Серия 1836

МИКРОСХЕМА ИНТЕГРАЛЬНАЯ КН1836ВМ3

ИТТиП, Зеленоград

Оглавление

1. Введение

Микросхема 1836ВМ3 представляет собой однокристальный шестнадцатиразрядный микропроцессор, предназначенный для обработки цифровой информации. Микросхема используется для встраивания в аппаратуру потребителя и может применяться в составе технологического оборудования, в контрольно-измерительных комплексах и в системах обработки цифровой информации общего назначения.

1.1. Основные технические данные

Формат данных	*Дополнительный двоичный код с фиксированной запятой*
Разрядность	*16 разрядов с возможностью обработки 8-ми и 32-ух разрядных слов*
Система команд	*Программная совместимость с PDP-11 фирмы DEC*
Виды адресации	*Регистровая, косвенно - регистровая, автоинкрементная, автодекрементная, косвенно - автоинкрементная, косвенно - автодекрементная, индексная, косвенно - индексная*
Число регистров общего назначения	*8*
Количество каналов передачи информации	*1*
Количество уровней запроса прерывания	*4*
Максимальный объем адресуемой памяти	*4 Мбайт*
Тактовая частота	*16 МГц*
Напряжение питания	*5В ± 5%*

1.2. Электрические параметры микросхемы 1836ВМ3

Наименование	**Обозначение**	**MIN**	**MAX**
Выходное напряжение низкого уровня, В	U_{OL}	*-*	*0.4*
Выходное напряжение высокого уровня, В	U_{OH}	*2.4*	*-*
Ток потребления, мА	I_{CC}	*-*	
Входное напряжение низкого уровня, В	U_{IL}	*-*	*0.8*
Входное напряжение высокого уровня, В	U_{IH}	*2.0*	*-*
Напряжение источника питания, В	U_{DD}	*-0.3*	*7.0*
Допустимое входное напряжение, В	U_{IN}	*-0.3*	U_{DD}*+0.3*

1.3. Структурная схема процессора

Общая структурная схема процессора приведена на рис.1.1. Процессор состоит из следующих блоков:

- *операционный блок;*
- *блок микропрограммного управления;*
- *блок прерываний;*
- *диспетчер памяти;*
- *контроллер системной магистрали.*

Блоки соединены между собой шиной адреса и данных, которая предварительно преобразовывается в блоке контроллера системной магистрали. Кроме того, блоки связаны специальными информационными и управляющими сигналами.

Операционный блок *предназначен для выполнения следующих функций:*

- *прием данных и их хранение в регистрах;*
- *прием и расшифровка микрокоманды;*
- *выполнение арифметико-логических операций;*
- *выдача данных в системную магистраль;*
- *формирование адресов векторов прерывания;*
- *формирование состояний.*

В операционном блоке находятся 16 основных регистров, из которых 8 являются регистрами общего назначения, один регистр PSW - слово состояния процессора, шесть - внутренние служебные регистры. Программно доступными являются все регистры общего назначения и регистр PSW. **Блок микропрограммного управления** *выполняет преобразование команды в последовательность микрокоманд.* **Блок прерываний** *предназначен для организации приоритетной системы прерываний процессора. Здесь принимаются и обрабатываются как внешие запросы на прерывание так и внутренние. Блок вырабатывает сигналы перехода к обработке прерывани для операционного блока и блока микропрограммного управления.* **Диспетчер памяти** *обеспечивает расширение емкости адресуемой памяти до 256 Кбайт, перераспределение виртуального адреса в физический и защиту памяти в системах с разделением времени.* **Контроллер системной магистрали** *предназначен для управления выборкой последовательности команд из памяти, для управления совмещением операций и согласования работы блоков синхронизации, для организации циклов обмена данными с внешней памятью и внутренними регистрами.*

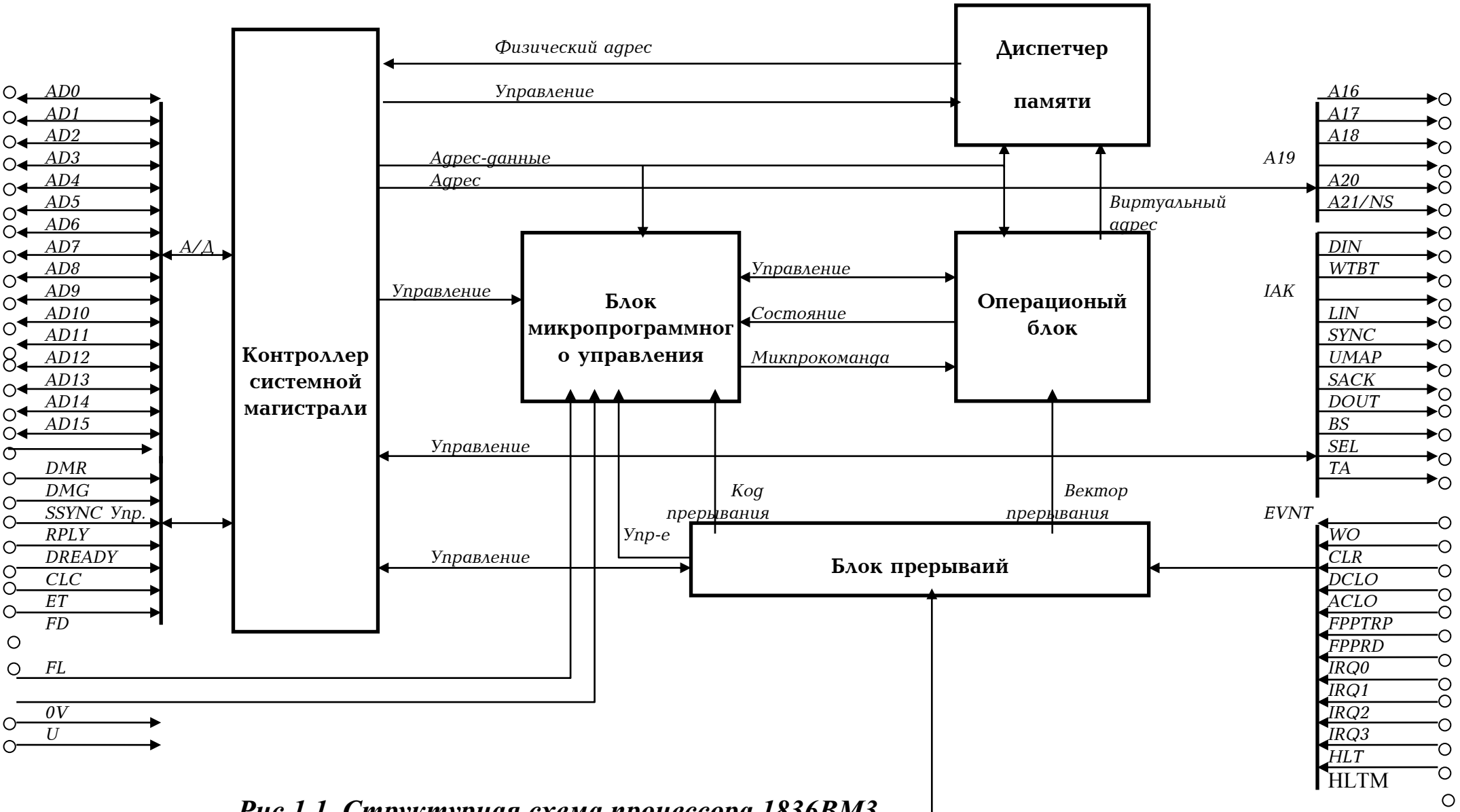


Рис 1.1. Структурная схема процессора 1836ВМ3

2. Регистры процессора

Процессор содержит восемь шестнадцатиразрядных регистров общего назначения (РОН), один регистр PSW - слово состояния процессора, шесть внутренних служебных регистров. Программно доступными являются все регистры общего назначения и регистр PSW. Остальные регистры доступны микропрограммно, причем регистры R0÷R4 доступны лишь из поля адресации команды.

2.1. Регистры общего назначения

Регистры общего назначения могут служить в качестве накопительных регистров, индексных регистров, регистров автоинкрементной и автодекрементной адресации, указателей стека и для других целей. РОН используется для выборки операндов и записи результатов при выполнении арифметико-логических операций аналогично ячейкам памяти и регистрам внешних устройств. Регистры R6 и R7 имеют, кроме того, специальное назначение. R6 используется как указатель стека (SP) и содержит адрес последней заполненной ячейки стека. R7 служит счетчиком команд (PC) и содержит адрес очередной выполняемой команды. Обычно он используется для целей адресации и не используется как накопительный регистр.

Операции по выполнению команд с регистровым методом адресации являются внутренними по отношению к процессору и не требуют выполнения циклов обращения к каналу (за исключением цикла выборки команды). обмен же данными с памятью и внешними устройствами выполняется через канал и занимает более длительное время. Таким образом, использование РОН для хранения операндов повышает скорость выполнения программ.

2.2. Слово состояния процессора (PSW)

Слово состояния процессора содержит информацию о текущем состоянии процессора. Это информация о текущем приоритете процессора, об операционной моде процессора, значение кодов условий ветвления, зависящих от результатов выполнения команд и состояние Т-разряда, используемого при отладке программ и вызывающего прерывание программ. Формат PSW показан на рис. 2.1.

15 14	13 12	11 8	7 5	4	3	2	1	0
Текущая мода	**Предыдущая мода**	**XXXX**	**Приоритет**	**T**	**N**	**Z**	**V**	**C**

Рис. 2.1. Слово состояния процессора

Тринадцатый и двенадцатый разряды PSW содержат информацию о предыдущей моде процессора, а пятнадцатый и четырнадцатый разряды - о текущей моде процессора. Возможность и способы загрузки различных разрядов приведены в табл. 2.1.

Так как выполнение команды MTPS в моде пользователя аппаратурой разрешено, то необходимо в математическом обеспечении предусмотреть запрет обращения по адресу PSW пользователю.

Процессор работает в любом из восеми уровней приоритета, от 0 до 7. Когда код приоритета равен 7, внешние устройства не могут вызвать прерывания программы. Если код приоритета больше или равен 4, то работа процессора может прерываться в случае возникновения сигналов IRQ3÷0.

Коды условий ветвления содержат информацию о результате последней выполненной процессором команды:

N - отрицательный результат
Z - нулевой результат
V - арифметическое переполнение
C - перенос

Процедуру их установки в соответствующее состояние выполняют все арифметические и логические команды. Установка отдельных разрядов этих кодов выполняется в следующих случаях:

Z = 1, если результат равен 0;
N = 1, если результат отрицательный;
V = 1, если в результате выполнения операции произошло арифметическое переполнение;
C = 1, если в результате выполнения операции произошел перенос из самого старшего разряда или, при сдвиге вправо или влево, из самого младшего или самого старшего разряда была выдвинута единица.

При загрузке нового слова состояния процессора может установится или очиститься T-разряд. Если он установлен, то по завершении выполнения текущей команды будет вызвано прерывание с вектором 14. Использование T-разряда особенно эффективно в отладочных программах.

	RTI, RTT		*Команды прерываний и внешние прерывания*		*Запись в PSW по адресу 177776*		*MTPS*	
Разряды PSW	*Пользователь*	*ОС*	*Пользователь*	*ОС*	*Пользователь*	*ОС*	*Пользователь*	*ОС*
Код признаков (биты 3-0)	*Загружается из стека*	*Загружается из стека*	*Загружается из вектора прерывания*	*Загружается из вектора прерывания*	*Загружается из источника*	*Загружается из источника*	*Загружается из источника*	*Загружается из источника*
Т-бит (бит 4)	*Загружается из стека*	*Загружается из стека*	*Загружается из вектора прерывания*	*Загружается из вектора прерывания*	*Не изменяется*	*Не изменяется*	*Не изменяется*	*Не изменяется*
Уровень приоритета (биты 7-5)	*Не изменяется*	*Загружается из стека*	*Загружается из вектора прерывания*	*Загружается из вектора прерывания*	*Загружается из источника*	*Загружается из источника*	*Загружается из источника*	*Загружается из источника*
Предыдущая мода (биты 13,12)	*Не изменяется*	*Загружается из стека*	*Переписывается из битов 15,14*	*Переписывается из битов 15,14*	*Загружается из источника*	*Загружается из источника*	*Не доступны*	*Не доступны*
Текущая мода (биты 15,14)	*Не изменяется*	*Загружается из стека*	*Загружается из вектора прерывания*	*Загружается из вектора прерывания*	*Загружается из источника*	*Загружается из источника*	*Не доступны*	*Не доступны*

Таблица 2.1. Способы загрузки разрядов PSW

3. Система команд

В процессоре используется 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 определяет:

- *выполняемую функцию (код операции);*
- *регистры общего назначения, используемые при выборе операндов;*
- *метод адресации (способ использования выбранного РОН).*

Большая часть информации, обрабатываемой ЭВМ, представляет собой данные, сформированные в массивы, списки, потоки символов и т.д. Поэтому процессор разработан с учетом возможности эффективной обработки сформированных структур данных.

Регистры общего назначения могут быть использованы:

- *как накопители (обрабатываемые данные хранятся в регистрах);*
- *как указатели адреса (регистр содержит адрес операнда, а не сам операнд);*
- *как указатели адреса, содержимое которых изменяется автоматически с заданным шагом, что позволяет обращаться к последовательно расположенным ячейкам памяти.*
- *как индексные регистры, содержимое которых прибавляется к индексному слову для вычисления адреса операнда, что позволяет обращаться к различным элементам списка.*

Автоматическое увеличение содержимого указателя адреса при обращении к последовательным ячейкам памяти носит название автоинкрементной адресации. Автоматическое уменьшение содержимого указателя при обращении к последовательным ячейкам памяти носит название автодекрементной адресации. Эти методы могут быть использованы при обработке табулированных данных.

Использование автоинкрементной и автодекрементной адресации дает возможность организации стековой памяти. В качестве указателя стека программно может быть использован любой РОН, однако определенные команды (используемые при обслуживании прерываний, обращения и возврата из подпрограмм) автоматически используют R6 как аппаратный указатель стека.

3.1. Формат одноадресных команд.

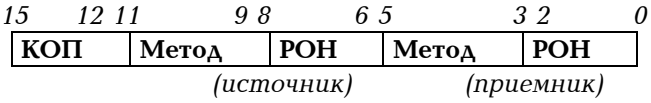
Формат одноадресных команд (таких как очистка, проверка) имеют следующий вид:

Разряды 15÷6 содержат код операции, который определяет исполняемую команду. Разряды 5÷0 образуют шестирязрядное поле, именуемое полем адресации операнда-приемника, которое в свою очередь состоит из двух полей:

1) *разряды 2÷0 определяют один из 8 РОН, который использует данная команда;*
2) *разряды 5÷3 определяют способ использования выбранного регистра (метод адресации). Причем разряд 3 определяет прямую или косвенную адресацию.*

3.2. Формат двухадресных команд

Операции над двумя операндами (такие как сложение, пересылка, сравнение) выполняются с помощью команд, в которых имеются два адреса. Задание разрядов в поле адресации первого и второго операндов определяют различные методы адресации и различные регистры общего назначения. Формат двухадресной команды имеет следующий вид:



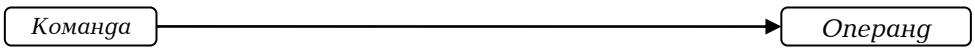
Поле адресации операнда источника используется для выборки операнда источника или первого операнда. Поле адресации операнда приемника используется для выборки второго операнда (операнда приемника) и занесения результата. Например по команде ADD A,B складывается содержимое ячейки A (операнд источника) с содержимым ячейки B (операнд приемника). После выполнения операции сложения в ячейке B будет находиться результат операции, а содержимое ячейки A не изменится.

3.3. Методы адресации

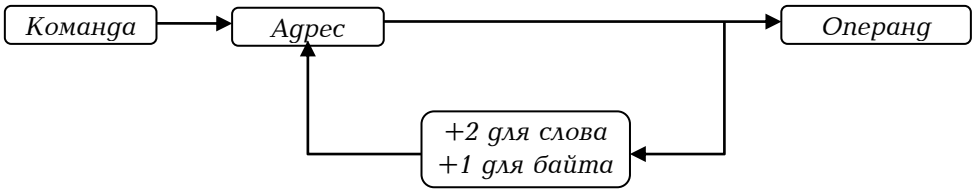
3.3.1. Методы прямой адресации

На рис.3.1. показаны последовательности операций выполнения команды с каждым из четырех методов прямой адресации. При регистровом методе адресации любой из восеми РОН может быть использован как накопитель. Следовательно, операнд будет находиться в выбранном регистре. Так как РОН реализованы аппаратно в процессоре 1836ВМ3, они обладают более высоким быстродействием, чем любая память, работающая под управлением процессора. Это преимущество особенно проявляется при операциях с переменными, к которым необходимо часто обращаться.

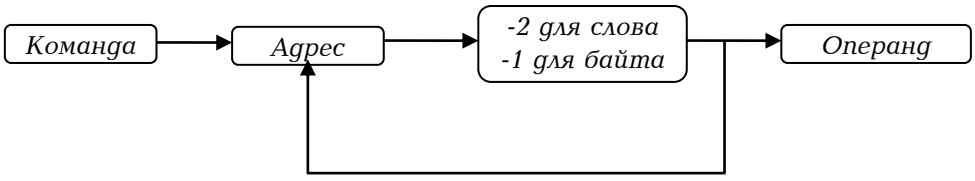
1) Регистровый метод адресации (метод 0)



2) Автоинкрементный метод адресации (метод 1)



3) Автодекрементный метод адресации (метод 4)



4) Индексный метод адресации (метод 6)

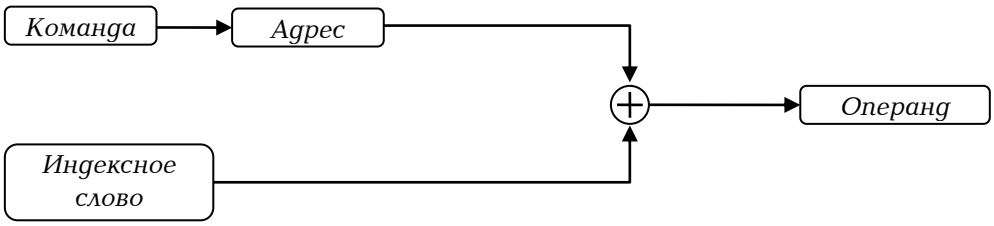
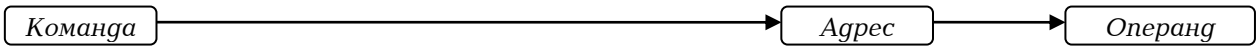


Рис. 3.1. Методы прямой адресации

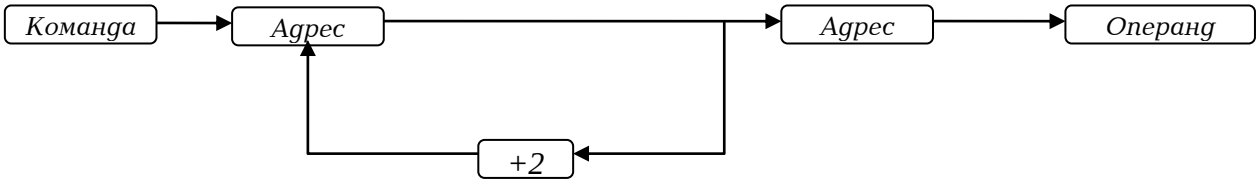
3.3.2. Методы косвенной адресации.

Четыре основных метода могут быть использованы с косвенной адресацией. Если в регистровом методе операндом является содержимое выбранного регистра, в косвенно-регистровом методе содержимое выбранного регистра является адресом операнда. В трех других косвенных методах содержимое регистра позволяет выбрать адрес операнда, а не сам операнд. Эти методы используются, когда таблица состоит из адресов, а не операндов. На рис.3.2. показаны последовательности операций выполнения команды с каждым из четырех методов косвенной адресации.

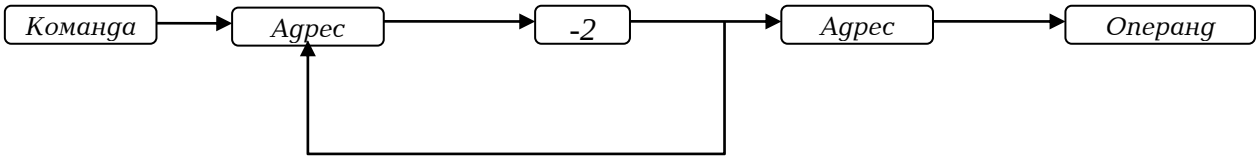
1) Косвенно-регистровый метод адресации (метод 1)



2) Косвено-автоинкрементный метод адресации (метод 3)



3) Косвено-автодекрементный метод адресации (метод 5)



4)Косвенно-индексный метод адресации (метод 7)

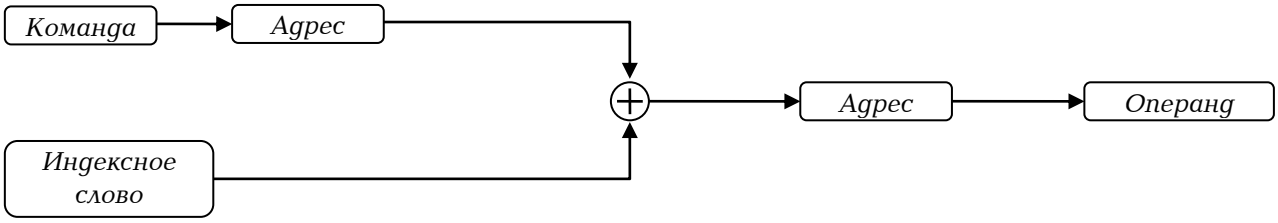


Рис. 3.2. Методы косвенной адресации

3.3.3. Использование счетчика команд (PC) в качестве регистра общего назначения

Регистр R7, являясь одним из восеми РОН, выполняет в центральном процессоре специальную функцию счетчика команд. Когда процессор использует счетчик команд для выборки слова из памяти, его содержимое автоматически увеличивается на 2. Новое содержимое счетчика команд является адресом слова, используемого при выполнении данной команды. Следует отметить, что при работе с байтами содержимое PC также увеличивается на 2.

Счетчик команд может быть использован во всех методах адресации, применяемых в процессоре. Однако, наиболее эффективно он используется только с четырьмя методами адресации. Эти методы адресации получили специальные наименования: непосредственный, абсолютный, относительный и косвенно-относительный. Использование этих методов дает возможность построения программы, работоспособность которой не теряется при перемещении ее в любую область памяти. В табл.3.1. приведены методы адресации с использованием R7. Методы адресации с использованием PC в значительной мере упрощают обработку данных, не сформированных в массивы.

Двоичный код	**Наименование**	**Функция**
010	*Непосредственный #N*	*Операнд выбирается из ячейки, следующей за командным словом*
011	*Абсолютный @#A*	*Из ячейки, следующей за командным словом, выбирается адрес операнда*
110	*Относительный X(PC) или A*	*Операнд выбирается из ячейки, адрес которой определяется как сумма содержимого PC и ячейки, следующей за командным словом*
111	*Косвенно-относительный @X(PC) или @A*	*Из ячейки, адрес которой определяется как сумма содержимого PC и ячейки, следующей за командным словом, выбирается адрес операнда*

Таблица 3.1 Методы адресации с использованием PC

Непосредственный метод адресации *имеет символическое обозначение #N. Он эквивалентен автоинкрементному методу адресации через PC. Этот метод обеспечивает удобство написания программы и экономию времени программиста путем помещения константы в ячейку памяти вслед за командным словом. Процессор выбирает командное слово и увеличивает PC на 2. В поле адреса операнда источника записан код 27, следовательно, PC используется как указатель при выборке операнда перед увеличением его содержимого на 2, для указания на следующую команду.*

Абсолютный метод адресации *имеет символическое обозначение @#A. Он эквивалентен косвенно-автоинкрементной адресации через PC. Этот метод удобен тем, что адрес операнда является его абсолютным адресом (т.е. он остается постоянным независимо от места расположения программы в памяти).*

Относительный метод адресации *имеет символическое обозначение X(PC) или A, где X - исполнительный адрес по отношению к счетчику команд. Этот метод эквивалентен индексной адресации через PC. Индексное слово хранится в следующей за командным словом ячейке и, будучи сложенным с содержимым PC, дает адрес операнда. Этот метод полезен при написании программы, которая может располагаться в различных местах памяти, так как адрес операнда фиксируется по отношению к PC. При перемещении программы в памяти операнд перемещается на то же число ячеек, что и сама команда.*

Косвенно-относительный метод адресации *имеет символическое обозначение @X(PC) или @A, где X - адрес ячейки, содержащий исполнительный адрес, по отношению к счетчику команд. Этот метод эквивалентен косвенно-индексной адресации через PC.*

3.3.4. Использование указателя стека (SP) в качестве регистра общего назначения

Регистр R6, являясь одним из РОН, используется в процессоре как указатель адреса при обращении к той части памяти, которая отводится под стек. С помощью автодекрементной адресации через R6 данные записываются в стек, а с помощью автоинкрементной адресации производится выборка данных из стека. Индексный метод адресации позволяет производить произвольную выборку элементов стека.

Так как R6(SP) используется для обслуживания прерываний, то его особенностью является то, что уменьшение и увеличение содержимого SP всегда производится с шагом два. В байтовых операциях содержимое ячеек с нечетными адресами не изменяется.

4.Описание команд

При описании команд используются следующие обозначения:

R	*- регистр общего назначения*
PC	*- счетчик команд (R7)*
SP	*- указатель стека (R6)*
PSW	*- регистр состояния процессора*
SS	*- поле адресации операнда источника*
src	*- приемник*
(src)	*- операнд приемника*
DD	*- поле адресации операнда приемника*
dst	*- приемник*
(dst)	*- операнд приемника*
XXX	*- смещение (8 разрядов)*
NN	*- смещение (6 разрядов)*
(A)	*- содержимое ячейки A*
$\wedge$	*- логическое умножение (И)*
$\vee$	*- логическое сложение (ИЛИ)*
$\oplus$	*- исключающее ИЛИ*
$\overline{A}$	*- отрицание A (НЕ)*
$\leftarrow$	*- становится равным*
$\downarrow$	*- запись в стек*
$\uparrow$	*- выборка из стека*
B	*- байтовая команда*
temp	*- временное хранение*

4.1. Одноадресные команды

CLR / CLRB - очистка

Код команды: *0050DD / 1050DD*

Действие: *(dst) ← 0*

Признаки:

N ← 0
Z ← 1
V ← 0
C ← 0

Описание: *В указанную ячейку записываются нули. Для байтовой команды нули записываются в указанный байт. Обнуление ячейки происходит в цикле записи.*

COM / COMB - инвертирование

Код команды: *0051DD / 1051DD*

Действие: *(dst) ← $\overline{(dst)}$*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 0
C ← 1

Описание: *Заменяет содержимое указанной ячейки его двоичным обратным кодом (каждый разряд, содержащий "0" устанавливается, а каждый разряд, содержащий "1", очищается). Для байтовой команды операция производится по отношению к указанному байту.*

INC / INCB - прибавление единицы

Код команды: *0052DD / 1052DD*

Действие: *(dst) ← (dst) + 1*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если операнд равен 077777, иначе V ← 0
C не изменяется

Описание: *Прибавляет единицу к содержимому указанной ячейки (или байту, если команда байтовая).*

DEC / DECB - вычитание единицы

Код команды: *0053DD / 1053DD*

Действие: *(dst) ← (dst) - 1*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если операнд был равен 100000, иначе V ← 0
C не изменяется

Описание: *Из содержимого указанной ячейки (или указанного байта для байтовых команд) вычитается единица.*

NEG / NEGB - изменение знака

Код команды: *0054DD / 1054DD*

Действие: *(dst) ← -(dst)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если результат = 100000, иначе V ← 0
C ← 0, если результат = 0, иначе C ← 1

Описание: *Содержимое указанной ячейки (для байта для байтовых команд) заменяется двоичным дополнением операнда. Следует заметить, что число 100000 заменяется самим собой, так как не существует соответствующего ему положительного числа.*

TST / TSTB - тестирование

Код команды: *0057DD / 1057DD*

Действие: *AC ← (dst)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если содержимое = 0, иначе Z ← 0
V ← 0
C ← 0

Описание: *В аккумулятор процессора считывается содержимое ячейки или регистра. В зависимости от содержимого указанной ячейки (или байта для байтовых команд) устанавливаются или очищаются признаки N и Z.*

ASR / ASRB - арифметический сдвиг вправо

Код команды: *0062DD / 1062DD*

Действие: *(dst) ← сдвинутое на один разряд вправо (dst)*

Признаки:

N ← 1, если старший разряд результата установлен, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← N ⊕ C
C ← содержимое младшего разряда указанной ячейки

Описание: *Сдвигаются все разряды операнда вправо на одну позицию. Содержимое знакового разряда восстанавливается. С-разряд загружается содержимым младшего разряда операнда.*

ASL / ASLB - арифметический сдвиг влево

Код команды: *0063DD / 1063DD*

Действие: *(dst) ← сдвинутое на один разряд влево (dst)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← N ⊕ C
C ← содержимое старшего разряда операнда

Описание: *Сдвигаются все разряды операнда на одну позицию влево. В младший разряд операнда записывается нуль. С-разряд загружается содержимым старшего разряда операнда. Таким образом, ASL или ASLB выполняет умножение числа со знаком на 2.*

ROR / RORB - циклический сдвиг вправо

Код команды: *0060DD / 1060DD*

Действие: *(dst) ← циклически сдвинутое на один разряд вправо (dst)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← N ⊕ C
C ← содержимое младшего разряда операнда

Описание: *Циклически сдвигает все разряды операнда на одну позицию вправо. Содержимое младшего разряда загружается в С-разряд, а прежнее содержимое С-разряда загружается в старший разряд операнда.*

ROL / ROLB - циклический сдвиг влево

Код команды: *0061DD / 1061DD*

Действие: *(dst) ← циклически сдвинутое на один разряд влево (dst)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← N ⊕ C
C ← содержимое старшего разряда операнда

Описание: *Циклически сдвигаются все разряды операнда на одну позицию влево. Содержимое старшего разряда загружается в С-разряд, а прежнее содержимое С-разряда загружается в младший разряд операнда.*

ADC / ADCB - прибавление переноса

Код команды: *0055DD / 1055DD*

Действие: *(dst) ← (dst) + (C)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если перед выполнением операции (dst) = 077777 и (C) = 1, иначе V ← 0
C ← 1, если перед выполнением операции (dst) = 177777 и (C) = 1, иначе C ← 0

Описание: *Операнд складывается с содержимым С-разряда.*

SBC / SBCB - вычитание переноса

Код команды: *0056DD / 1056DD*

Действие: *(dst) ← (dst) - (C)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если перед выполнением операции (dst) = 100000 и (C) = 1, иначе V ← 0
C ← 1, если перед выполнением операции (dst) = 0 и (C) = 1, иначе C ← 0

Описание: *Вычитается содержимое С-разряда из операнда. Это позволяет вычесть перенос, получившийся при вычитании двух младших шестнадцатиразрядных слов, из результата вычитания двух старших шестнадцатиразрядных слов.*

SXT - расширение знака

Код команды: *0067DD*

Действие:

(dst) ← 0, если N очищен
(dst) ← -1, если N установлен

Признаки:

N не изменяется
Z ← 1, если (N) = 0
V ← 0
C не изменяется

Описание: *Если (N) = 1, операнд заменяется кодом 177777. Если (N)=0, указанная ячейка очищается. Эта команда обычно используется при выполнении операций с повышенной точностью для расширения знака в тридцатидвухразрядном слове.*

SWAB - перестановка байтов

Код команды: *0003DD*

Действие: *(Байт1, Байт0) ← (Байт0, Байт1)*

Признаки:

N ← 1, если установлен бит 7 результата, иначе N ← 0
Z ← 1, если младший байт результата = 0, иначе Z ← 0
V ← 0
C ← 0

Описание: *Меняет местам и старший байт с младшим байтом указанной ячейки. Адресация происходит по полному слову.*

MFPS - чтение PSW

Код команды: *1067DD*

Действие: *(dst) ← PSW*

Признаки:

N ← 1, если установлен бит 7 слова состояния процессора, иначе N ← 0
Z ← 1, если все восемь разрядов PSW = 0, иначе Z ← 0
V ← 0
C не изменяется

Описание: *Восемь разрядов слова состояния процессора PSW пересылаются в указанную ячейку. Если при этом используется регистровый метод адресации, в указанном регистре происходит расширение знака. Адрес операнда приемника воспринимается как адрес байта.*

MTPS - запись PSW

Код команды: *1064DD*

Действие: *PSW ← (src)*

Признаки: *Устанавливаются или очищаются в соответствии с разрядами 0-3 источника*

Описание: *Восемь разрядов указанного операнда замещают содержимое слово состояния процессора. Адрес операнда источника воспринимается как адрес байта. Следует заметить, что T-разряд (разряд 4 PSW) не может быть установлен этой командой. Операнд источника (src) не изменяет своего содержимого. Эта команда может быть использована для изменения приоритетных разрядов PSW (разряды 5-7). Загрузка PSW из ячейки памяти происходит в цикле чтения.*

4.2. Двухадресные команды

Использование двухадресных команд обеспечивает экономию машинного времени и сокращает количество команд в программе. Список двухадресных команд содержит 4 арифметические команды и 4 логические команды.

4.2.1. Арифметические команды

MOV / MOVB - пересылка

Код команды: *01SSDD / 11SSDD*

Действие: *(dst) ← (src)*

Признаки:

N ← 1, если (src) < 0, иначе N ← 0
Z ← 0, если (src) = 0, иначе Z ← 0
V ← 0
C не изменяется

Описание: *Операнд источника (src) пересылается по адресу операнда приемника. Прежнее содержимое ячейки dst теряется. Содержимое ячейки src не изменяется. При операциях с байтами команда MOVB с использованием регистрового метода адресации (единственная среди байтовых команд) расширяет старший разряд младшего байта (расширение знака). Все разряды старшего байта устанавливаются или сбрасываются в зависимости от того, установлен или сброшен (знаковый) разряд младшего байта. В других случаях MOVB оперирует с байтами так, как MOV со словами.*

CMP / CMPB - сравнение

Код команды: *02SSDD / 12SSDD*

Действие: *(src) - (dst)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если было арифметическое переполнение, иначе V ← 0
C ← 1, если был перенос из старшего разряда результата, иначе C ← 0

Описание: *Сравнивает операнды источника и приемника и изменяет признаки, которые затем могут быть использованы для команд условных переходов. Оба операнда не изменяются. Единственным действием является изменение признаков. За командой сравнения обычно следует команда условного ветвления. Заметим, что в отличие от команды вычитания, порядок действия следующий: (src) - (dst), а не (dst) - (src).*

ADD - сложение

Код команды: *06SSDD*

Действие: *(dst) ← (src) + (dst)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если было арифметическое переполнение, иначе V ← 0
C ← 1, если был перенос из старшего разряда результата, иначе C ← 0

Описание: *Операнд источника (src) складывается с операндом приемника (dst) и результат записывается по адресу операнда приемника. Первоначальное содержимое теряется. Содержимое src не изменяется. Сложение выполняется в двоичном дополнительном коде.*

SUB - вычитание

Код команды: *16SSDD*

Действие: *(dst) ← (dst) - (src)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если было арифметическое переполнение, иначе V ← 0
C ← 1, если был перенос из старшего разряда результата, иначе C ← 0

Описание: *Из операнда приемника вычитается операнд источника (src) и результат записывается по адресу dst. Первоначальное содержимое теряется, а содержимое src остается без изменения. При арифметических операциях с удвоенной точностью установка C-разряда означает заем единицы из старшей части вычитаемого.*

4.2.2. Логические команды

BIT / BITB - проверка разрядов

Код команды: *03SSDD / 13SSDD*

Действие: *(src) ∧ (dst)*

Признаки:

Z ← 1, если все разряды результата = 0, иначе Z ← 0
N ← 1, если старший разряд результата установлен, иначе N ← 0
V ← 0
C не изменяется

Описание: *Выполняет логическую функцию "И" над (src) и (dst), изменяя соответствующим образом признаки. Оба операнда не изменяют своего значения. Команда BIT используется для проверки состояния разрядов операнда (src), для которых установлены соответствующие разряды в операнде (dst).*

BIC / BICB - очистка разрядов

Код команды: *04SSDD / 14SSDD*

Действие: *(dst) ← ($\overline{src}$) ∧ (dst)*

Признаки:

N ← 1, если старший разряд результата установлен, иначе N ← 0
Z ← 0
V не изменяется
C ← 1, если все разряды результата очищены, иначе C ← 0

Описание: *Очищает каждый разряд операнда (dst), соответствующий установленному разряду операнда (src). Первоначальное содержимое dst теряется. Содержимое src не изменяется.*

BIS / BISB - установка разрядов

Код команды: *05SSDD / 15SSDD*

Действие: *(dst) ← (src) ∨ (dst)*

Признаки:

N ← 1, если старший разряд результата установлен, иначе N ← 0
Z ←1, если все разряды результата очищены, иначе Z ←0
V ← 0
C не изменяется

Описание: *Выполняет логическую операцию "ИЛИ" над содержимым src и dst записывает результат по адресу dst. Разряды (dst) устанавливаются в "1", если соответствующие им разряды (src) находятся в "1". Прежнее содержимое dst теряется, а содержимое src остается неизменным.*

XOR - исключающее ИЛИ

Код команды: *074RDD*

Действие: *(dst) ← R ⊕ (dst)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 0
C не изменяется

Описание: *Выполняет операцию "исключающее ИЛИ" над содержимым указанного регистра и содержимым dst. Результат записывается в dst. Содержимое регистра R не изменяется.*

4.3. Команды управления программой

К командам управления программой относятся команды ветвления, перехода к подпрограмме, возврата из подпрограммы, безусловного перехода и другие.

4.3.1. Команды ветвления

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

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

Восьмиразрядное смещение позволяет производить ветвление в обратном направлении максимально на 2008 слов от слова, на которое указывает текущее содержимое PC и на 1778 слов в прямом направлении.

BR - ветвление безусловное

Код команды: *000400 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX*

Признаки: *не изменяются*

Описание: *Обеспечивает способ передачи управления в программе ячейки, адрес которой находится в ограниченной области, с помощью одного слова команды. Новое содержимое PC = текущее содержимое PC + 21 (смещение), где текущее содержимое PC = адрес команды ветвления + 2.*

4.3.2. Простые условные ветвления

BNE - ветвление, если равно (нулю)

Код команды: *001000 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если Z = 0*

Признаки: *не изменяются*

Описание: *Проверяет состояние разряда Z и вызывает ветвление, если он очищен. BNE обратная по действию BEQ. Вместе с командой CMP она используется для проверки того, что установленные разряды операнда источника соответствуют установленным разрядам операнда приемника. В общем случае она используется для проверки неравенства нулю результата предыдущей операции.*

BEQ - ветвление если равно (нулю)

Код команды: *014000 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если Z = 1*

Признаки: *не изменяются*

Описание: *Проверяет состояние разряда Z и вызывает ветвление, если он установлен. Вместе с командой CMP она используется для проверки равенства двух величин. Вместе с командой BIT используется для проверки того, что очищенные разряды операнда источника соответствуют установленным разрядам операнда приемника. В общем случае эта команда используется для проверки равенства нулю результата предыдущей операции.*

BPL - ветвление, если плюс

Код команды: *100000 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если N = 0*

Признаки: *не изменяются*

Описание: *Проверяет разряд N и вызывает ветвление, если он очищен, BPL обратна по действию команде BMI.*

BMI - ветвление, если минус

Код команды: *100400 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если N = 1*

Признаки: *не изменяются*

Описание: *Проверяет состояние N-разряда и вызывает ветвление, если он установлен. Она используется для проверки знака (старший разряд) результата предыдущей операции.*

BVC - ветвление, если нет арифметического переполнения

Код команды: *102000 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если V = 0*

Признаки: *не изменяются*

Описание: *Проверяет состояние разряда V и вызывает ветвление, если он очищен.*

BVS - ветвление, если арифметическое переполнение

Код команды: *102400 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если V = 1*

Признаки: *не изменяются*

Описание: *Проверяет состояние разряда V и вызывает ветвление, если он установлен. BVS используется для обнаружения арифметического переполнения в результате исполнения предыдущей операции. BVS обратна по действию команде BVC.*

BCC - ветвление, если нет переноса

Код команды: *103000 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если C = 0*

Признаки: *не изменяются*

Описание: *Проверяет состояние разряда C и вызывает ветвление, если он очищен.*

BCS - ветвление, если перенос

Код команды: *103400 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если C = 1*

Признаки: *не изменяются*

Описание: *Проверяет разряд C и вызывает ветвление, если он установлен, BCS используется для проверки наличия переноса в результате предыдущей операции. BCS обратна по действию команде BCC. Условные ветвления по результату операций над числами. Особые комбинации разрядов признаков проверяются с помощью команд условного ветвления по результату операций над числами. Эти команды используются для проверки результатов команд, в которых операнды рассматриваются как двоичные числа, имеющие знак.*

BGE - ветвление, если больше или равно (нулю)

Код команды: *002000 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если N ⊕ V = 0*

Признаки: *не изменяются*

Описание: *Вызывает ветвление, если оба разряда признаков N и V установлены или очищены. Таким образом, BGE всегда будет вызывать ветвление, если она следует за операцией сложения двух положительных чисел. BGE будет также вызывать ветвление по нулевому результату.*

BLT - ветвление, если меньше (нуля)

Код команды: *002400 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если N ⊕ V = 1*

Признаки: *не изменяются*

Описание: *Вызывает ветвление, если результат операции "исключающее ИЛИ" над содержимым разрядов N и V равен 1. Команда BLT обратна по действию команде BGE. Таким образом, BLT будет всегда вызывать ветвление, если она следует за командой сравнения отрицательного операнда источника и положительного операнда приемника даже, если произошло переполнение. BLT никогда не будет вызывать ветвление, если она следует за командой сравнения (CMP) положительного операнда источника и отрицательного операнда приемника. BLT не будет вызывать ветвления, если результат предыдущей операции равен 0 без переполнения.*

BGT - ветвление, если больше (нуля)

Код команды: *003000 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если Z ∨ (N ⊕ V) = 0*

Признаки: *не изменяются*

Описание: *Команда BGT подобна команде BGE, за исключением того, что BGT не будет вызывать ветвления по нулевому результату.*

BLE - ветвление, если меньше или равно

Код команды: *003400 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если Z ∨ (N ⊕ V) = 1*

Признаки: *не изменяются*

Описание: *Команда BLE подобна команде BLT, но дополнительно будет вызывать ветвление, если результат предыдущей операции был равен нулю. Условные ветвления по результату операции над кодами. Условные ветвления по результату операций над кодами обеспечивают методы проверки результата операций сравнения операндов, рассматриваемых как величины без знака.*

BHI - ветвление, если больше

Код команды: *101000 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если C ∨ Z = 0*

Признаки: *не изменяются*

Описание: *Вызывает ветвление, если предыдущая операция не вызвала переноса и появления нулевого результата. Это происходит при операциях сравнения (CMP), когда операнд источника больше операнда приемника.*

BLOS - ветвление, если меньше или равно

Код команды: *101400 + XXX*

Действие: *(PC) ← (PC) + 2 * XXX, если C ∨ Z = 1*

Признаки: *не изменяются*

Описание: *Вызывает ветвление, если предыдущая операция вызывает перенос или появление нулевого результата. Команда BLOS является обратной по действию команде BHI. Ветвление будет происходить, если операнд источника меньше или равен операнду приемника.*

BHIS - ветвление, если больше или равно

Описание: *По своему действию команда BHIS идентична команде BCC. Эта мнемоника вводится только для удобства.*

BLO - ветвление, если меньше

Описание: *По своему действию команда BLO идентична команде BCS. Эта мнемоника вводится только для удобства.*

JMP - безусловный переход

Код команды: *0001DD*

Действие: *(PC) ← адрес dst*

Признаки: *не изменяются*

Описание: *Команда JMP обеспечивает возможность перехода программы на любую ячейку памяти с использованием всех методов адресации, за исключением регистрового. Использование регистровой адресации вызывает прерывание программы по условию "запрещенная команда" через адрес вектора 10. Метод косвенной адресации может применяться и вызывает передачу управления программой по адресу, содержащемуся в указанном регистре. Заметим, что команды - это полные слова и, поэтому должны выбираться из ячеек с четным адресом. Косвенно-индексный метод адресации позволяет командой JMP передать управление по адресу, являющемуся элементом таблицы адресов. Команды обращения к подпрограмме и выхода из подпрограммы. Эти команды обеспечивают возможность автоматического вложения программ, выход из подпрограмм, многократный вход в подпрограмму. Подпрограммы могут обращаться к другим подпрограммам (или к самим себе) без специального обеспечения хранения адресов возврата. Процедура обращения к подпрограмме и выхода из нее не изменяет подпрограмму. Это позволяет использовать одну и ту же подпрограмму несколькими процессами, осуществляющими прерывание программы.*

4.3.3. Команды обращения к подпрограмме и выхода из подпрограммы

JSR - обращение к подпрограмме

Код команды: *004RDD*

Действие:

↓(SP) ← (R)
(R) ← (PC)
(PC) ← адрес dst

Признаки: *не изменяются*

Описание: *При выполнении команды JSR старое содержимое указанного регистра ("указателя связи") автоматически засылается в стек, и новая связующая информация поступает в регистр. Таким образом, обращение к подпрограммам, вложенным в подпрограммы на любую глубину, осуществляются с помощью регистра "указателя связи". Нет необходимости в том, чтобы задавать максимальную глубину обращения к данной подпрограмме или включать команды запоминания и восстановления "указателя связи " в каждую подпрограмму. Так как вся связующая информация сохраняется в стеке, выполнение программы может быть прервано и подпрограмма обслуживания прерывания может обращаться к той же самой прерванной подпрограмме. Выполнение подпрограммы может быть затем возобновлено по окончании обслуживания прерывания. Этот процесс, называемый вложением, может продолжиться до любого уровня. Обращение к подпрограмме по команде JSR может осуществляться с помощью автоинкрементной адресации (если каждый последующий вход в подпрограмму осуществляется через ячейку, адрес которой на 2 больше предыдущего) или индексной адресации (если вход в подпрограмму осуществляется по адресам, расположенным в произвольном порядке), а также с помощью косвенных методов адресации.*

Команда JSR PC, dst является особым случаем обращения к подпрограмме. В этом случае не изменяется содержимое ни одного из общих регистров, кроме PC. Другим особым случаем команды JSR является JSR PC, @ (SP) + , при выполнении которой последняя заполненная ячейка стека и PC обмениваются содержимым. Использование этой команды позволяет двум подпрограммам попеременно передавать управление друг другу и каждый раз возобновлять работу с того места, где осуществлялась передача управления. Такие подпрограммы называются "со-программами".

Возврат из подпрограммы осуществляется командой RTS. По команде RTS содержимое регистра передается в PC, а содержимое верхней ячейки стека - в указанный регистр.

RTS - возврат из подпрограммы

Код команды: *00020R*

Действие:

(PC) ← (R)
(R) ← (SP)↑

Признаки: *не изменяются*

Описание: *Загружает содержимое регистра (R) в PC, 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. Таким образом, выход из подпрограммы, обращение к которой осуществлялось командой JSR PC, dst выполняется командой RTS PC, а выход из подпрограммы, обращение к которой осуществлялось командой JSR R5, dst с использованием любого из методов адресации, выполняется командой RTS R5.*

MARK - восстановление указателя стека

Код команды: *0064NN*

Действие:

*(SP) ← (PC) + 2 * NN*
(PC) ← (R5)
(R5) ← (SP)↑

Признаки: *не изменяются*

Описание: *Эта команда используется для облегчения выхода из подпрограммы. При использовании стека для записи в него параметров команда MARK восстанавливает указатель стека (SP) во время выхода из подпрограммы.*

SOB - вычитание единицы и ветвление

Код команды: *077RNN*

Действие:

(R) ← (R) - 1;
если результат ≠ 0, (PC) ← (PC) - 2
если результат = 0, (PC) ← (PC)

Признаки: *не изменяются*

Описание: *Содержимое регистра уменьшается на единицу. Если результат ≠ 0, в счетчик команд загружается новое содержимое, определяемое вычитанием из текущего содержимого PC удвоенного смещения. В команде SOB смещением является шестиразрядное положительное число. Эта команда может быть эффективно использована для организации различного рода счетчиков. Следует заметить, что команда SOB не может быть использована для передачи управления в прямом направлении.*

4.4. Команды прерывания программы

Команды прерывания обеспечивают обращение к моделирующим программам, программам управления вводом-выводом, программам отладки и программам, разработанным пользователем. Когда происходит прерывание, текущее содержимое счетчика команд и содержимое регистра состояния процессора записывается в стек. Новое содержимое PC и PSW загружается из вектора прерывания, состоящего из двух слов. При выходе из прерывания используются команды RTI и RTT, которые восстанавливают PC и PSW, извлекая их прежнее содержание из стека. Векторы прерывания расположены по фиксированным, приписанным каждому виду прерывания, адресам.

EMT - командное прерывание для системных программ

Код команды: *104000 ÷ 104377*

Действие:

↓(SP) ← (PSW)
↓(SP) ← (PC)
(PC) ← (30)
(PSW) ←(32)

Признаки: *загружаются из вектора прерывания*

Описание: *Команды с кодами операций от 104000 до 104377 являются командами EMT и могут быть использованы для передачи информации в моделирующую программу (т. е. информацию о функции, которая должна быть выполнена). Вектор прерывания для EMT находится по адресу 30. Новое содержимое PC берется из ячейки с адресом 30, а новое содержимое PSW - из ячейки с адресом 32.*

TRAP - программное прерывание

Код команды: *104400 ÷ 104777*

Действие:

↓(SP) ← (PSW)
↓(SP) ← (PC)
(PC) ← (34)
(PSW) ←(36)

Признаки: *загружаются из вектора прерывания*

Описание: *Команды с кодами операций от 104400 до 104777 являются командами TRAP, которые по своему действию идентичны командам EMT, за исключением того, что вектор прерывания команды TRAP имеет адрес 34.*

IOT - командное прерывание для ввода-вывода

Код команды: *000004*

Действие:

↓(SP) ← (PSW)
↓(SP) ← (PC)
(PC) ← (20)
(PSW) ←(22)

Признаки: *загружаются из вектора прерывания*

Описание: *Осуществляет прерывание с вектором прерывания, расположенным по адресу 20. Используется для обращения к подпрограмме управления вводом-выводом.*

BPT - командное прерывание для отладки

Код команды: *000003*

Действие:

↓(SP) ← (PSW)
↓(SP) ← (PC)
(PC) ← (14)
(PSW) ←(16)

Признаки: *загружаются из вектора прерывания*

Описание: *выполняется прерывание с вектором 14. Используется для обращения к подпрограммам отладки. Пользователю запрещается употребление кода 000003 в программах, которые выполняются под управлением подпрограмм отладки.*

RTI - возврат из прерывания

Код команды: *000002*

Действие:

(PC) ← (SP)↑
(PSW) ← (SP)↑

Признаки: *загружаются из стека*

Описание: *Используется для выхода из подпрограмм обслуживания внешних и внутренних прерываний. Содержимое PC и PSW восстанавливается с помощью стека. Если при выполнении этой команды будет установлен T-разряд PSW, то следующая после RTI команда выполняться не будет.*

RTT - возврат из прерывания

Код команды: *000006*

Действие:
(PC) ← (SP)↑
(PSW) ← (SP)↑

Признаки: *загружаются из стека*

Описание: *Эта команда по своему действию идентична команде RTI за исключением того, что при установке Т-разряда PSW прерывание будет иметь место после того, как выполнится первая команда, следующая за RTT.*

4.5. Специальные команды

HALT - останов

Код команды: *000000*

Действие:
↓(HSP) ← PSW
↓(HSP) ← PC
PC ← 0
PSW ← 340

Признаки: *не изменяются*

Описание: *Осуществляется установка режима "HALT". В специальном регистре организуется указатель стека HSP с начальным виртуальным значением 100000 и с его использованием в стек загружаются значения регистров PSW и PC. В регистр R7 загружается виртуальное значение 0. Примечание: в режиме пользователя выполнение команды HALT вызывает прерывание с вектором 10.*

WAIT - ожидание

Код команды: *000001*

Признаки: *не изменяются*

Описание: *Процессор ожидает незамаскированного им запроса на прерывание. Применение этой команды обеспечивает наиболее быструю передачу канала внешнему устройству при поступлении запроса от него. По команде WAIT процессору запрещается выбирать команды из памяти. Это позволяет наиболее быстро осуществить обмен между внешними устройствами и памятью, т. к. процессор не вносит задержки при обслуживании запроса канала на время, когда он освобождает занятый им канал. При выполнении команды WAIT, как и при выполнении всех других команд, в PC содержится адрес команды, следующей за командой WAIT. Таким образом, когда по прерыванию вызывается передача содержимого PC и PSW в стек, адрес команды, следующий за командой WAIT, сохраняется. Выход из подпрограммы, обслуживающей прерывание (т. е. выполнение команды RTI или RTT) вызовет возобновление прерванного процесса с команды, следующей за командой WAIT.*

RESET - сброс внешних устройств

Код команды: *000005*

Признаки: *не изменяются*

Описание: *по этой команде на выводе микропроцессора вырабатывается импульс длительностью в 1545 периодов тактовой частоты CLC. После импульса микропроцессор возобновит свою работу через время ожидания, равное 1545 периодам тактовой частоты CLC.*

Примечание: в моде пользователя команда выполняется как NOP.

MFPI - засылка инструкции в стек текущей моды по адресу предыдущей моды

Код команды: *0065SS*

Действие:

(temp) ← (src)
↓(SP) ← (temp)

Признаки:

N ← 1, если (src) < 0, иначе N ← 0
Z ← 1, если (src) = 0, иначе Z ← 0
V ← 0
C не изменяется

Описание: *Эта команда засылает слово в стек текущей моды по адресу, вычисленному в предыдущей моде. Виртуальный адрес источника вычисляется с помощью текущих регистров ДП.*

MFPD - засылка данных в стек текущей моды по адресу предыдущей моды

Код команды: *1065SS*

Описание: *Эта команда выполняется также как и команда MFPI.*

MTPI - засылка инструкции из стека текущей моды по адресу предыдущей моды

Код команды: *0066DD*

Действие:

(temp) ← (SP)↑
(dst) ← (temp)

Признаки:

N ← 1, если (dst) < 0, иначе N ← 0
Z ← 1, если (dst) = 0, иначе Z ← 0
V ← 0
C не изменяется

Описание: *Эта команда берет слово из стека текущей моды, определенного PSW (биты 15, 14) и засылает это слово по адресу, вычисленному в предыдущей моде. PSW (13, 12). Виртуальное значение адреса приемника вычисляется с помощью текущих регистров ДП.*

MTPD - засылка данных из стека текущей моды по адресу предыдущей моды

Код команды: *1066DD*

Описание: *Эта команда выполняется также как и команда MTPI*

4.6. Команды изменения признаков

Описание: *Разряды признаков, соответствующие установленным разрядам в команде изменения признаков (разряды 0-3), изменяются в соответствии с состоянием разряда 4 (разряда установки/сброса). Эти разряды PSW устанавливаются, если установлен четвертый разряд. Если же он очищен, то очищаются,*

Ниже перечисляются команды изменения признаков.

Обозначение	Операция	Код
CLN	*очистка N*	*000250*
CLZ	*очистка Z*	*000244*
CLV	*очистка V*	*000242*
CLC	*очистка C*	*000241*
CCC	*очистка всех разрядов (N, Z, V, C)*	*000257*
SEN	*установка N*	*000270*
SEZ	*установка Z*	*000264*
SEV	*установка V*	*000262*
SEC	*установка C*	*000261*
SCC	*установка всех разрядов (N, Z, V, C)*	*000277*
NOP	*нет операции*	*000240*

4.7. Команды расширенной арифметики

MUL - умножение

Код команды: *070RSS*

Действие: *(R, R ∨ 1) ← R * (src)*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 0
C ← 1, если результат меньше чем -2^{15} или больше чем $2^{15} - 1$

Описание: *Перемножаются операнды источника и приемника, взятые в двоично дополнительном коде. Результат помещается в регистр, используемый в качестве приемника, и в следующий за ним регистр, если регистр приемника имеет четный номер. Если же регистр приемника имеет нечетный номер, сохранятся только младшая часть результата.*

DIV - деление

Код команды: *071RSS*

Действие: *(R, R ∨ 1) ← (R, R ∨ 1) / (src)*

Признаки:

N ← 1, если частное < 0, иначе N ← 0
Z ← 1, если частное = 0, иначе Z ← 0
V ← 1, если (src) = 0, или если (dst) > (src) по абсолютной величине (в этом случае выполнение команды прекращается, т. к. частное будет превышать 15 разрядов);
C ← 1, если делитель = 0

Описание: *Тридцатидвухразрядное число в двоично-дополнительном коде, находящееся в регистрах R и R ∨ 1, делится на операнд источника. Частное заносится в R, а остаток - в R ∨ 1. После выполнения операции деления знак остатка будет таким же, как и у делимого. Следует заметить, что номер регистра R должен быть четным.*

ASH - арифметический сдвиг

Код команды: *072RSS*

Действие: *R ← R, сдвинутое на NN позиций влево или вправо, где NN - это 6 младших разрядов источника (src).*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если после выполнения операции сдвига операнд изменил знак, иначе V ← 0
C загружается содержимым последнего разряда, выдвинутого из регистра

Описание: *Содержимое указанного регистра сдвигается влево или вправо на количество позиций, определяемое счетчиком сдвига. Функцию счетчика сдвига выполняют шесть младших разрядов операнда источника. Его значение может изменяться в пределах от -32 до 31. Отрицательные значения счетчика определяет сдвиг вправо, положительное - влево.*

ASHC - арифметический сдвиг двойного слова

Код команды: *073RSS*

Действие: *(R, R ∨ 1) ← (R, R ∨ 1), сдвинутое на NN позиций влево или вправо, где NN - это 6 младших разрядов источника (src).*

Признаки:

N ← 1, если результат < 0, иначе N ← 0
Z ← 1, если результат = 0, иначе Z ← 0
V ← 1, если после выполнения операции сдвига операнд изменил знак, иначе V ← 0
C загружается содержимым выдвинутого из регистра 32-разрядного слова

Описание: *Содержимое регистров R и R ∨ 1 интерпретируется как одно тридцатидвухразрядное слово. Причем, младшая часть слова (разряды 00-15) содержится в R ∨ 1, а старшая часть (разряды 16-32) - в R. Тридцатидвухразрядное слово сдвигается вправо или влево на количество позиций, определяемом счетчиком сдвига. Функцию счетчика сдвига выполняют шесть младших разрядов операнда источника. Его значение может изменяться в пределах от -32 до 31. Отрицательное значение счетчика определяет сдвиг вправо, положительное - влево. Если выбранный регистр имеет нечетный номер, то R и R ∨ 1 являются одним и тем же регистром. В этом случае сдвиг вправо будет выполняться циклически (шестнадцатиразрядное слово сдвигается циклически на количество позиций, определяемое счетчиком сдвига).*

5. Система команд процессора 1836ВМ3

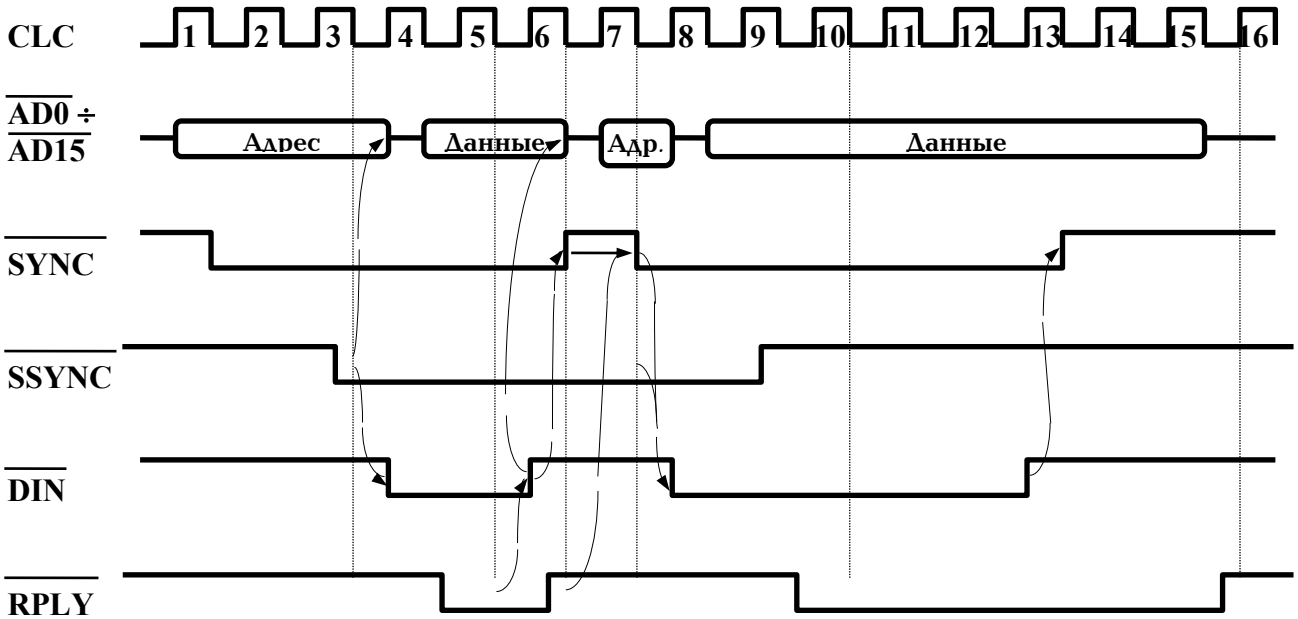
Мнемоника	Код	Наименование команды
HALT	*000000*	*Останов*
WAIT	*000001*	*Ожидание*
RTT	*000002*	*Возврат из прерывания*
BPT	*000003*	*Командное прерывание для отладки*
IOT	*000004*	*Командное прерывание для ввода/вывода*
RESET	*000005*	*Сброс внешних устройств*
RTT	*000006*	*Возврат из прерывания*
JMP	*0001DD*	*Безусловный переход*
RTS	*00020R*	*Возврат из подпрограммы*
JSR	*004RDD*	*Переход к подпрограмме*
EMT	*104000-104377*	*Командное прерывание для системных программ*
TRAP	*104400-104777*	*Командное прерывание*
NOP	*000240*	*Нет операции*
CLC	*000241*	*Очистка флага C*
CLV	*000242*	*Очистка флага V*
CLZ	*000244*	*Очистка флага Z*
CLN	*000250*	*Очистка флага N*
SEC	*000261*	*Установка флага C*
SEV	*000262*	*Установка флага V*
SEZ	*000264*	*Установка флага Z*
SEN	*000270*	*Установка флага N*
SCC	*000277*	*Установка всех разрядов (N,Z,V,C)*
CCC	*000257*	*Очистка всех разрядов (N,Z,V,C)*
SWAB	*0003DD*	*Перестановка байтов*
CLR(B)	**050DD*	*Очистка*
COM(B)	**051DD*	*Инвертирование*
INC(B)	**052DD*	*Прибавление единицы*
DEC(B)	**053DD*	*Вычитание единицы*
NEG(B)	**054DD*	*Изменение знака*
ADC(B)	**055DD*	*Прибавление переноса*
SBC(B)	**056DD*	*Вычитание переноса*
TST(B)	**057DD*	*Проверка*
ROR(B)	**060DD*	*Циклический сдвиг вправо*
ROL(B)	**061DD*	*Циклический сдвиг влево*
ASR(B)	**062DD*	*Арифметический сдвиг вправо*
ASL(B)	**063DD*	*Арифметический сдвиг влево*
MARK	*0064NN*	*Восстановление SP*
SXT	*0067DD*	*Расширение знака*
MTPS	*1064SS*	*Запись PSW*
MFPS	*1067DD*	*Чтение PSW*
MOV(B)	**1SSDD*	*Пересылка*
CMP(B)	**2SSDD*	*Сравнение*
BIT(B)	**3SSDD*	*Проверка разрядов*
BIC(B)	**4SSDD*	*Очистка разрядов*
BIS(B)	**5SSDD*	*Установка разрядов*
XOR	*074RDD*	*Исключающее ИЛИ*
ADD	*06SSDD*	*Сложение*
SUB	*16SSDD*	*Вычитание*
BR	*0004XX*	*Ветвление безусловное*
BNE	*0010XX*	*Ветвление, если не равно (нулю)*
BEQ	*0014XX*	*Ветвление, если равно (нулю)*
BGE	*0020XX*	*Ветвление, если больше или равно (нулю)*
BLT	*0024XX*	*Ветвление, если меньше (нулю)*

Мнемоника	Код	Наименование команды
BGT	*0030XX*	*Ветвление, если больше (нулю)*
BLE	*0034XX*	*Ветвление, если меньше или равно (нулю)*
SOB	*077RNN*	*Вычитание единицы и ветвление*
BPL	*1000XX*	*Ветвление, если плюс*
BMI	*1004XX*	*Ветвление, если минус*
BHI	*1010XX*	*Ветвление, если больше*
BLOS	*1014XX*	*Ветвление, если меньше или равно*
BVC	*1020XX*	*Ветвление, если нет переполнения*
BVS	*1024XX*	*Ветвление, если есть переполнение*
BHIS,BCC	*1030XX*	*Ветвление, если больше или равно*
BLO,BCS	*1034XX*	*Ветвление, если меньше*
MUL	*070RSS*	*Умножение*
DIV	*071RSS*	*Деление*
ASH	*072RSS*	*Сдвиг на N разрядов одного слова*
ASHC	*073RSS*	*Сдвиг на N разрядов двойного слова*
MFPD	*1065SS*	*Засылка слова (D-данные, I-инструкция) в стек*
MFPI	*0065SS*	*текущей моды по адресу предварительной моды*
MTPD	*1066DD*	*Засылка слова из стека текущей моды по адресу*
MTPI	*0066DD*	*предварительной моды*
	X07XXX *075XXX* *076XXX* *0000(4-7)X* *00(0,2)(2,3)1X* *000(0,2)1X* *000007*	*Резервные коды команд*

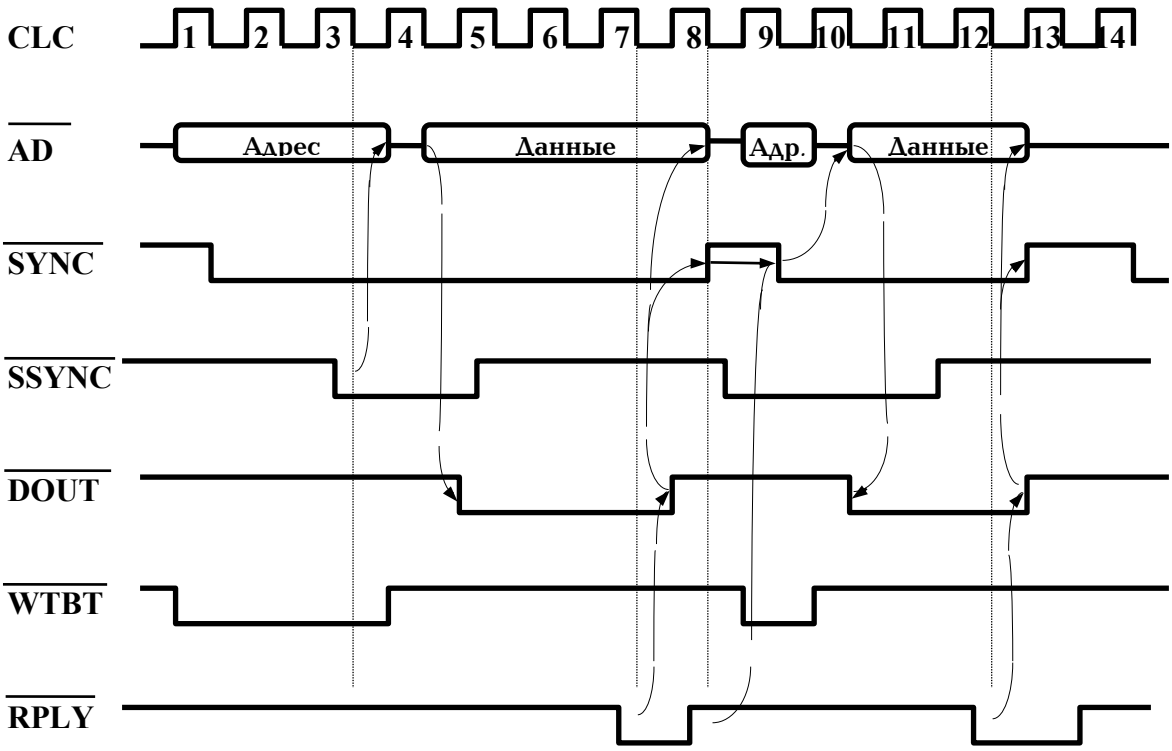
SS - поле адреса операнда источника;
DD - поле адреса операнда приемника;
XX - смещение (8 разрядов);
NN - смещение (6 разрядов);
Знак "" имеет значение "0" для команд с полными словами и "1" - для байтовых команд.*

6. Временные диаграммы обмена

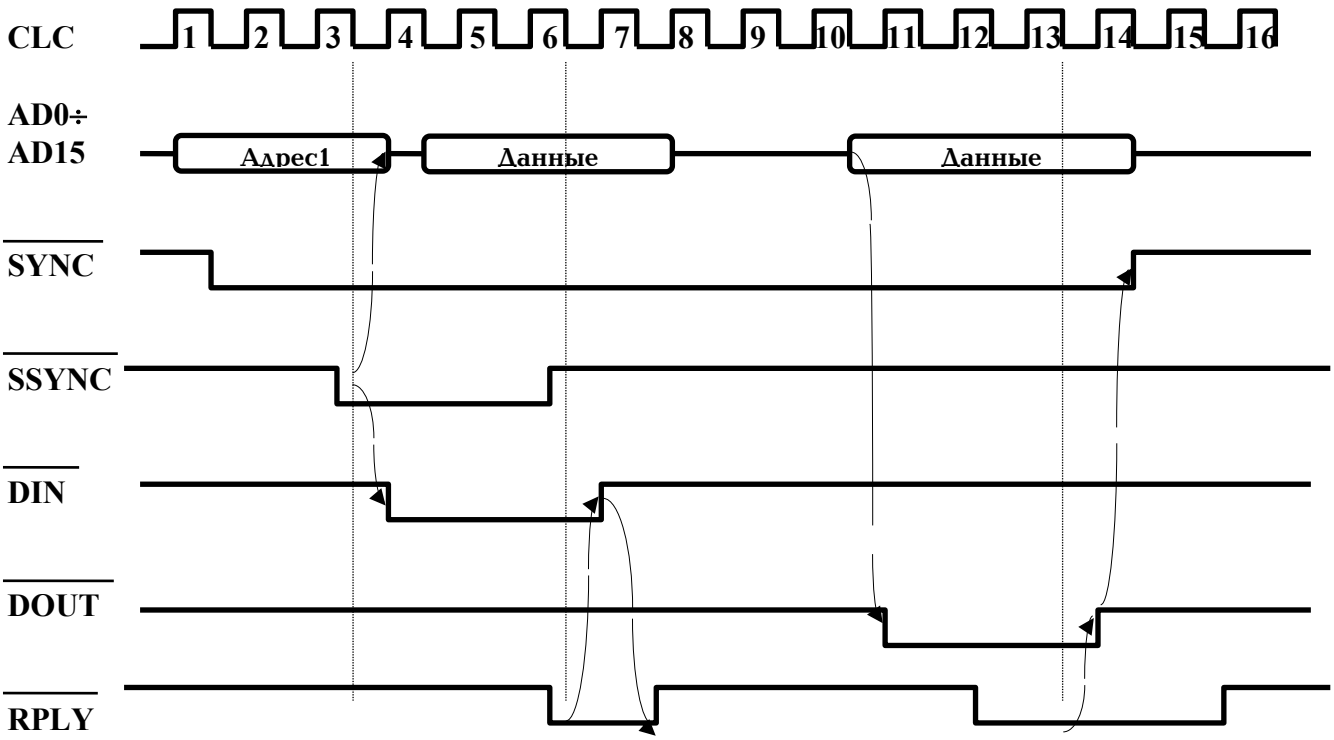
6.1. Временная диаграмма чтения данных по адресу



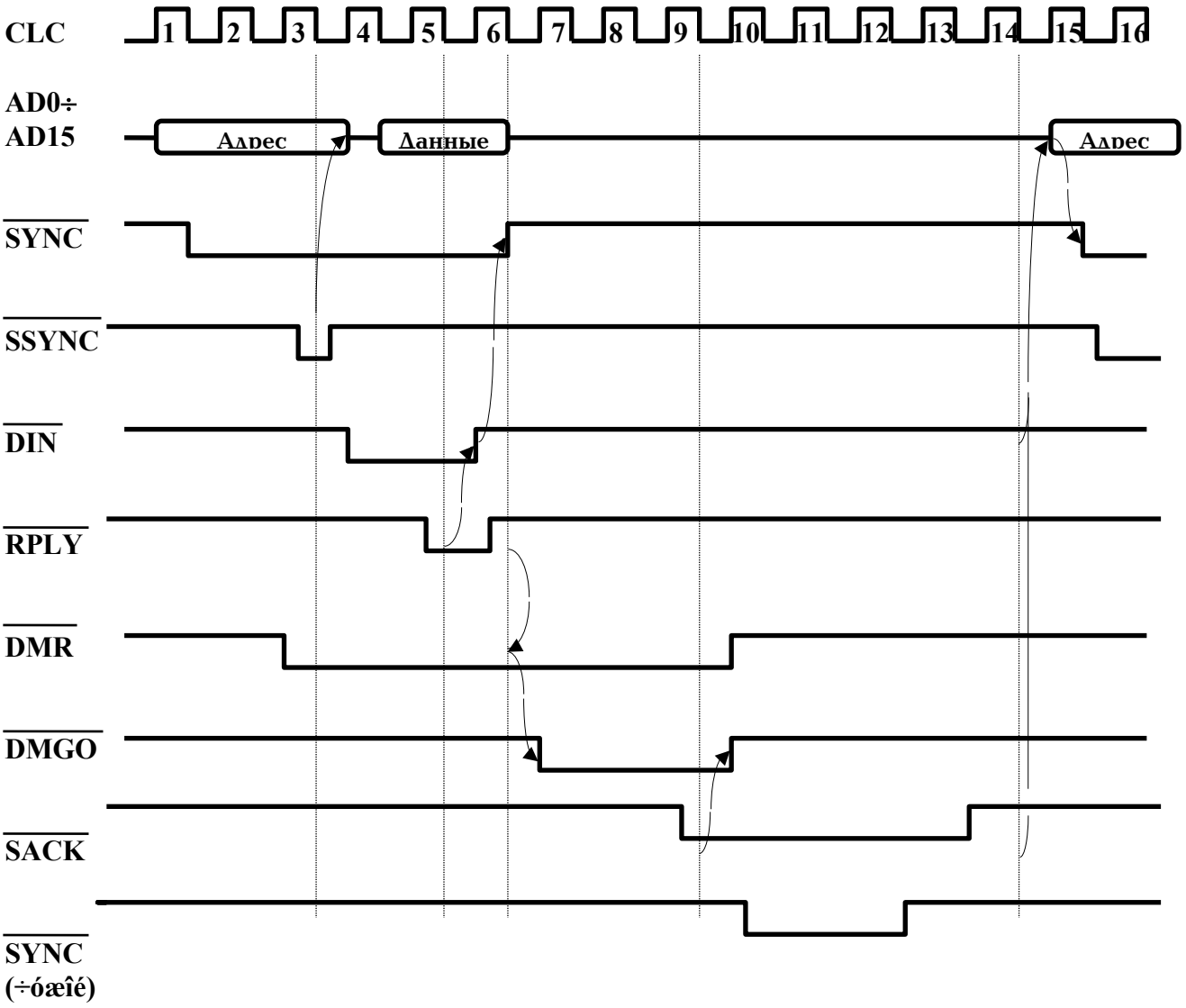
6.2. Временная диаграмма записи данных по адресу



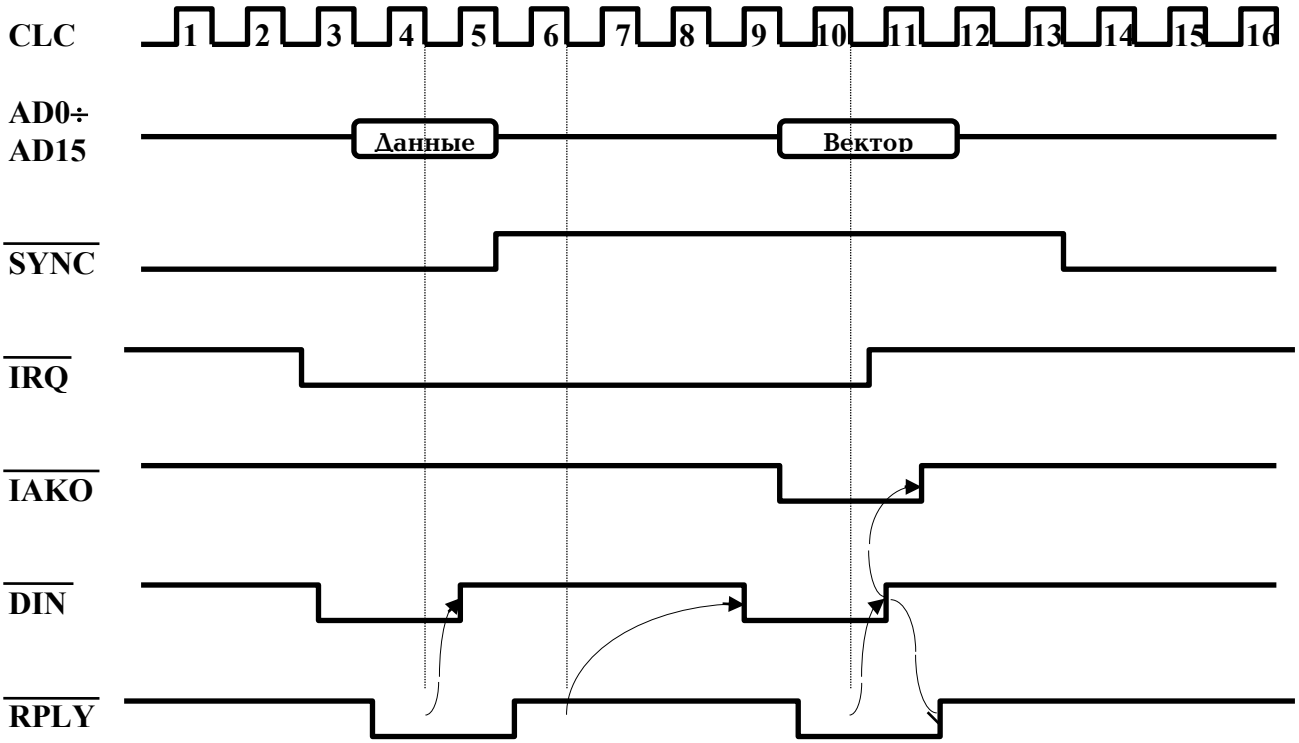
6.3. Временная диаграмма чтение-модификация-запись



6.4. Временная диаграмма захвата магистрали



6.5. Временная диаграмма чтения вектора прерывания



7. Условное графическое обозначение микросхемы 1836ВМ3

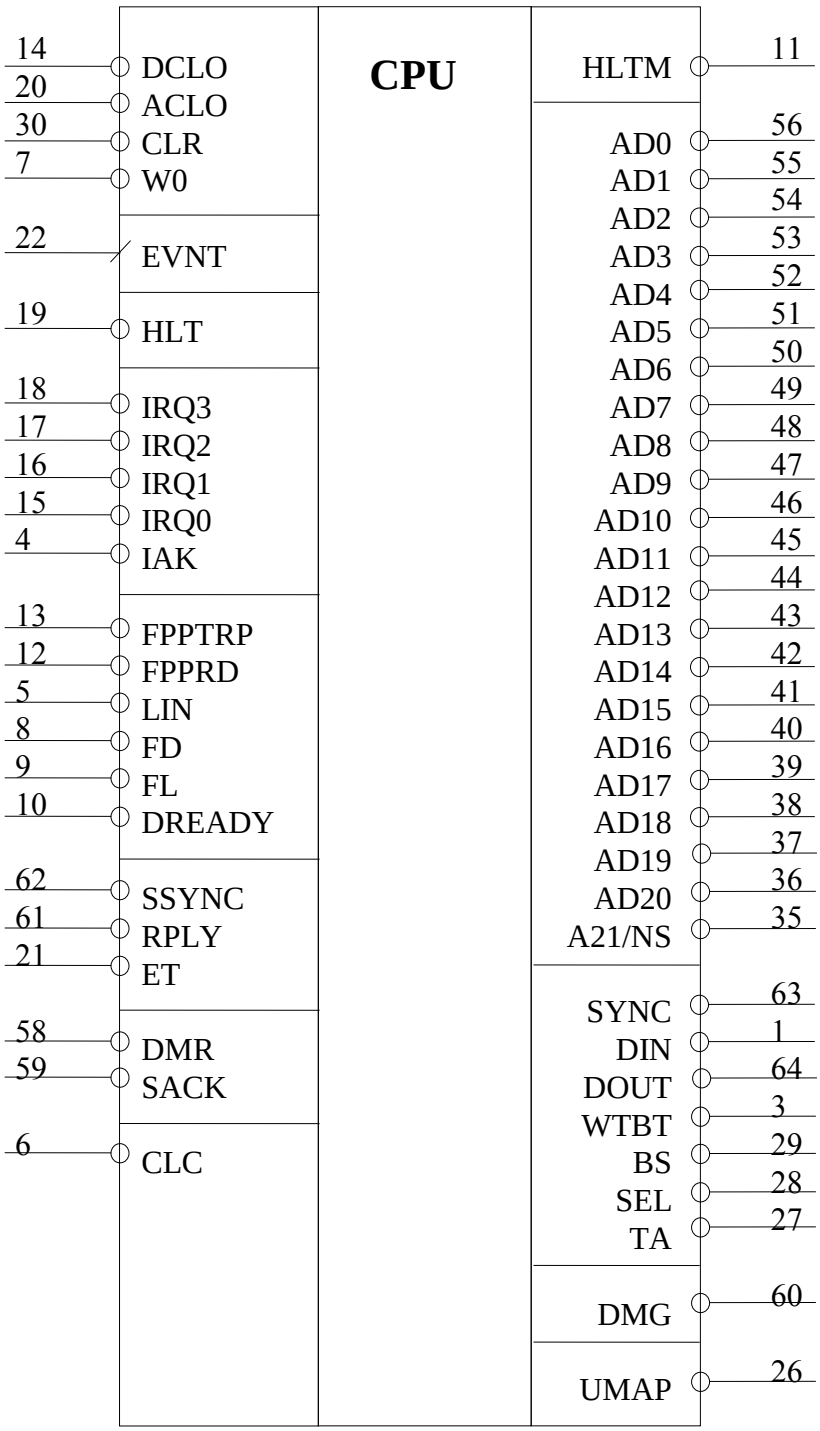


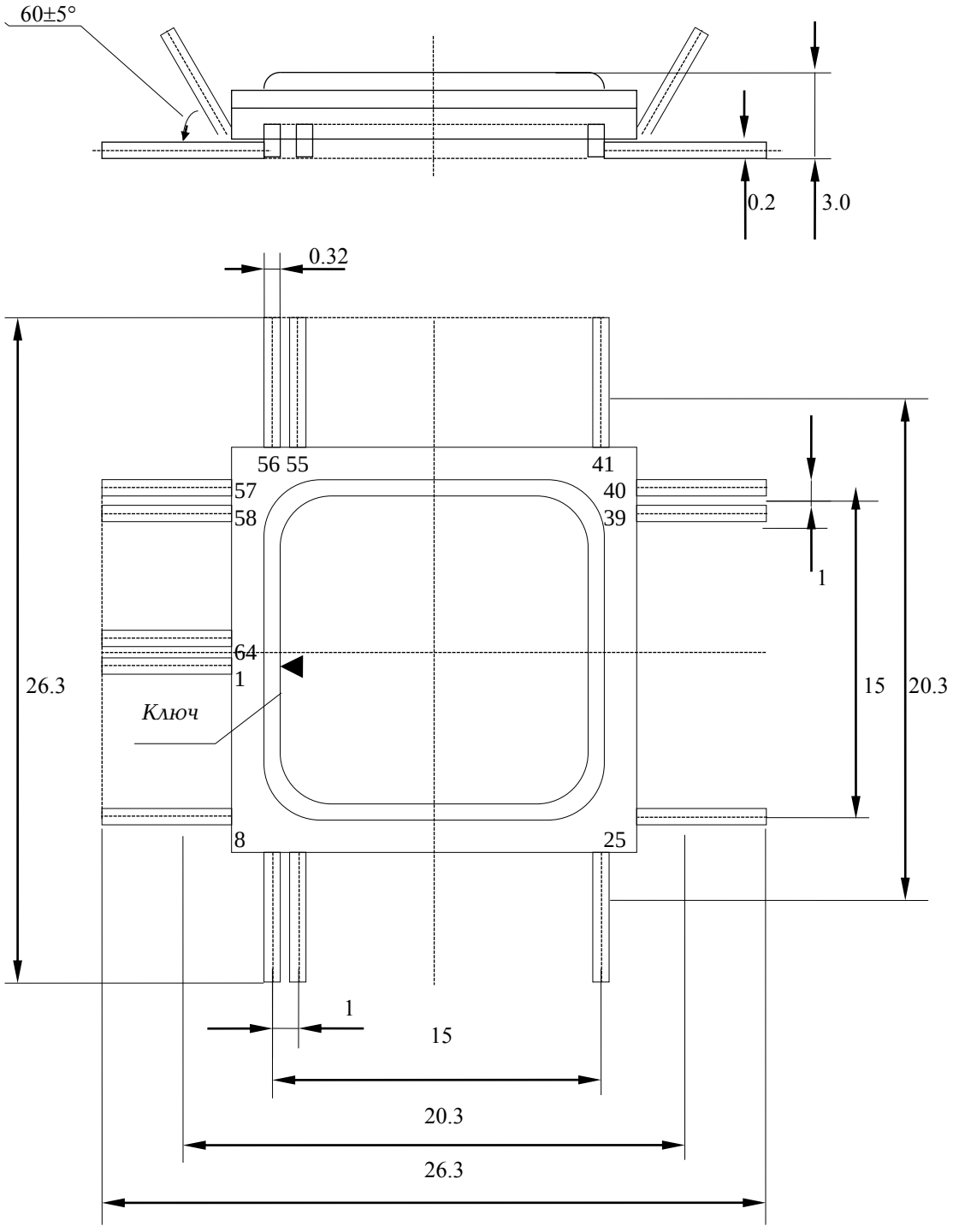
Таблица 7.1. Назначение выводов

Номер	Обозначение	Наименование
1	*DIN*	*Сигнал управления вводом данных*
2	*U*	*Вывод питания от источника напряжения*
3	*WTBT*	*Сигнал управления запись-байт*
4	*IAK*	*Сигнал разрешения запроса на прерывание*
5	*LIN*	*Сигнал загрузки команды*
6	*CLC*	*Тактовый импульс*
7	*WO*	*Сигнал включения*
8	*FD*	*Признак двойной точности*
9	*FL*	*Признак длинного целого*
10	*DREADY*	*Сигнал готовности данных*
11	*HLTM*	*Сигнал отладочного режима*
12	*FPPRD*	*Сигнал готовности*
13	*FPPTRP*	*Сигнал прерывания*
14	*DLCO*	*Сигнал источника питания постоянного напряжения*
15	*IRQ0*	*Сигнал запроса на прерывание с приоритетом 4*
16	*IRQ1*	*Сигнал запроса на прерывание с приоритетом 5*
17	*IRQ2*	*Сигнал запроса на прерывание с приоритетом 6*
18	*IRQ3*	*Сигнал запроса на прерывание с приоритетом 7*
19	*HLT*	*Сигнал останова*
20	*ACLO*	*Сигнал источника питания переменного напряжения*
21	*ET*	*Сигнал разрешения зависания*
22	*EVNT*	*Сигнал радиального прерывания*
23	*U*	*Вывод питания от источника напряжения*
24-25	*0V*	*Общий вывод*
26	*UMAP*	*Сигнал разрешения преобразования адресов UNIBUS*
27	*TA*	*Сигнал выдачи адреса*
28	*SEL*	*Сигнал выборки при HALT-моде*
29	*BS*	*Сигнал обращения к банку внешних устройств*
30	*CLR*	*Сигнал установки внешних устройств*
31	*0V*	*Обший вывод*
32-34	*SP*	*Свободный вывод*
35	*A21/NS*	*Сигнал адреса-инструкции*
36	*A20*	*Двадцатый разряд адреса системной магистрали*
37	*A19*	*Девятнадцатый разряд адреса системной магистрали*
38	*A18*	*Восемнадцатый разряд адреса системной магистрали*
39	*A17*	*Семнадцатый разряд адреса системной магистрали*
40	*A16*	*Шестнадцатый разряд адреса системной магистрали*
41	*AD15*	*Пятнадцатый разряд адреса-данных системной магистрали*
42	*AD14*	*Четырнадцатый разряд адреса-данных системной магистрали*
43	*AD13*	*Тринадцатый разряд адреса-данных системной магистрали*
44	*AD12*	*Двенадцатый разряд адреса-данных системной магистрали*
45	*AD11*	*Одиннадцатый разряд адреса-данных системной магистрали*
46	*AD10*	*Десятый разряд адреса-данных системной магистрали*
47	*AD9*	*Девятый разряд адреса-данных системной магистрали*
48	*AD8*	*Восьмой разряд адреса-данных системной магистрали*
49	*AD7*	*Седьмой разряд адреса-данных системной магистрали*
50	*AD6*	*Шестой разряд адреса-данных системной магистрали*
51	*AD5*	*Пятый разряд адреса-данных системной магистрали*

Номер	Обозначение	Наименование
52	*AD4*	*Четвертый разряд адреса-данных системной магистрали*
53	*AD3*	*Третий разряд адреса-данных системной магистрали*
54	*AD2*	*Второй разряд адреса-данных системной магистрали*
55	*AD1*	*Первый разряд адреса-данных системной магистрали*
56	*AD0*	*Нулевой разряд адреса-данных системной магистрали*
57	*0V*	*Общий вывод*
58	*DMR*	*Сигнал запроса прямого доступа к памяти*
59	*SACK*	*Сигнал подтверждения запроса прямого доступа к памяти*
60	*DMG*	*Сигнал разрешения прямого доступа к памяти*
61	*RPLY*	*Сигнал ответа приемника информации*
62	*SSYNC*	*Сигнал синхронизации устройства*
63	*SYNC*	*Сигнал синхронизации обмена*
64	*DOUT*	*Сигнал управления выводом данных*

8. Габаритный чертеж микросхемы 1836ВМ3

(Металлокерамический корпус типа Н18.64 - 2В)



Для контактов:

АО ИТТиП
103460 Москва, Зеленоград, Институт Точной Технологии и Проектирования
тел/факс: (095) 532-95-50, (095) 531-44-20
Генеральный директор: Машевич П.Р. тел: (095) 532-07-02