 перед прибавлением к PC, который всегда указывает на слово. Старший разряд смещения (разряд 07) является знаковым разрядом. Если он установлен, смещение отрицательное, ветвление происходит в сторону уменьшения адреса (в обратном направлении). Если в разряде 07 содержится 0, смещение положительное и ветвление происходит в сторону увеличения адреса (в прямом направлении).

Восьмиразрядное смещение позволяет производить ветвление в обратном направлении максимально на 200_8 слов от слова, на которое указывает текущее содержимое PC, и на 177_8 слов в прямом направлении.

BR 000400+XXX (000400–000777)

BRANCH (UNCONDITIONAL) / ВЕТВЛЕНИЕ БЕЗУСЛОВНОЕ

0	0 0 0	0 0 0	1 X X	X X X	X X X

Алгоритм: PC := PC + 2*XXX

Описание: Передача управления по адресу, который определяется состоянием знакового разряда 07 и значением смещения.

Признаки: не изменяются.

BNE 001000+XXX (001000–001377)

BRANCH IF NOT EQUAL (TO ZERO) / ВЕТВЛЕНИЕ, ЕСЛИ НЕ РАВНО (НУЛЮ)

0	0 0 0	0 0 1	0 X X	X X X	X X X

Алгоритм: IF (*Z*==0) THEN PC := PC + 2*XXX

Описание: Проверяется состояние разряда *Z* PSW и вызывает ветвление, если он очищен.

Признаки: не изменяются.

BEQ 001400+XXX (001400–001777)

BRANCH IF EQUAL (TO ZERO) / ВЕТВЛЕНИЕ, ЕСЛИ РАВНО (НУЛЮ)

0	0 0 0	0 0 1	1 X X	X X X	X X X

Алгоритм: IF (*Z*==1) THEN PC := PC + 2*XXX

Описание: Проверяется состояние разряда *Z* PSW и вызывает ветвление, если он установлен.

Признаки: не изменяются.

BPL 100000+XXX (100000–100377)

BRANCH IF PLUS / ВЕТВЛЕНИЕ, ЕСЛИ ПЛЮС

1	0 0 0	0 0 0	0 X X	X X X	X X X

Алгоритм: IF (*N*==0) THEN PC := PC + 2*XXX

Описание: Проверяется состояние разряда *N* PSW и вызывает ветвление, если он очищен.

Признаки: не изменяются.

BMI 100400+XXX (100400–100777)

BRANCH IF MINUS / ВЕТВЛЕНИЕ, ЕСЛИ МИНУС

1	0 0 0	0 0 0	1 X X	X X X	X X X

Алгоритм: IF (*N*==1) THEN PC := PC + 2*XXX
Описание: Проверяется состояние разряда *N* PSW и вызывает ветвление, если он установлен.
Признаки: не изменяются.

BVC 102000+XXX (102000–102377)

BRANCH IF OVERFLOW IS CLEAR / ВЕТВЛЕНИЕ, ЕСЛИ НЕТ ПЕРЕПОЛНЕНИЯ

1	0 0 0	0 1 0	0 X X	X X X	X X X

Алгоритм: IF (*V*==0) THEN PC := PC + 2*XXX
Описание: Проверяется состояние разряда *V* PSW и вызывает ветвление, если он очищен.
Признаки: не изменяются.

BVS 102400+XXX (102400–102777)

BRANCH IF OVERFLOW IS SET / ВЕТВЛЕНИЕ, ЕСЛИ ЕСТЬ ПЕРЕПОЛНЕНИЕ

1	0 0 0	0 1 0	1 X X	X X X	X X X

Алгоритм: IF (*V*==1) THEN PC := PC + 2*XXX
Описание: Проверяется состояние разряда *V* PSW и вызывает ветвление, если он установлен.
Признаки: не изменяются.

BCC 103000+XXX (103000–103377)
BHIS

BRANCH IF CARRY IS CLEAR / ВЕТВЛЕНИЕ, ЕСЛИ НЕТ ПЕРЕНОСА
BRANCH IF HIGHER OR SAME / ВЕТВЛЕНИЕ, ЕСЛИ БОЛЬШЕ ИЛИ РАВЕН

1	0 0 0	0 1 1	0 X X	X X X	X X X

Алгоритм: IF (*C*==0) THEN PC := PC + 2*XXX
Описание: Проверяется состояние разряда *C* PSW и вызывает ветвление, если он очищен. Это происходит в операции беззнакового сравнения, когда операнд источника больше или равен операнду приёмника.
Признаки: не изменяются.

BCS 103400+XXX (103400–103777)
BLO

BRANCH IF CARRY IS SET / ВЕТВЛЕНИЕ, ЕСЛИ ЕСТЬ ПЕРЕНОС
BRANCH IF LOWER / ВЕТВЛЕНИЕ, ЕСЛИ МЕНЬШЕ

1	0 0 0	0 1 1	1 X X	X X X	X X X

Алгоритм: IF (*C*==1) THEN PC := PC + 2*XXX
Описание: Проверяется состояние разряда *C* PSW и вызывает ветвление, если он установлен. Это происходит в операции беззнакового сравнения, когда операнд источника меньше операнда приёмника.
Признаки: не изменяются.

BGE 002000+XXX (002000–002377)

BRANCH IF GREATER THAN OR EQUAL (TO ZERO) / ВЕТВЛЕНИЕ, ЕСЛИ БОЛЬШЕ ИЛИ РАВНО (НУЛЮ)

0	0 0 0	0 1 0	0 X X	X X X	X X X

Алгоритм: IF ((*N* xor *V*)==0) THEN PC := PC + 2*XXX
Описание: Вызывается ветвление, если оба разряда признаков *N* и *V* установлены или очищены. Это происходит в операции знакового сравнения, когда операнд источника больше или равен операнду приёмника.
Признаки: не изменяются.

BLT 002400+XXX (002400–002777)

BRANCH IF LESS THAN (ZERO) / ВЕТВЛЕНИЕ, ЕСЛИ МЕНЬШЕ (НУЛЯ)

0	0 0 0	0 1 0	1 X X	X X X	X X X

Алгоритм: IF ((*N* xor *V*)==1) THEN PC := PC + 2*XXX
Описание: Вызывается ветвление, если установлен только один из проверяемых признаков *N* или *V*. Это происходит в операции знакового сравнения, когда операнд источника меньше операнда приёмника.
Признаки: не изменяются.

BGT 003000+XXX (003000–003377)

BRANCH IF GREATER THAN (ZERO) / ВЕТВЛЕНИЕ, ЕСЛИ БОЛЬШЕ (НУЛЯ)

0	0 0 0	0 1 1	0 X X	X X X	X X X

Алгоритм: IF (((*N* xor *V*) or *Z*)==0) THEN PC := PC + 2*XXX
Описание: Команда BGT, подобна команде BGE, за исключением того, что BGT не будет вызывать ветвление по нулевому результату. Это происходит в операции знакового сравнения, когда операнд источника больше операнда приёмника.
Признаки: не изменяются.

BLE 003400+XXX (003400–003777)

BRANCH IF LESS THAN OR EQUAL (TO ZERO) / ВЕТВЛЕНИЕ, ЕСЛИ МЕНЬШЕ ИЛИ РАВНО (НУЛЮ)

0	0 0 0	0 1 1	1 X X	X X X	X X X

Алгоритм: IF (((*N* xor *V*) or *Z*)==1) THEN PC := PC + 2*XXX
Описание: Команда BLE, подобна команде BLT, за исключением того, что BLE дополнительно будет вызывать ветвление по нулевому результату. Это происходит в операции знакового сравнения, когда операнд источника меньше или равен операнду приёмника.
Признаки: не изменяются.

BHI 101000+XXX (101000–101377)

BRANCH IF HIGHER / ВЕТВЛЕНИЕ, ЕСЛИ БОЛЬШЕ

1	0 0 0	0 0 1	0 X X	X X X	X X X

Алгоритм: IF ((*C* or *Z*)==0) THEN PC := PC + 2*XXX
Описание: Вызывает ветвление, если предыдущая операция не вызвала переноса и появления нулевого результата. Это происходит в операции беззнакового сравнения, когда операнд источника больше операнда приёмника.

Признаки: не изменяются.

BLOS 101400+XXX (101400–101777)

BRANCH IF LOWER OR SAME / ВЕТВЛЕНИЕ, ЕСЛИ МЕНЬШЕ ИЛИ РАВНО

1	0 0 0	0 0 1	1 X X	X X X	X X X

Алгоритм: IF ((*C* or *Z*)==1) THEN PC := PC + 2*XXX
Описание: Вызывает ветвление, если предыдущая операция вызывает перенос или появление нулевого результата. Это происходит в операции беззнакового сравнения, когда операнд источника меньше или равен операнду приёмника.
Признаки: не изменяются.

Команды перехода и управления подпрограммами.

JMP 0001DD (000100–000177)

JUMP / БЕЗУСЛОВНЫЙ ПЕРЕХОД

0	0 0 0	0 0 0	0 0 1	D D D	D D D

Алгоритм: PC := dst
Описание: Команда JMP обеспечивает возможность перехода программы на любую ячейку памяти с использованием всех методов адресации, за исключением регистрового. В отличие от других команд используется не операнд, а адрес операнда, который копируется в счётчик команд PC. Использование регистровой адресации вызывает прерывание программы по условию «запрещённый код» через вектор 4, так как переход на регистры невозможен.
Признаки: не изменяются.

JSR 004RDD (004000–004777)
CALL 0047DD (004700–004777)

JUMP TO SUBROUTINE / ОБРАЩЕНИЕ К ПОДПРОГРАММЕ

0	0 0 0	1 0 0	R R R	D D D	D D D

Алгоритм: tmp := dst; –(SP) := R; R := PC; PC := tmp
tmp := dst; –(SP) := PC; PC := tmp
Описание: Содержимое указанного регистра пересылается в стековую память и заполняется содержимым счётчика команд, который указывает на ячейку, следующую за командой JSR. Новое содержимое счётчика команд определяется адресом операнда. Так как регистр R указывает на слово, следующее за инструкцией JSR, то это даёт возможность расположить после инструкции параметры подпрограммы, которые должны быть извлечены с применением автоинкрементной или косвенно-автоинкрементной адресации через регистр R. Использование регистровой адресации вызывает прерывание программы по условию «запрещённый код» через вектор 4_8, так как переход на регистры невозможен. При исполнении команды сначала вычисляется адрес dst, и если при вычислении адреса происходит зависание, то исполнение команды прекращается и соответственно занос регистра в стек и присвоение ему значения счётчика команд не выполняются. Если вычисление адреса операнда завершилось успешно, а зависание произошло во время занесения в стек содержимого регистра, то не смотря на зависание, процессор копирует в регистр значение счётчика команд, а в счётчик команд адрес операнда, и только после этого прерывается по зависанию. При использовании в качестве регистра счётчика команд алгоритм исполнения инструкции состоит в занесении в стек старого значения счётчика команд, а новое значение счётчика команд равно адресу операнда.
Признаки: не изменяются.

RTS 00020R (000200–000207)
RETURN 000207

RETURN FROM SUBROUTINE / ВОЗВРАТ ИЗ ПОДПРОГРАММЫ

0	0 0 0	0 0 0	0 1 0	0 0 0	R R R

Алгоритм: PC := R; R := (SP)+
PC := (SP)+
Описание: Загружает содержимое регистра R в счётчик команд, после чего извлекает верхний элемент стека и засылает его в указанный регистр. Возврат из подпрограммы обычно выполняется через тот же самый регистр, который используется при обращении к ней по команде JSR. При использовании в качестве регистра счётчика команд алгоритм исполнения инструкции состоит в извлечении из стека нового значения счётчика команд.
Признаки: не изменяются.

MARK 0064NN (006400–006477)

MARK / ВОССТАНОВЛЕНИЕ УКАЗАТЕЛЯ СТЕКА

0	0 0 0	1 1 0	1 0 0	N N N	N N N

Алгоритм: SP := PC + 2 * NN; PC := R5; R5 := (SP)+
Описание: Эта команда используется для облегчения выхода из подпрограммы, заносящей в стек параметры. Команда MARK восстанавливает указатель стека во время выхода из подпрограммы. Под восстановлением указателя стека подразумевается загрузка в него нового содержимого, которое указывало бы на последнюю заполненную ячейку стека перед тем, как возникла необходимость записи в стек N параметров.
Пример:

```
        MOV     R5,-(SP)
        MOV     P1,-(SP)
        MOV     P2,-(SP)
        ....
        MOV     PN,-(SP)
        MOV     #<MARK!N>,-(SP)
        MOV     SP,R5
        CALL    SUB

        ....

SUB:
        ....
        RTS     R5
```

Признаки: не изменяются.

SOB 077RNN (077000–077777)

SUBTRACT ONE AND BRANCH (IF ≠ 0) / ВЫЧИТАНИЕ ЕДИНИЦЫ И ВЕТВЛЕНИЕ

0	1 1 1	1 1 1	R R R	N N N	N N N

Алгоритм: R := R – 1; IF (R != 0) THEN PC := PC – 2 * NN
Описание: Содержимое регистра уменьшается на единицу. Если результат не равен нулю, в PC заносится новое содержимое, определяемое вычитанием из текущего содержимого PC удвоенного смещения. Команда не может быть использована для передачи управления в прямом направлении.
Признаки: не изменяются.

Команды изменения признаков.

Эти команды позволяют устанавливать или очищать признаки *N*, *Z*, *V*, *C* в регистре слова состояния процессора PSW. Разряды признаков, соответствующие установленным разрядам в команде изменения признаков (разряды 00–03), изменяются в соответствии с состоянием разряда 04 (установки / сброса). Эти разряды устанавливаются в PSW, если установлен разряд 4, и сбрасываются, если он очищен. Общий формат команд:

000240–000277

CONDITION CODE OPERATORS / КОМАНДЫ ИЗМЕНЕНИЙ ПРИЗНАКОВ

0	0 0 0	0 0 0	0 1 0	1 0/1 N	Z V C

Частные случаи описаны ниже:

NOP 000240

NO OPERATION / НЕТ ОПЕРАЦИИ

0	0 0 0	0 0 0	0 1 0	1 0 0	0 0 0

Алгоритм: ничего не изменяется
Описание: Данная команда не очищает ни одного признака в PSW, и поэтому используется обычно в качестве пустой операции для внесения небольшой задержки в исполняемую программу. Хотя по этой команде не изменяется ни один признак, после её выполнения содержимое PSW копируется в копию слова состояния CPSW, если она не заблокирована. Аналогичные действия совершает и команда с кодом 000260 (не устанавливает ни одного признака).
Признаки: не изменяются.

CLC 000241

CLEAR C / ОЧИСТКА C

0	0 0 0	0 0 0	0 1 0	1 0 0	0 0 1

Алгоритм: *C* := 0
Описание: Данная команда очищает признак *C* в PSW.
Признаки: *N* – не изменяется; *Z* – не изменяется; *V* – не изменяется; *C* – очищается.

CLV 000242

CLEAR V / ОЧИСТКА V

0	0 0 0	0 0 0	0 1 0	1 0 0	0 1 0

Алгоритм: *V* := 0
Описание: Данная команда очищает признак *V* в PSW.
Признаки: *N* – не изменяется; *Z* – не изменяется; *V* – очищается; *C* – не изменяется.

CLZ 000244

CLEAR Z / ОЧИСТКА Z

0	0 0 0	0 0 0	0 1 0	1 0 0	1 0 0

Алгоритм: *Z* := 0
Описание: Данная команда очищает признак *Z* в PSW.
Признаки: *N* – не изменяется; *Z* – очищается; *V* – не изменяется; *C* – не изменяется.

CLN 000250

CLEAR N / ОЧИСТКА N

0	0 0 0	0 0 0	0 1 0	1 0 1	0 0 0

Алгоритм: $N := 0$
Описание: Данная команда очищает признак *N* в PSW.
Признаки: *N* – очищается; *Z* – не изменяется; *V* – не изменяется; *C* – не изменяется.

CCC 000257

CLEAR ALL CONDITION CODES / ОЧИСТКА ВСЕХ ПРИЗНАКОВ

0	0 0 0	0 0 0	0 1 0	1 0 1	1 1 1

Алгоритм: $N := 0$; $Z := 0$; $V := 0$; $C := 0$
Описание: Данная команда очищает все признаки *N*, *Z*, *V*, *C* в PSW.
Признаки: *N* – очищается; *Z* – очищается; *V* – очищается; *C* – очищается.

SEC 000261

SET C / УСТАНОВКА C

0	0 0 0	0 0 0	0 1 0	1 1 0	0 0 1

Алгоритм: $C := 1$
Описание: Данная команда устанавливает признак *C* в PSW.
Признаки: *N* – не изменяется; *Z* – не изменяется; *V* – не изменяется; *C* – устанавливается.

SEV 000262

SET V / УСТАНОВКА V

0	0 0 0	0 0 0	0 1 0	1 1 0	0 1 0

Алгоритм: $V := 1$
Описание: Данная команда устанавливает признак *V* в PSW.
Признаки: *N* – не изменяется; *Z* – не изменяется; *V* – устанавливается; *C* – не изменяется.

SEZ 000264

SET Z / УСТАНОВКА Z

0	0 0 0	0 0 0	0 1 0	1 1 0	1 0 0

Алгоритм: $Z := 1$
Описание: Данная команда устанавливает признак *Z* в PSW.
Признаки: *N* – не изменяется; *Z* – устанавливается; *V* – не изменяется; *C* – не изменяется.

SEN 000270

SET N / УСТАНОВКА N

0	0 0 0	0 0 0	0 1 0	1 1 1	0 0 0

Алгоритм: $N := 1$
Описание: Данная команда устанавливает признак *N* в PSW.
Признаки: *N* – устанавливается; *Z* – не изменяется; *V* – не изменяется; *C* – не изменяется.

SCC 000277

SET ALL CONDITION CODES / УСТАНОВКА ВСЕХ ПРИЗНАКОВ

0	0 0 0	0 0 0	0 1 0	1 1 1	1 1 1

Алгоритм: $N := 1$; $Z := 1$; $V := 1$; $C := 1$
Описание: Данная команда устанавливает все признаки *N*, *Z*, *V*, *C* в PSW.
Признаки: *N* – устанавливается; *Z* – устанавливается; *V* – устанавливается; *C* – устанавливается.

Комбинации указанных выше операций очистки или установки, соединённых по схеме ИЛИ, могут образовывать комбинированные команды. Например, команда с кодом 000243_8 одновременно очищает признаки *C* и *V*.

Команды прерывания программы.

EMT 104000–104377

EMULATOR TRAP / КОМАНДНОЕ ПРЕРЫВАНИЕ ДЛЯ СИСТЕМНЫХ ПРОГРАММ

1	0 0 0	1 0 0	0 . .	. . .	. . .

Алгоритм: –(SP) := CPSW; –(SP) := CPC; PC := (30_8); PSW := (32_8)
Описание: Команды с кодами 104000_8 до 104377_8 являются командами EMT. Текущее содержимое CPSW и CPC пересылается в стековую память и замещается содержимым ячеек 30_8 и 32_8, иначе говоря реализуется прерывание по вектору 30_8. Алгоритм функционирования команды прерывания рассмотрен выше при описании работы микропрограммы обработки прерываний.
Признаки: *N*, *Z*, *V*, *C* – загружаются содержимым ячейки с адресом 32_8.

TRAP 104400–104777

TRAP / КОМАНДНОЕ ПРЕРЫВАНИЕ

1	0 0 0	1 0 0	1 . .	. . .	. . .

Алгоритм: –(SP) := CPSW; –(SP) := CPC; PC := (34_8); PSW := (36_8)
Описание: Команды с кодами 104400_8 до 104777_8 являются командами TRAP. Текущее содержимое CPSW и CPC пересылается в стековую память и замещается содержимым ячеек 34_8 и 36_8, иначе говоря реализуется прерывание по вектору 34_8. Алгоритм функционирования команды прерывания рассмотрен выше при описании работы микропрограммы обработки прерываний.
Признаки: *N*, *Z*, *V*, *C* – загружаются содержимым ячейки с адресом 36_8.

IOT 000004

INPUT/OUTPUT TRAP / КОМАНДНОЕ ПРЕРЫВАНИЕ ДЛЯ ВВОДА/ВЫВОДА

0	0 0 0	0 0 0	0 0 0	0 0 0	1 0 0

Алгоритм: –(SP) := CPSW; –(SP) := CPC; PC := (20_8); PSW := (22_8)
Описание: Текущее содержимое CPSW и CPC пересылается в стековую память и замещается содержимым ячеек 20_8 и 22_8, иначе говоря реализуется прерывание по вектору 20_8. Алгоритм функционирования команды прерывания рассмотрен выше при описании работы микропрограммы обработки прерываний.
Признаки: *N*, *Z*, *V*, *C* – загружаются содержимым ячейки с адресом 22_8.

BPT 000003

BREAKPOINT TRAP / КОМАНДНОЕ ПРЕРЫВАНИЕ ДЛЯ ОТЛАДКИ

0	0 0 0	0 0 0	0 0 0	0 0 0	0 1 1

Алгоритм: –(SP) := CPSW; –(SP) := CPC; PC := (14_8); PSW := (16_8)
Описание: Текущее содержимое CPSW и CPC пересылается в стековую память и замещается содержимым ячеек 14_8 и 16_8, иначе говоря реализуется прерывание по вектору 14_8. Алгоритм функционирования команды прерывания рассмотрен выше при описании работы микропрограммы обработки прерываний. Запрещается употребление кода 000003_8 в программах, которые выполняются под управлением подпрограмм отладки. Этот код обычно употребляется отладчиками для расстановки точек останова.
Признаки: *N*, *Z*, *V*, *C* – загружаются содержимым ячейки с адресом 16_8.

RTI 000002

RETURN FROM INTERRUPT / ВОЗВРАТ ИЗ ПРЕРЫВАНИЯ

0	0 0 0	0 0 0	0 0 0	0 0 0	0 1 0

Алгоритм: PC := (SP)+; IF (PC>=160000_8) THEN PSW[8:0] := (SP)+ ELSE PSW[7:0] := (SP)+
Описание: Новые содержимые PC и PSW загружаются из стека. Если новое значение PC больше или равно 160000_8, то из стека загружается и 8-й разряд PSW (H/U), в ином случае он остается неизменным. Данная особенность даёт возможность возвратиться из режима USER в режим HALT, если прерывание USER было вызвано из режима HALT. Если при загрузке нового PSW устанавливается бит T в PSW, то отладочное прерывание будет вызвано до исполнения первой команды нового процесса. Если перед исполнением RTI был установлен T-бит, а при загрузке нового PSW он очищается, то в этом случае всё равно будет вызвано отладочное прерывание, но в стеке сохранится уже новое PSW без установленного бита T. Если во время первой выборки из стека происходит зависание, то второй выборки не происходит и процессор прерывается (т.е. указатель стека SP увеличивается на 2, а не на 4). Загрузка новых значений PC и PSW происходит из того же адресного пространства, из которого выполняется команда RTI. Если возврат происходит из режима USER или режима HALT с разрешёнными прерываниями в режим HALT с запрещёнными прерываниями, то после загрузки нового значения PC изменяется и CPC, т.к. в данный момент ещё регистры копий разблокированы, а CPSW остаётся неизменным, т.к. новое значение PSW блокирует изменение копий, т.е. в данном случае значение CPC равно новому значению PC, а CPSW равно значению PSW до исполнения команды RTI.
Признаки: *N*, *Z*, *V*, *C* – загружаются из стека.

RTT 000006

RETURN FROM TRAP / ВОЗВРАТ ИЗ ОТЛАДОЧНОГО ПРЕРЫВАНИЯ

0	0 0 0	0 0 0	0 0 0	0 0 0	1 1 0

Алгоритм: PC := (SP)+; IF (PC>=160000_8) THEN PSW[8:0] := (SP)+ ELSE PSW[7:0] := (SP)+
Описание: Алгоритм работы команда RTT аналогичен команде RTI. Единственным отличием является то, что если при загрузке нового PSW устанавливается бит T, то после исполнения RTT не запускается блок обработки прерываний, и соответственно выполняется команда нового процесса. Эта особенность используется отладчиками при возврате из подпрограммы обработки прерывания по вектору 14 (отладочное прерывание по T-разряду).
Признаки: *N*, *Z*, *V*, *C* – загружаются из стека.

Команды управления микропроцессором.

HALT 000000

HALT / ОСТАНОВ

0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0

Алгоритм: PC := ($SEL170_8$); PSW := ($SEL170_8$+2)
Описание: Микропроцессор переводится в режим HALT для загрузки новых значений PC и PSW из адресного пространства HALT. С помощью процедуры безадресного чтения считывается безадресный регистр SEL и формируется вектор прерывания VECT[15:8] := SEL[15:8], VECT[7:0] := 170_8, иначе говоря реализуется прерывание режима HALT по вектору $SEL170_8$. Алгоритм функционирования команды прерывания рассмотрен выше при описании работы микропрограммы обработки прерываний. Эта команда обычно используется для перехода в пультовый отладчик.
Признаки: *N*, *Z*, *V*, *C* – загружаются содержимым ячейки с адресом $SEL170_8$+2.

WAIT 000001

WAIT FOR INTERRUPT / ОЖИДАНИЕ ПРЕРЫВАНИЯ

0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 1

Алгоритм: флаг ожидания := 1
Описание: По команде WAIT процессор выставляет флаг ожидания, в результате чего прекращается выборка команд из памяти. Счётчик команд указывает на следующую, после WAIT, команду. Сброс флага ожидания осуществляет любое незамаскированное прерывание (ACLO, HALT, EVNT, VIRQ). После исполнения подпрограммы обработки прерывания исполнение программы продолжается с точки останова. Если эта команда выполнена в режиме HALT с запрещёнными прерываниями (установлены биты H/U и P в PSW), то микропроцессор подвисает, т.к. выборка команд из памяти запрещена и сбросить флаг ожидания невозможно, т.к. все прерывания замаскированы. Команда WAIT игнорирует бит T в PSW, это значит, что после исполнения WAIT прерывания по вектору 14 при установленном T-разряде в PSW не произойдет (флаг ожидания маскирует прерывание по T-разряду).
Признаки: не изменяются.

RESET 000005

RESET EXTERNAL BUS / СБРОС ВНЕШНИХ УСТРОЙСТВ

0	0 0 0	0 0 0	0 0 0	0 0 0	1 0 1

Описание: По этой команде вырабатывается сигнал INIT на выходе микропроцессора. Внешние устройства, присоединённые к каналу микропроцессора, устанавливаются в исходное состояние. По этой команде также сбрасывается и триггер сигнала EVNT, что снимает запрос на прерывание по событию (вектор 100). Триггер сигнала ACLO по этой команде не сбрасывается.
Признаки: не изменяются.

Команды расширенной арифметики (EIS).

MUL 070RSS (070000–070777)

MULTIPLY / УМНОЖЕНИЕ

0	1 1 1	0 0 0	R R R	s s s	s s s

Алгоритм: tmp32 := R * (src); R := tmp32[31:16]; R or 1 := tmp32[15:0]
Описание: Перемножаются операнды источника и приёмника, взятые в двоичном дополнительном коде. Результат помещается в регистр, используемый в качестве приёмника, и в следующий за ним регистр, если регистр приёмника имеет чётный номер. Если же регистр приемника имеет нечётный номер, сохраняется только младшая часть результата.
Признаки: *N* – устанавливается, если результат tmp32 меньше нуля, в противном случае очищается, *Z* – устанавливается, если результат tmp32 равен нулю, в противном случае очищается, *V* – очищается, *C* – устанавливается, если результат меньше, чем -2^{15}(-32768), или больше, чем 2^{15}–1(32767), в противном случае очищается.

DIV 071RSS (071000–071777)

DIVIDE / ДЕЛЕНИЕ

0	1 1 1	0 0 1	R R R	s s s	s s s

Алгоритм: tmp32[31:16] := R; tmp32[15:0] := R or 1; tmp32[частное, остаток] := tmp32 / (src); R or 1 := tmp32[15:0](остаток) ; R := tmp32[31:16](частное)
Описание: 32-разрядное слово в двоично-дополнительном коде, находящееся в регистрах R и R or 1, делится на операнд источника. Частное заносится в R, а остаток в R or 1. После выполнения операции деления знак остатка будет таким же, как и у делимого. Если в качестве R используется нечётный регистр, то старшая часть 32-разрядного делимого будет повторять его младшую часть, т.е. фактически выполняется операция ((R << 16) or R) и после завершения операции деления сохранится только частное. При арифметическом переполнении во время операции (частное не помещается в 16 разрядов) или делении на ноль выполнение операции прекращается, регистры не изменяются, очищаются признаки *N* и *Z* в PSW, устанавливается признак *V*.
Признаки: *N* – устанавливается, если частное меньше нуля, в противном случае (и в случае арифметического переполнения) очищается, *Z* – устанавливается, если частное равно нулю, в противном случае (и в случае арифметического переполнения) очищается, *V* – устанавливается при арифметическом переполнении и делении на ноль ((src)==0), в противном случае очищается, *C* – устанавливается, если осуществляется деление на ноль ((src)==0), в противном случае очищается.

ASH 072RSS (072000–072777)

ARITHMETIC SHIFT / АРИФМЕТИЧЕСКИЙ СДВИГ

0	1 1 1	0 1 0	R R R	s s s	s s s

Алгоритм: R := R сдвинутое на NN позиций влево или вправо, где NN – шесть младших разрядов (src).
Описание: Содержимое указанного регистра сдвигается влево или вправо на количество позиций, определяемое счётчиком сдвига. Функцию счётчика сдвига выполняют шесть младших разрядов операнда источника, представленных в двоичном дополнительном коде. Значение счётчика сдвига может изменяться в пределах от –32 до +31. Отрицательному значению соответствует единица в старшем разряде счётчика сдвига и обеспечивает сдвиг вправо, положительному значению соответствует 0 и сдвиг влево.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается, *Z* – устанавливается, если результат равен нулю, в противном случае очищается,

V – устанавливается, если во время проведения операции сдвига было зафиксировано изменение знакового разряда результата (возможно только при сдвиге влево), в противном случае очищается, *C* – загружается содержимым последнего выдвинутого разряда.

ASHC 073RSS (073000–073777)

ARITHMETIC SHIFT COMBINED / АРИФМЕТИЧЕСКИЙ СДВИГ ДВОЙНОГО СЛОВА

0	1 1 1	0 1 1	R R R	s s s	s s s

Алгоритм: tmp32[31:16] := R; tmp32[15:0] := R or 1; tmp32 := tmp32 сдвинутое на NN позиций влево или вправо, где NN – шесть младших разрядов (src); R := tmp32[31:16]; R or 1 := tmp32[15:0]

Описание: Содержимое регистров R и R or 1 копируется в 32-разрядную переменную и сдвигается влево или вправо на количество позиций, определяемое счётчиком сдвига. Функцию счётчика сдвига выполняют шесть младших разрядов операнда источника, представленных в двоичном дополнительном коде. Значение счётчика сдвига может изменяться в пределах от –32 до +31. Отрицательному значению соответствует единица в старшем разряде счётчика сдвига и обеспечивает сдвиг вправо, положительному значению соответствует 0 и сдвиг влево. Если в качестве R используется нечётный регистр, то старшая часть 32-разрядной переменной будет повторять её младшую часть, т.е. фактически выполняется операция ((R << 16) or R) и после завершения операции сдвига сохранится только младшая часть.

Признаки: *N* – устанавливается, если результат tmp32 меньше нуля, в противном случае очищается, *Z* – устанавливается, если результат tmp32 равен нулю, в противном случае очищается, *V* – устанавливается, если во время проведения операции сдвига было зафиксировано изменение знакового разряда результата tmp32 (возможно только при сдвиге влево), в противном случае очищается, *C* – загружается содержимым последнего выдвинутого разряда tmp32.

Команды арифметики с плавающей запятой (FIS).

В микропроцессоре команды арифметики с плавающей запятой не реализованы на микропрограммном уровне. Для их исполнения используется программный уровень, для чего при исполнении инструкций FIS (коды 075000_8–075037_8) процессор переходит в режим HALT, прочитывает безадресный регистр SEL и анализирует состояние его 7-го разряда. Если разряд установлен, то считается, что программа эмуляции FIS отсутствует и процессор прерывается по вектору 10_8 (резервный код). При сброшенном разряде 7 исполняется прерывание по вектору $SEL010_8$, который указывает на программу обработки инструкций FIS. Следует заметить, что как и все прерывания HALT, прерывание по обработке FIS не сохраняет старые значения PC и PSW, поэтому команды FIS должны употребляться только в режиме USER (H/U в PSW равен нулю) или режиме HALT с разрешёнными прерываниями (H/U равен единице, а P равен нулю), а программа обработки должна начинаться в режиме HALT с запрещёнными прерываниями (H/U и P равны единице), т.е. в новом значении PSW по адресу $SEL010_8$+2 должны быть установлены биты 7 и 8. Это даёт возможность прочесть старые значения PC и PSW из заблокированных копий CPC и CPSW.

В УКНЦ программа эмуляции инструкций FIS расположена в адресном пространстве HALT по адресу 165612_8. Она эмулирует четыре команды арифметики с плавающей запятой: сложение (FADD), вычитание (FSUB), умножение (FMUL) и деление (FDIV). В эмуляторе используются вещественные числа одинарной точности длиной 32 разряда.

15	14 … 07	06 … 00
s	Порядок	Старшая часть мантиссы

15	00
Младшая часть мантиссы	

S – разряд 15 старшей части аргумента используется как знаковый разряд. Единица в этом разряде говорит о том, что число отрицательное. Порядок – восемь разрядов старшей части аргумента используется для задания порядка. Порядок задается числом от 0 до 255, при этом если порядок равен нулю, то это используется для задания вещественного нуля (0.0). Остальные значения интерпретируются по формуле $2^{(\text{порядок-128})}$, то есть порядок может принимать значения от 2^{-127} до 2^{+127}, или от 5,877471761E-39 до 1,7014118326E+38. Мантисса – 23 разряда дробной части с двоичной запятой слева. Мантисса всегда нормализована, то есть самый старший разряд справа от двоичной запятой всегда равен единице, является скрытым (избыточным) и он интерпретируется как 2^{-1}. Остальные разряды мантиссы интерпретируются от 2^{-2} до 2^{-24}. Соответственно мантисса может принимать значения от 0.5 до 0.99999994. Таким образом вещественное число представляется формулой знак*$2^{(\text{порядок-128})}$*мантисса. По абсолютной величине самое малое число составляет 2.9387358805E–39, а самое большое 1,7014117311E+38.
Примеры чисел с плавающей запятой:

$9 = 16 * 0.5625 = 2^4 * (2^{-1} * 2^{-4}) =$ 0 10000100 (1)00100000000000000000000 = 041020 000000

Самое малое абсолютное число:

$2.9387358805E{-}39 = 5.877471761E{-}39 * 0.5 = 2^{-127} * (2^{-1}) =$ 0 00000001 (1)00000000000000000000000 = 000200 000000

Самое большое абсолютное число:

$1{,}7014117311E{+}38 = 1.7014118326E{+}38 * 0.99999994 = 2^{127} * (2^{-1} * \ldots * 2^{-24}) =$ 0 11111111 (1)11111111111111111111111 = 077777 177777

Операнды, используемые при выполнении команд FIS, хранятся в памяти. Содержимое регистра R указывает на адрес аргументов, которые расположены следующим образом:

(R) – адрес старшей части аргумента B;
(R+2) – адрес младшей части аргумента B;
(R+4) – адрес старшей части аргумента A;
(R+6) – адрес младшей части аргумента A.

После выполнения команды FIS результат помещается на место аргумента A, а используемый РОН увеличивается на 4 и будет указывать на старшую часть аргумента A.
Значение регистра R обязательно должно быть чётным, и если оно нечётное, то эмулятор FIS эмулирует прерывание по вектору 4_8, занося в стек значение PSW до выполнения команды FIS и значение PC, который указывает на следующее слово после команды FIS, значение R не изменяется (если оно не является SP или PC). Если во время выполнения операций с плавающей запятой возникает ошибка, то эмулятор эмулирует прерывание по вектору 244_8 (вектор прерывания по ошибкам FIS). Значение R не изменяется, в стек помещаются код ошибки (на место PSW) и значение PC, которое указывает на следующее слово после команды FIS. Ошибки могут возникать следующие:

– переполнение порядка (2). Возникает, если во время нормализации мантиссы значение порядка становится больше, чем 2^{127}.

– исчезновение порядка (12_8). Возникает, если во время нормализации мантиссы значение порядка становится меньше, чем 2^{-127}.

– деление на ноль (13_8). Возникает во время эмуляции команды FDIV, если аргумент B равен вещественному нулю (0.0).

FADD 07500R (075000–075007)

FLOATING ADD / СЛОЖЕНИЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

0	1 1 1	1 0 1	0 0 0	0 0 0	R R R

Алгоритм: A := A + B; R := R + 4
Описание: Складываются аргументы A и B. Результат помещается на место аргумента A.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается, *Z* – устанавливается, если результат равен нулю, в противном случае очищается, *V* – очищается, *C* – очищается.

FSUB 07501R (075010–075017)

FLOATING SUBTRACT / ВЫЧИТАНИЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

0	1 1 1	1 0 1	0 0 0	0 0 1	R R R

Алгоритм: A := A – B; R := R + 4
Описание: Из аргумента A вычитается аргумент B. Результат помещается на место аргумента A.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается, *Z* – устанавливается, если результат равен нулю, в противном случае очищается, *V* – очищается, *C* – очищается.

FMUL 07502R (075020–075027)

FLOATING MULTIPLY / УМНОЖЕНИЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

0	1 1 1	1 0 1	0 0 0	0 1 0	R R R

Алгоритм: A := A * B; R := R + 4
Описание: Аргумент A умножается на аргумент B. Результат помещается на место аргумента A.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается, *Z* – устанавливается, если результат равен нулю, в противном случае очищается, *V* – очищается, *C* – очищается.

FDIV 07503R (075030–075037)

FLOATING DIVIDE / ДЕЛЕНИЕ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

0	1 1 1	1 0 1	0 0 0	0 1 1	R R R

Алгоритм: A := A / B; R := R + 4
Описание: Аргумент A делится на аргумент B. Результат помещается на место аргумента A.
Признаки: *N* – устанавливается, если результат меньше нуля, в противном случае очищается, *Z* – устанавливается, если результат равен нулю, в противном случае очищается, *V* – очищается, *C* – очищается.

Команды режима HALT.

В режиме работы HALT в микропроцессоре присутствует группа команд для работы с копиями счётчика команд СРС и слова состояния процессора CPSW, работы с памятью адресного пространства USER, запуска по значению СРС и чтения безадресного регистра. Эти команды можно использовать только тогда, когда процессор работает в режиме HALT, при использовании этих команде в режиме USER процессор прерывается по вектору 10_8 (резервная команда).

START/RUN 000010–000013

ПУСК

0	0 0 0	0 0 0	0 0 0	0 0 1	0 . .

Алгоритм: PC := CPC; PSW := CPSW
Описание: Эта команда позволяет осуществить запуск по установленным значениям копий счётчика команд СРС и слова состояния процессора CPSW. В PSW копируются все 9 битов, таким образом новый процесс можно запускать как в режиме USER, так и в режиме HALT. Так как эта команда работает с регистрами копий СРС и CPSW, то она должна выполняться при заблокированных значениях СРС и CPSW, т.е. бит 7 в PSW должен быть установлен. После выполнения этой команды сразу же запускается блок обработки прерываний, и если есть незамаскированные прерывания, то процессор прерывается до исполнения первой команды нового процесса.
Признаки: загружаются из копии слова состояния процессора CPSW.

STEP/$STEP$ 000014–000017

ШАГ

0	0 0 0	0 0 0	0 0 0	0 0 1	1 . .

Алгоритм: PC := CPC; PSW := CPSW
Описание: Алгоритм исполнения этой команды ничем не отличается от алгоритма исполнения команды START за исключением того, что после выполнения этой команды блок обработки прерываний не запускается, и процессор сразу же переходит к исполнению первой команды нового процесса. Обычно эта команда используется в пультовом отладчике для пошаговой отладки исполняемого кода. В данном случае обязательным условием является наличие активного уровня на входе HALT. В этом случае после исполнения первой команды нового процесса запускается блок обработки прерываний и происходит прерывание по сигналу HALT (вектор $SEL170_8$).
Признаки: загружаются из копии слова состояния процессора CPSW.

RSEL/$MFSL$ 000020

ЧТЕНИЕ БЕЗАДРЕСНОГО РЕГИСТРА

0	0 0 0	0 0 0	0 0 0	0 1 0	0 0 0

Алгоритм: R0 := SEL
Описание: По этой команде по процедуре безадресного чтения прочитывается безадресный регистр SEL и помещается в регистр R0. Старший байт этого безадресного регистра используется для формирования векторов прерываний HALT. В младшем байте установленный бит 7 указывает на отсутствие программы обработки команд FIS (075000_8–075037_8), что вызывает в данном случае прерывание процессора по вектору 10_8, в противном случае вызывается обработчик команд FIS по вектору $SEL010_8$. В УКНЦ в магистрали как центрального процессора, так и периферийного, регистр SEL равен 160000_8, то есть вектора HALT начинаются с адреса 160000_8 и присутствует программа обработки команд FIS.

Признаки: не изменяются.

MFUS/$MFPM$ 000021

ЧТЕНИЕ ПАМЯТИ АДРЕСНОГО ПРОСТРАНСТВА USER

0	0 0 0	0 0 0	0 0 0	0 1 0	0 0 1

Алгоритм: R0 := (R5)+
Описание: Данная команда на микропрограммном уровне сбрасывает бит H/U в PSW и прочитывает ячейку памяти по адресу, находящемуся в R5, помещая прочитанное содержимое в регистр R0. После чтения содержимое R5 увеличивается на 2, и в PSW устанавливается бит H/U. Если во время чтения произошло зависание, то несмотря на то, что производится чтение адресного пространства USER, процессор прерывается по вектору зависания в режиме HALT со значением $SEL004_8$, значение R5 в этом случае все равно увеличивается на 2, а содержимое R0 не изменяется.
Признаки: не изменяются.

RCPC/$MFPC$ 000022, 000023

ЧТЕНИЕ РЕГИСТРА КОПИИ СЧЕТЧИКА КОМАНД

0	0 0 0	0 0 0	0 0 0	0 1 0	0 1 .

Алгоритм: R0 := CPC
Описание: Эта команда помещает в регистр R0 значение регистра копии счётчика команд CPC. Естественно, что смысл эта команда имеет только в режиме работы с запрещёнными прерываниями, когда регистры копий заблокированы. Она обычно используется в подпрограммах обработки прерываний HALT, для того чтобы узнать старое значение счётчика команд прерванного процесса.
Признаки: не изменяются.

RCPS/$MFPS$ 000024–000027

ЧТЕНИЕ РЕГИСТРА КОПИИ СЛОВА СОСТОЯНИЯ ПРОЦЕССОРА

0	0 0 0	0 0 0	0 0 0	0 1 0	1 . .

Алгоритм: R0 := CPSW
Описание: Эта команда помещает в регистр R0 значение регистра копии слова состояния процессора CPSW. Естественно, что смысл эта команда имеет только в режиме работы с запрещёнными прерываниями, когда регистры копий заблокированы. Она обычно используется в подпрограммах обработки прерываний HALT, для того чтобы узнать старое значение слова состояния процессора прерванного процесса.
Признаки: не изменяются.

MTUS/$MTPM$ 000031

ЗАПИСЬ В ПАМЯТЬ АДРЕСНОГО ПРОСТРАНСТВА USER

0	0 0 0	0 0 0	0 0 0	0 1 1	0 0 1

Алгоритм: –(R5) := R0
Описание: Данная команда на микропрограммном уровне сбрасывает бит H/U в PSW, уменьшает значение регистра R5 на 2 и записывает в ячейку памяти по адресу, находящемуся в R5 содержимое регистра R0. После записи в PSW устанавливается бит H/U. Если во время записи произошло зависание, то несмотря на то, что производится запись адресного пространства USER, процессор прерывается по вектору зависания в режиме HALT со значением $SEL004_8$.
Признаки: не изменяются.

WCPC/$MTPC$ 000032, 000033

ЗАПИСЬ РЕГИСТРА КОПИИ СЧЕТЧИКА КОМАНД

0	0 0 0	0 0 0	0 0 0	0 1 1	0 1 .

Алгоритм: CPC := R0

Описание: Эта команда помещает в регистр копии счётчика команд CPC значение регистра R0. Естественно, что смысл эта команда имеет только в режиме работы с запрещёнными прерываниями, когда регистры копий заблокированы. Она обычно используется в пультовом отладчике и программе обработки команд FIS, для того чтобы установить точку пуска для команд START/STEP.

Признаки: не изменяются.

WCPS/$MTPS$ 000034–000037

ЗАПИСЬ РЕГИСТРА КОПИИ СЛОВА СОСТОЯНИЯ ПРОЦЕССОРА

0	0 0 0	0 0 0	0 0 0	0 1 0	1 . .

Алгоритм: CPSW := R0

Описание: Эта команда помещает в регистр копии слова состояния процессора CPSW значение регистра R0. Естественно, что смысл эта команда имеет только в режиме работы с запрещёнными прерываниями, когда регистры копий заблокированы. Когда она исполняется в режиме с разрешёнными прерываниями, то регистр CPSW сохраняет свое установленное значение до первой команды, изменяющей регистр PSW (запись в PSW, изменение признаков арифметическими и логическими командами). Она обычно используется в пультовом отладчике и программе обработки команд FIS, для того чтобы установить новое значение PSW для команд START/STEP.

Признаки: не изменяются.

Ну и наконец еще одна команда, исполняющаяся только в режиме HALT, смысл которой я так и не понял.

?????? 000030

НЕИЗВЕСТНАЯ КОМАНДА

0	0 0 0	0 0 0	0 0 0	0 1 1	0 0 0

Алгоритм:

```
R0:=0;
while ((R0[7]==0) || (R2[7]==0))
 {
  R1:=R1<<1 | 0; R2[07:00]:=R2[07:00]<<1 | C;
  R2[15:08]:=R2[7]; R3:=R3<<1 | C; R0++;
 }
N:=0; Z:=(R0==0); V:=0; C:=0
```

Описание: По этой команде сперва очищается регистр R0. Далее исполняется цикл, окончанием которого является установка в разряде 07 R0 или R2 единицы. В цикле над регистрами проводятся следующие действия: регистры с R1 по R3 сдвигаются влево, при этом в R1 в младший разряд вдвигается ноль, а в R2 и R3 – содержимое разряда *C*, при этом старшая часть R2 расширяется знаковым разрядом младшей части, R0 инкрементируется. Так как останов исполнения команды производится при наличии единицы в разряде 7 в R0 или R2, то после исполнения команды R0 может принимать значения от 0 до 10_8 или 200_8. Значение 200_8 получается в том случае, если до исполнения операции младшая часть R2 была равна нулю и был сброшен бит *C*.

Признаки: *N* – очищается, *Z* – устанавливается, если значение в R0 равно нулю, в противном случае очищается, *V* – очищается, *C* – очищается.