# Процессорное ядро M8

**Назначение**

Процессор M8 является вычислительным ядром для семейства микроконтроллеров M8X.

Основные характеристики процессора приведены в таблице 1.

Таблица 1

| Наименование | Параметры |
|---|---|
| Разрядность данных | 8-32 бит |
| Длина команды | 1-2 байта |
| Производительность | 1 такт на байт команды |
| ОЗУ | 64 Кбайт |
| ПЗУ | 64 Кбайт |
| Количество прерываний | 16 |
| Количество регистров внешних устройств | 64 |

Процессор M8 отличает:

а) возможность обработки данных длиной 8, 16 и 32 разряда;

б) байтовый формат команд с безадресными стековыми командами обработки данных, выполняющихся за один такт. Имеются префиксные команды, которые дополняют информационное поле команды до 8 и более разрядов, а также расширяют безадресные стековые команды до команд обработки данных в памяти;

в) страничная адресация памяти с возможность косвенной, индексной, инкрементной и декрементной адресации, а также прямой адресации регистров ВУ;

г) высокая эффективность вызова программ. Процессор имеет два стека - один для обработки данных и передачи параметров вызываемой программе, другой для хранения адресов возврата;

д) ориентация на логическую обработку данных. Система команд содержит команды сравнения, тестирования и ветвления, битовой обработки данных в памяти и организации циклов;

ж) двухуровневая система прерываний, которая обеспечивает гарантированное обслуживание прерываний независимо от состояния процессора;

з) низкое потребление за счет работы на пониженных частотах от часового кварца или встроенного RC генератора с возможностью умножения частоты;

и) поддержка прямого доступа к памяти, что обеспечивает передачу информации между источником и приемником данных (ОЗУ или ВУ) без участия процессора;

к) наличие внутрисхемного отладчика, который обеспечивает отладку программного кода процессора в составе целевой системы, а также фиксацию нештатных ситуаций в процессе эксплуатации.

**Состав**

Процессор M8 содержит АЛУ, шины ОЗУ, ПЗУ, ВУ, стек данных, стек возвратов и специальные регистры.

**Организация ПЗУ**

ПЗУ предназначено для хранения программ и постоянных данных. Структура ПЗУ приведена на рисунке 2. ПЗУ имеет байтовую организацию объемом до 64 Кбайт и разделено на 16 банков по 4 Кбайт.

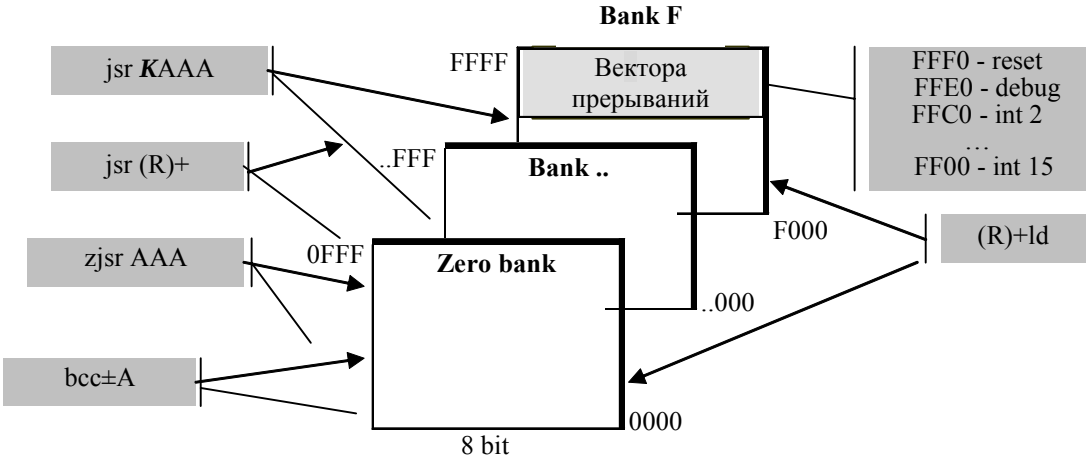


Рисунок 2

Доступ к программам внутри банка осуществляется с помощью команд вызова программ. Переходы к программам из банка в банк сопровождаются префиксом константы, значение которой равно номеру банка. Из каждого банка можно вызвать программы, расположенные в нулевом банке. Кроме этого имеется возможность косвенного вызова программ по указателю в стеке возвратов.

Команды проверки и ветвления обеспечивают безусловную и условную передачу управления по 6-разрядному смещению относительно текущего значения счетчика команд. Префикс константы перед командой передачи управления позволяет увеличить смещение до 10 и 14 разрядов.

В старших адресах ПЗУ содержится область 16 программ обработки прерываний, для каждой из которых выделен участок памяти в 16 байт.

Доступ к данным, хранящимся в ПЗУ, осуществляется путем косвенной адресации через указатель в стеке возвратов.

**Организация памяти данных**

Память данных процессора включает в себя оперативную память (ОЗУ) и регистры внешних устройств (ВУ). Структура памяти процессора приведена на рисунке 3.

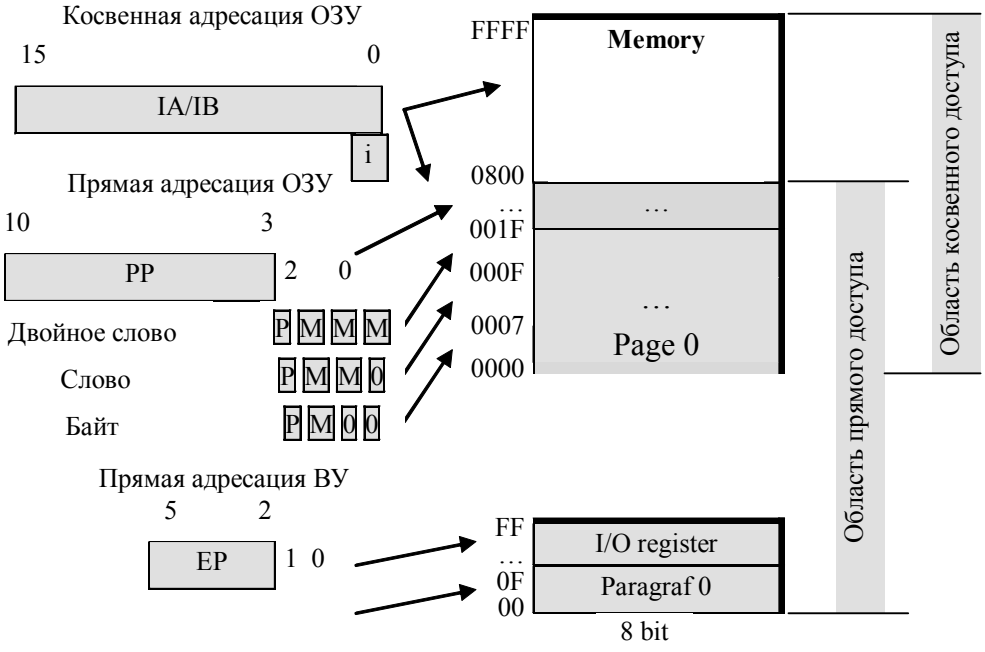


Рисунок 3

ОЗУ имеет объем до 64 Кбайт. Доступ к ОЗУ осуществляется в режиме прямой и косвенной адресации. В режиме прямой адресации доступны младшие 4 Кбайт ОЗУ, в режиме косвенной адресации доступно все ОЗУ.

Прямая адресация ОЗУ имеет страничную организацию. Длина страницы составляет 8 байт. Данные могут иметь длину 8, 16 и 32 бита (байт, слово и двойное слово). Данные длиной 16 и 32 бит выравнены, соответственно, по четным и $2^2$ адресам. Доступ к ОЗУ осуществляется с

использованием номера страницы и элемента внутри страницы. Одновременно доступны две страницы – текущая и предыдущая. Номер страницы хранится в 8-разрядном поле Pp (**P**age pointer) регистра состояния программы RS. Поле Pp устанавливается специальной командой или задается префиксом константы в командах обращения к памяти, подменяя соответствующие разряды поля номера страницы. Номер элемента внутри страницы задается адресным полем команды доступа к памяти. В таблице 2 приведены способы вычисления адреса данного при прямой адресации ОЗУ.

Косвенная адресация ОЗУ осуществляется через два 16-разрядных регистра IA, IB (**I**ndex registers). Содержимое регистров устанавливается специальной командой. Префикс константы задает смещение адресуемого данного относительно индексного регистра. Имеются две префиксные команды автомодификации, которые задают пре-декремент и пост-инкремент индексного регистра. Данные могут иметь длину 8, 16 и 32 бита (байт, слово и двойное слово). Автомодификация индексного регистра осуществляется с учетом длины данного.

Адресация регистров ВУ осуществляется в режиме прямой адресации предыдущей страницы ОЗУ при установленном признаке E регистра состояния программы RS с использованием номера страницы, которое хранится в 4-разрядном поле EP (**E**xternal **P**age) регистра состояния программы RS. Номер страницы EP устанавливается специальной командой или задается префиксом константы. Номер регистра ВУ внутри страницы задается адресным полем команды доступа к памяти к текущей странице ОЗУ. Регистры ВУ могут иметь длину 8, 16 и 32 бита (байт, слово и двойное слово). В таблице 2 приведены способы вычисления адреса ВУ.

Таблица 2

| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Адрес | Данное |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Текущая страница | | | | | | | | | | | | | |
| 0 | P4..7/КК | | | P0..3/K | | | | | M | M | M | P4..7/KK*$2^7$+ P0..3/K*$2^3$+M | байт |
| 0 | P4..7/КК | | | P0..3/K | | | | | M | M | 0 | P4..7/KK*$2^7$+ P0..3/K*$2^3$+M*$2^1$ | слово |
| 0 | P4..7/КК | | | P0..3/K | | | | | M | 0 | 0 | P4..7/KK*$2^7$+ P0..3/K*$2^3$+M*$2^2$ | двойное слово |
| Предыдущая страница | | | | | | | | | | | | | |
| 0 | P0..7-1 | | | | | | | | M | M | M | (P0..7-1)*$2^3$+M | байт |
| 0 | P0..7-1 | | | | | | | | M | M | 0 | (P0..7-1)*$2^3$+M*$2^1$ | слово |
| 0 | P0..7-1 | | | | | | | | M | 0 | 0 | (P0..7-1)*$2^3$+M*$2^2$ | двойное слово |
| Предыдущая страница с префиксом | | | | | | | | | | | | | |
| 0 | P4..7-1 | | | K | | | | | M | M | M | (P4..7-1)*$2^3$+K*$2^3$+M | байт |
| 0 | P4..7-1 | | | K | | | | | M | M | 0 | (P4..7-1)*$2^3$+K*$2^3$+M*$2^1$ | слово |
| 0 | P4..7-1 | | | K | | | | | M | 0 | 0 | (P4..7-1)*$2^3$+K*$2^3$+M*$2^2$ | двойное слово |
| Предыдущая страница с двойным префиксом | | | | | | | | | | | | | |
| 1 | КК | | | K | | | | | M | M | M | $2^{11}$+ KK*$2^7$+K*$2^3$+M | байт |
| 1 | КК | | | K | | | | | M | M | 0 | $2^{11}$+ KK*$2^7$+K*$2^3$+M*$2^1$ | слово |
| 1 | КК | | | K | | | | | M | 0 | 0 | $2^{11}$+ KK*$2^7$+K*$2^3$+M*$2^2$ | двойное слово |
| Адресация ВУ | | | | | | | | | | | | | |
| | 0 | | | EP/K | | | | | M | M | M | EP/K*$2^3$+M | байт |
| | 0 | | | EP/K | | | | | M | M | 0 | EP/K*$2^3$+M*$2^1$ | слово |
| | 0 | | | EP/K | | | | | M | 0 | 0 | EP/K*$2^3$+M*$2^2$ | двойное слово |

Примечания.
1. M - номер элемента внутри страницы
2. Pi..j - разряды поля указателя страницы
3. K, КК - одинарный и двойной префикс константы

**Стек данных**

Стек данных S (data **S**tack) имеет байтовую организацию и реализован аппаратно. Операции обработки данных выполняются над верхними элементами стека. Попытка выборки из пустого стека и записи в заполненный стек вызывают внутренний сигнал прерывания.

На рисунке 4 приведена структура стека вычислений и даны примеры 16-разрядных операций работы со стеком. Верхушка стека имеет обозначение S, следующие элементы стека - S1.. и т.д.

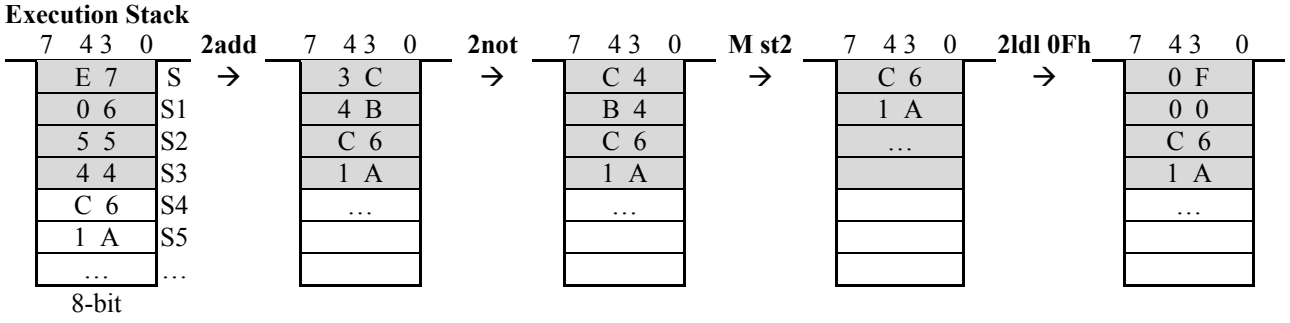


Рисунок 4

**Стек возвратов**

Стек возвратов R (**R**eturn stack) имеет 16-разрядую организацию и реализован аппаратно.

Стек возвратов используется для хранения адреса возврата при вызове программ, хранения регистров процессора и данных. Верхушка стека возвратов используется также как указатель для косвенного чтения данных из ПЗУ и вызова программ.

Команды процессора обеспечивают передачу данных между верхушкой стека возвратов, стека данных и регистрами процессора. Попытка выборки из пустого стека и записи в заполненный стек вызывают внутренний сигнал прерывания.

На рисунке 5 приведена структура стека возвратов и даны примеры типовых операций работы со стеком.

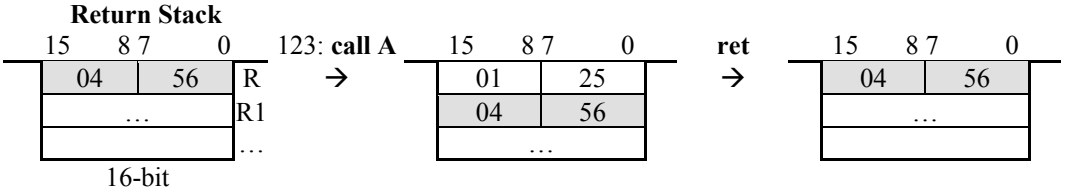


Рисунок 5

**Регистры процессора**

Регистры процессора M8 приведены в таблице 3.

Таблица 3

| 15 12 11 8 7 4 3 2 1 0 | Наименование регистра |
|---|---|
| PC | **P**rogram **C**aunter |
| I E EP Pp | **R**egister **S**tatus of **P**rogram |
| IA | **I**ndex register A |
| IB | **I**ndex register B |

16-разрядный счетчик команд PC (**P**rogram **C**ounter) содержит адрес очередной команды. Для линейных участков кода счетчик команд инкрементируется после каждой выборки байта команды. В командах ветвления, вызова программ, возврата в вызывающую программу и обработке прерываний в счетчик команд загружается новый адрес.

Регистр состояния программы RS (**R**egister of program **S**tatus) объединяет три регистра:

- регистр указателя текущей страницы ОЗУ Pp (Page pointer);
- регистр EP (External Page pointer), содержащий номер страницы регистров ВУ;
- регистр флагов состояния программы:
  - флаг запрета прерывания I (**I**nterrupt) запрещает обработку маскируемых прерываний;
  - флаг доступа E (**Extend**) к регистрам внешних устройств.

Индексные регистры IA, IB (**I**ndex **R**egisters) используются для косвенной адресации данных в ОЗУ. Содержимое регистров устанавливается специальной командой.

**АЛУ**

АЛУ процессора M8 обеспечивает выполнение арифметических, логических и сдвиговых операций над 8, 16 и 32-разрядными данными.

**Система команд**

Процессор M8 имеет байтовую систему команд переменной длины, выполняющихся за один такт на байт команды.

Команды процессора включают в себя команды обработки данных на стеке, обработки данных с константой, загрузки и выгрузки стека, организации управления ходом выполнения программы и специальные команды.

В состав команд процессора входят префиксные команды:

а) префикс константы расширяет информационное поле команды до 8 и более разрядов;

б) префикс операнда задает операнд в памяти для команд обработки данных;

в) префикс модификации регистра косвенной адресации задает предекремент или постинкремент индексных регистров при косвенной адресации данных.

**Типы данных**

Процессор M8 позволяет обрабатывать данные длиной 1, 2 и 4 байта.

Данные длиной более байта в стеке размещены с младшего байта и далее в глубь стека, в памяти - с младшего байта и далее в сторону старших адресов. Данные длиной 2 и 4 байта выравнены, соответственно, по четным и $2^2$ адресам.

Обработку 16 и 32-разрядных данных процессор выполняет потактно, начиная с младших байт, за исключением операции загрузки данных в стек и сдвигов вправо, которые выполняются, начиная со старшего байта данного.

В командах сравнения байты сравниваются как беззнаковые целые числа, слова (два байта) – как знаковые, так и беззнаковые, двойные слова (4 байта) – как знаковые целые числа.

В таблице приведены характеристики данных процессора M8.

Таблица

| Байт | Тип | Диапазон значений |
|---|---|---|
| 1 | целое без знака | от 0 до 255 |
| 2 | целое без знака | от 0 до 65535 |
| 2 | целое со знаком | от -32768 до 32767 |
| 4 | целое со знаком | от -2147483648 до 2147483647 |

**Обработка прерываний**

Процессор M8 поддерживает обработку 14 сигналов внешних прерываний и одного внутреннего.

Внешние прерывания могут быть маскируемыми и не маскируемыми. Обработка маскируемых прерываний запрещается установкой флага I регистра состояния программы в 1. Немаскируемые прерывания вызывают прерывание независимо от состояния флага I. Имеются специальные команды установки и сброса флага I.

Внутреннее прерывание не маскируется и не прерывается никакими другими прерываниями. Внутреннее прерывание формируется при обнаружении ошибочной ситуации, в том числе, переполнении и выборки из пустого стека, а также внутрисхемным отладчиком.

С каждым сигналом прерывания связан свой вектор, который определяет 16-байтовую область программного кода - обработчика прерывания, расположенных в старших адресах ПЗУ. В таблице 4 приведено распределение векторов прерывания. Нулевой вектор содержит программу обработки сигнала Reset.

Таблица 4

| Номер | Адрес | Источник |
|---|---|---|
| 0 | FFF0 | Сигнал Reset |
| 1 | FFE0 | Встроенный отладчик и ошибочные ситуации |
| 2 | FFD0 | Немаскируемое внешнее прерывание INT2 |
| 3 | FFC0 | Маскируемое внешнее прерывание INT3 |
| … | … | … |
| 14 | FF10 | Маскируемое внешнее прерывание INTE |
| 15 | FF00 | Немаскируемое внешнее прерывание INTF |

При прерывании содержимое регистров счетчика команд и состояния программы сохраняется в стеке возвратов и управление передается программе обработки прерывания в соответствии с номером прерывания. Выход из прерывания с восстановлением содержимого регистров состояния программы и счетчика команд из стека возвратов осуществляется по команде RETI.

**Организация конвейера**

Процессор M8 имеет 2-ступенчатый конвейер. Ступень конвейера выполняется за один такт и содержит 2 стадии.

На первой ступени конвейера выполняются FD-стадии (**F**etch and **D**ecode) - выборка и дешифрация команды. На стадии F по содержимому счетчика команд осуществляется выборка команды из ПЗУ. На стадии D осуществляется предварительная дешифрация команды. На этой ступени выполняется обработка прерывания, а в одно байтовых командах управления выполнением программ устанавливается новый адрес выборки команды.

На второй ступени конвейера, в зависимости от команды, выполняются стадии:

- RE-стадия (**R**ead **m**emory or **E**xecute) - чтение из памяти или выполнение операции АЛУ;
- W-стадия (**W**rite **M**emory) - запись в память.

**Управление энергопотреблением**

В процессоре M8 предусмотрены специальные меры по снижению энергопотребления - управление подачей тактовой частоты, а также питания на устройства процессора.

Процессор содержит модуль генератора, который обеспечивает формирование тактовой частоты от внешнего сигнала или встроенного генератора. Имеется возможность умножения частоты с помощью PLL. В целях снижения энергопотребления тактовые сигналы подаются только на те устройства процессора, которые участвуют в выполняемой процессором операции, вплоть до снятия частоты с самого процессора и перевода его в режим SLEEP. Выход из режима SLEEP происходит по сигналу прерывания.

**Устройство прямого доступа к памяти**

Процессор M8 содержит встроенное устройство прямого доступа к памяти.

Преимущество прямого доступа к памяти заключается в возможности передачи информации между источником и приемником данных без участия процессора. Роль процессора при этом сводится к установке адресов источника и приемника данных и объема (количества) передаваемых данных.

В качестве источника и приемника данных могут выступать ОЗУ и регистры ВУ.

**Внутрисхемный отладчик**

Процессор M8 обеспечивает отладку программного кода процессора непосредственно в составе системы. За счет этого достигается полная идентичность временных и электрических параметров работы системы, как в отладочном, так и в рабочем режимах.

Устройство внутрисхемной отладки содержит управляемый компаратор состояния внутренних шин процессора, формирующий внутреннее отладочное прерывание, буфер трассы, содержащий значения счетчика последних команд передач управления, и интерфейсный блок, который обеспечивает полудуплексный обмен данными по отладочному интерфейсу.

**Описание системы команд**

**Форматы команд**

Формат команды определяется старшими разрядами командного байта. Младшие разряды содержат информационное поле команды. Форматы базовых команд приведены в таблице 5. Кодировка базовых команд приведена в таблице 8.

Таблица 5

| 7 4 3 0 | Мнемоника | Наименование |
|---|---|---|
| 0 0 0 0 K | ***K.***K | Префикс константы |
| 1 0 0 0 p M M M<br>1 0 0 0 i | ***K.p***M<br>K. IA/IB | Префикс операнда |
| bccw<br>0/1 +/- A A | ***K***AA bccw | Проверка и ветвление |
| opw | *K* ***Md*** opw | Операция обработки данных |
| oplw<br>Lw | *K* ***Md*** oplw Lw | Операция обработки данных с константой |
| op | op | Операция |
| 1 0 w w p M M M<br>1 0 w w i | ***K p***M ldw<br>***K IA/IB*** ldw | Загрузка памяти в стек |
| i<br>Lw | Lw Ldw | Загрузка константы в стек |
| 1 1 w w p M M M<br>1 1 w w i | *K.****Ms K p***M st***w***<br>K IA/IB stw | Выгрузка данных в память |
| ldi<br>L | L ldi | Загрузка байта в регистр |
| ldi<br>L<br>L | LL ldi | Загрузка слова в регистр |
| 1 1 0 0 A<br>A A | ***K***AAA jsr | Вызов подпрограммы в текущем банке |
| 0 1 0 0 A<br>A A | AAA zjsr | Вызов подпрограммы в нулевом банке |

1. K – префикс константы.
2. Md - операнд-источник в памяти.
***3. w*** - длина операнда.
4. L – контстанта.

Базовые команды могут предваряться префиксом.

Префикс константы содержит в информационном поле 4-разрядный код константы, которая расширяет адресное поле операнда, ветвления и адреса вызова программы.

Префикс операнда содержит 4-разрядное поле операнда в памяти или номер индексного регистра. Префикс константы модифицирует разряды номера страницы ОЗУ, хранящегося в регистре состояния программы, а также задает смещение индекса при косвенной адресации через индексные регистры. Имеется две префиксные команды, которые задают постинкремент и предекремент индексного регистра при косвенной адресации.

Префикс операнда расширяет стековые команды обработки данных до команд модификации данных в памяти и вводит битовые команды обработки данных в памяти. Форматы битовых команд обработки данных в памяти приведены в таблице 7. Коды команд с префиксом операнда приведены в таблице 9.

Таблица 7

| 7 4 3 0 | Мнемоника | Наименование | Примечания |
|---|---|---|---|
| 0 0 X x M<br>0 1 X 0 x N N N | *K* ***Md*** #N bxx | Операция обработки бита N | ***Md*** операнд-приемник в памяти |
| 0 0 X x M<br>0 1 1 0 x N N N<br>0/1 +/- A | *K* ***Ms*** #N btc ***K***AA | Проверка бита N и ветвление | ***Ms*** операнд-источник в памяти<br>***K*** старшие разряды ветвления |

Команды обработки данных имеют 2 формата - операции на стеке и с константой:

а) операции на стеке выполняются над верхними элементами стека с сохранением результата операции в стеке. Префикс операнда задает операцию модификации данных в памяти;

б) операции с константой выполняются над верхушкой стека и константой, которая расположена в следующих байтах после команды, результат операции записывается в стек. Префикс операнда задает операцию модификации данных в памяти.

Команды загрузки-выгрузки стека содержат поле адресации памяти. Префикс константы позволяет модифицировать адресное поле команды.

Команды вызова программ имеют длину 2 байта и содержат 12-разрядный адрес передачи управления, который позволяет прямо адресовать 4 Кбайт текущего или нулевого банка ПЗУ. Префикс константы позволяет задавать значение 4 старших разрядов счетчика команд и выполнять передачу управления в любой банк ПЗУ.

Команды проверки и ветвления имеют длину 2 байта. Первый байт команды содержит код проверки и тип данных. Старший разряд второго байта содержит условие передачи управления, остальные разряды содержат 7-разрядное смещение адреса передачи управления в дополнительном коде относительно счетчика команд. Префикс константы расширяет 7-разрядное смещение до 11 (15) разрядов.

В таблице 8 приведены коды базовых команд. В таблице 10 приведены коды команд с префиксом операнда. В таблице 11 приведено описание команд. Принятые сокращения приведены в таблице 7.

Таблица 7

| Сокращение | Содержание |
|---|---|
| K | Константа 4 разряда |
| E (S1 S -- **S**) | Содержимое стека данных до и после выполнения операции |
| R (R1 R -- **R**) | Содержимое стека возвратов до и после выполнения операции |
| L | Константа 8 разрядов |
| N | Номер 3 разряда |
| $\ldots_w$ | Длина операнда 1, 2, 4 байта |
| M | Операнд в памяти |
| AAA | Адрес передачи управления 12 разрядов |
| ±AA | Смещение 7 разрядов |

Таблица 8

Коды базовых команд

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 10 | **swap** | *20* | **swapd** | 30 | **swapw** | 40 | zjsr 0AA | 50 | **ad** | 60 | **add** | *70* | **adw** |
| 01 | 1 | 11 | **dup** | *21* | **dupd** | 31 | **dupw** | 41 | zjsr 1AA | 51 | **adl** | 61 | **adld** | *71* | **adlw** |
| 02 | 2 | 12 | drop | *22* | dropd | 32 | dropw | 42 | zjsr 2AA | 52 | **sub** | 62 | **subd** | *72* | **subw** |
| 03 | 3 | 13 | over | *23* | overd | 33 | overw | 43 | zjsr 3AA | 53 | **inc** | 63 | **incd** | *73* | **incw** |
| 04 | 4 | 14 | | *24* | | 34 | | 44 | zjsr 4AA | 54 | **umul** | 64 | ldlA | *74* | **umulw** |
| 05 | 5 | 15 | **bz** | *25* | **bzd** | 35 | **bzw** | 45 | zjsr 5AA | 55 | **umull** | 65 | ldlB | *75* | **umullw** |
| 06 | 6 | 16 | **blo** | *26* | -@ | 36 | **blow** | 46 | zjsr 6AA | 56 | **dec** | 66 | **decd** | *76* | **decw** |
| 07 | 7 | 17 | **blol** | *27* | @+ | 37 | **blolw** | 47 | zjsr 7AA | 57 | **and** | 67 | **andd** | *77* | **andw** |
| 08 | 8 | 18 | **bhi** | *28* | incP | 38 | **bhiw** | 48 | zjsr 8AA | 58 | **andl** | 68 | **andld** | *78* | **andlw** |
| 09 | 9 | 19 | **bhil** | *29* | decP | 39 | **bhilw** | 49 | zjsr 9AA | 59 | **xor** | 69 | **xord** | *79* | **xorw** |
| 0A | A | 1A | **beq** | *2A* | **beqd** | 3A | **beqw** | 4A | zjsr AAA | 5A | **xorl** | 6A | **xorld** | *7A* | **xorlw** |
| 0B | B | 1B | **beql** | *2B* | **beqld** | 3B | **beqlw** | 4B | zjsr BAA | 5B | **or** | 6B | **ord** | *7B* | **orw** |
| 0C | C | 1C | ldl | *2C* | ldlw | 3C | ldld | 4C | zjsr CAA | 5C | **orl** | 6C | **orld** | *7C* | **orlw** |
| 0D | D | 1D | (R)+ld | *2D* | (R)+ldw | 3D | (R)+ldd | 4D | zjsr DAA | 5D | **shl** | 6D | **shld** | *7D* | **shlw** |
| 0E | E | 1E | (IA) ld | *2E* | (IA) ldw | 3E | (IA) ldd | 4E | zjsr EAA | 5E | **st (IA)** | 6E | **stw (IA)** | *7E* | **std (IA)** |
| 0F | F | 1F | (IB) ld | *2F* | (IB) ldw | 3F | (IB) ldd | 4F | zjsr FAA | 5F | **st (IB)** | 6F | **stw (IB)** | *7F* | **std (IB)** |
| *80* | P0 | *90* | p0 ld | *A0* | p0 ldw | *B0* | p0 ldd | *C0* | jsr 0AA | *D0* | **st p0** | *E0* | **stw p0** | *F0* | **std p0** |
| *81* | P1 | *91* | p1 ld | *A1* | p2 ldw | *B1* | p4 ldd | *C1* | jsr 1AA | *D1* | **st p1** | *E1* | **stw p2** | *F1* | **std p4** |
| *82* | P2 | *92* | p2 ld | *A2* | p4 ldw | *B2* | ApushR | *C2* | jsr 2AA | *D2* | **st p2** | *E2* | **stw p4** | *F2* | SpushR |
| *83* | P3 | *93* | p3 ld | *A3* | p6 ldw | *B3* | BpushR | *C3* | jsr 3AA | *D3* | **st p3** | *E3* | **stw p6** | *F3* | RSpushR |
| *84* | P4 | *94* | p4 ld | *A4* | | *B4* | | *C4* | jsr 4AA | *D4* | **st p4** | *E4* | **muld** | *F4* | **mulw** |
| *85* | P5 | *95* | p5 ld | *A5* | | *B5* | | *C5* | jsr 5AA | *D5* | **st p5** | *E5* | **mulld** | *F5* | **mullw** |
| *86* | P6 | *96* | p6 ld | *A6* | | *B6* | | *C6* | jsr 6AA | *D6* | **st p6** | *E6* | ldlA | *F6* | (IA) |
| *87* | P7 | *97* | p7 ld | *A7* | **bmid** | *B7* | **bmiw** | *C7* | jsr 7AA | *D7* | **st p7** | *E7* | ldlB | *F7* | (IB) |
| *88* | Pp0 | *98* | pp0 ld | *A8* | pp0 ldw | *B8* | pp0 ldd | *C8* | jsr 8AA | *D8* | **st pp0** | *E8* | **stw pp0** | *F8* | **std pp0** |
| *89* | Pp1 | *99* | pp1 ld | *A9* | pp2 ldw | *B9* | pp4 ldd | *C9* | jsr 9AA | *D9* | **st pp1** | *E9* | **stw pp2** | *F9* | **std pp4** |
| *8A* | Pp2 | *9A* | pp2 ld | *AA* | pp4 ldw | *BA* | jsr (R) | *CA* | jsr AAA | *DA* | **st pp2** | *EA* | **stw pp4** | *FA* | RpopA |
| *8B* | Pp3 | *9B* | pp3 ld | *AB* | pp6 ldw | *BB* | ldlP | *CB* | jsr BAA | *DB* | **st pp3** | *EB* | **stw pp6** | *FB* | RpopB |
| *8C* | Pp4 | *9C* | pp4 ld | *AC* | **bgtd** | *BC* | **bgtw** | *CC* | jsr CAA | *DC* | **st pp4** | *EC* | ldlE | *FC* | RpopS |
| *8D* | Pp5 | *9D* | pp5 ld | *AD* | **bgtld** | *BD* | **bgtlw** | *CD* | jsr DAA | *DD* | **st pp5** | *ED* | Rdrop | *FD* | RpopRS |
| *8E* | Pp6 | *9E* | pp6 ld | *AE* | **bged** | *BE* | **bgew** | *CE* | jsr EAA | *DE* | **st pp6** | *EE* | ldlR | *FE* | |
| *8F* | Pp7 | *9F* | pp7 ld | *AF* | **bgeld** | *BF* | **bgelw** | *CF* | jsr FAA | *DF* | **st pp7** | *EF* | ret | *FF* | reti |

Примечания.

1. **Жирным** шрифтом выделены команды, в которых аргумент операции может задаваться префиксом операнда.
2. **Подчеркиванием** выделены команды, содержащие дополнительные информационные байты.

Таблица 9

Коды команд с префиксом операнда

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 10 | swap | *20* | swapd | 30 | swapw | 40 | #0 bic | 50 | ad | 60 | add | *70* | adw |
| 01 | 1 | 11 | dup | *21* | dupd | 31 | dupw | 41 | #1 bic | 51 | adl | 61 | adld | *71* | adlw |
| 02 | 2 | 12 | | *22* | | 32 | | 42 | #2 bic | 52 | sub | 62 | subd | *72* | subw |
| 03 | 3 | 13 | | *23* | | 33 | | 43 | #3 bic | 53 | inc | 63 | incd | *73* | incw |
| 04 | 4 | 14 | | *24* | | 34 | | 44 | #4 bic | 54 | umul | 64 | | *74* | umulw |
| 05 | 5 | 15 | bz | *25* | Bzd | 35 | bzw | 45 | #5 bic | 55 | umull | 65 | | *75* | umullw |
| 06 | 6 | 16 | blo | *26* | | 36 | blow | 46 | #6 bic | 56 | dec | 66 | decd | *76* | decw |
| 07 | 7 | 17 | blol | *27* | | 37 | blolw | 47 | #7 bic | 57 | and | 67 | andd | *77* | andw |
| 08 | 8 | 18 | bhi | *28* | | 38 | bhiw | 48 | #0 bis | 58 | andl | 68 | andld | *78* | andlw |
| 09 | 9 | 19 | bhil | *29* | | 39 | bhilw | 49 | #1 bis | 59 | xor | 69 | xord | *79* | xorw |
| 0A | A | 1A | beq | *2A* | Beqd | 3A | beqw | 4A | #2 bis | 5A | xorl | 6A | xorld | *7A* | xorlw |
| 0B | B | 1B | beql | *2B* | Beqld | 3B | beqlw | 4B | #3 bis | 5B | or | 6B | ord | *7B* | orw |
| 0C | C | 1C | | *2C* | | 3C | | 4C | #4 bis | 5C | orl | 6C | orld | *7C* | orlw |
| 0D | D | 1D | | *2D* | | 3D | | 4D | #5 bis | 5D | shl | 6D | shld | *7D* | shlw |
| 0E | E | 1E | | *2E* | | 3E | | 4E | #6 bis | 5E | st (IA) | 6E | stw (IA) | *7E* | std (IA) |
| 0F | F | 1F | | *2F* | | 3F | | 4F | #7 bis | 5F | st (IB) | 6F | stw (IB) | *7F* | std (IB) |
| *80* | P0 | *90* | | *A0* | | *B0* | | *C0* | #0 bix | *D0* | st p0 | *E0* | stw p0 | *F0* | std p0 |
| *81* | P1 | *91* | | *A1* | | *B1* | | *C1* | #1 bix | *D1* | st p1 | *E1* | stw p2 | *F1* | std p4 |
| *82* | P2 | *92* | | *A2* | | *B2* | | *C2* | #2 bix | *D2* | st p2 | *E2* | stw p4 | *F2* | |
| *83* | P3 | *93* | | *A3* | | *B3* | | *C3* | #3 bix | *D3* | st p3 | *E3* | stw p6 | *F3* | |
| *84* | P4 | *94* | | *A4* | | *B4* | | *C4* | #4 bix | *D4* | st p4 | *E4* | muld | *F4* | mulw |
| *85* | P5 | *95* | | *A5* | | *B5* | | *C5* | #5 bix | *D5* | st p5 | *E5* | mulld | *F5* | mullw |
| *86* | P6 | *96* | | *A6* | | *B6* | | *C6* | #6 bix | *D6* | st p6 | *E6* | | *F6* | |
| *87* | P7 | *97* | | *A7* | Bmid | *B7* | bmiw | *C7* | #7 bix | *D7* | st p7 | *E7* | | *F7* | |
| *88* | Pp0 | *98* | | *A8* | | *B8* | | *C8* | #0 btc | *D8* | st pp0 | *E8* | stw pp0 | *F8* | std pp0 |
| *89* | Pp1 | *99* | | *A9* | | *B9* | | *C9* | #1 btc | *D9* | st pp1 | *E9* | stw pp2 | *F9* | std pp4 |
| *8A* | Pp2 | *9A* | | *AA* | | *BA* | | *CA* | #2 btc | *DA* | st pp2 | *EA* | stw pp4 | *FA* | |
| *8B* | Pp3 | *9B* | | *AB* | | *BB* | | *CB* | #3 btc | *DB* | st pp3 | *EB* | stw pp6 | *FB* | |
| *8C* | Pp4 | *9C* | | *AC* | Bgtd | *BC* | bgtw | *CC* | #4 btc | *DC* | st pp4 | *EC* | | *FC* | |
| *8D* | Pp5 | *9D* | | *AD* | Bgtld | *BD* | bgtlw | *CD* | #5 btc | *DD* | st pp5 | *ED* | | *FD* | |
| *8E* | Pp6 | *9E* | | *AE* | Bged | *BE* | bgew | *CE* | #6 btc | *DE* | st pp6 | *EE* | | *FE* | |
| *8F* | Pp7 | *9F* | | *AF* | Bgeld | *BF* | bgelw | *CF* | #7 btc | *DF* | st pp7 | *EF* | | *FF* | |

Примечание. **Подчеркиванием** выделены команды, содержащие дополнительные информационные байты.

Таблица 10

| Код | Мнемоника | Наименование | Описание | |
|---|---|---|---|---|
| *0K* **0K** | *K*. **K** | префикс константы | константа $\mathbf{K_4}+K*2^4$ | |
| *0K* **0M** | *K*. **M** | префикс операнда | операнд в памяти *K*. **M** | |
| 0K | **@+** | префикс постинкремента | постинкремент индексного регистра | |
| 0K | **-@** | префикс предекремента | предекремент индексного регистра | |
| Арифметические операции | | | | |
| | **ad***w* | сложение | $E(S1_w\ S_w - S1_w+S_w)$ | |
| | ***M* ad***w* | | $E(S_w -),\ M_w=+S_w$ | |
| | **uad***w* | сложение без знака | $E(S1_w\ S_w - S1_w+S_w)$ | |
| | ***M u*ad***w* | | $E(S_w -),\ M_w=+S_w$ | |
| | **sub***w* | вычитание | $E(S1_w\ S_w - S1_w-S_w)$ | |
| | ***M* sub***w* | | $E(S_w -),\ M_w= -S_w$ | |
| | **usub***w* | вычитание без знака | $E(S1_w\ S_w - S1_w-S_w)$ | |
| | ***M u*sub***w* | | $E(S_w -),\ M_w= -S_w$ | |
| | **mul***w* | умножение со знаком | $E\ (S1_w\ S_w - S1_w*S_w)$ | |
| | ***M* mul***w* | | $E\ (S - M_w*S_w),\ M_w= *S_w$ | |
| | **umul***w* | умножение без знака | $E\ (S1_w\ S_w - S1_w*S_w)$ | |
| | ***M* umul***w* | | $E\ (S - M_w*S_w),\ M_w= *S_w$ | |
| Арифметические операции с константой | | | | |
| | **addl***w* | сложение | $E(S_w - S_w+L_w)$ | |
| | ***M* addl***w* | | $M_w=+L_w$ | |
| | **uaddl***w* | сложение без знака | $E(S_w - S_w+L_w)$ | |
| | ***M u*addl***w* | | $M_w=+L_w$ | |
| | **mull***w* | умножение со знаком | $E\ (S_w - S_w*L_w)$ | |
| | ***M* mull***w* | | $E\ (-M_w*L_w),\ M_w=*L_w$ | |
| | **umull***w* | умножение без знака | $E\ (S_w - S_w*L_w)$ | |
| | ***M* umull***w* | | $E\ (-M_w*L_w),\ M_w=*L_w$ | |
| Логические операции | | | | |
| | **xor** | исключающее ИЛИ | E(S1 S – S1 XOR S) | |
| | ***M* xor** | | E( S – ), M= XOR S | |
| | **or** | логическое ИЛИ | E(S1 S – S1 OR S) | |
| | ***M* or** | | E( S – ), M= OR S | |
| | **and** | логическое И | E(S1 S – S1 AND S) | |
| | ***M* and** | | E( S – ), M= AND S | |
| Логические операции с константой | | | | |
| | **xorl** | исключающее ИЛИ | E(S – S XOR L) | |
| | ***M* xorl** | | M=XOR L | |
| | **orl** | логическое ИЛИ | E(S– S OR L) | |
| | ***M* orl** | | M=OR L | |
| | **andl** | логическое И | E(S– S AND L) | |
| | ***M* andl** | | M= AND L | |
| Унарные операции | | | | |
| | **inc***w* | добавление 1 | $E(S_w - S_w+1)$ | |
| | ***M* inc***w* | | $M_w=+1$ | |
| | **dec***w* | вычитание 1 | $E(S_w - S_w-1)$ | |
| | ***M* dec***w* | | $M_w= -1$ | |
| | **neg***w* | унарный минус | $E(S_w - -S_w)$ | |
| | ***M* neg***w* | | $M_w= -M_w$ | |
| | **not** | инверсия | $E(S_w - \neg S_w$ | |
| | ***M* not** | | $M_w= \neg M_w$ | |
| Сдвиговые операции | | | | |
| | **shl***w* | сдвиг влево | $E(S_w - <<S_w<<0)$ | |
| | ***M* shl***w* | | $M_w= <<M_w<<0$ | |
| | **shr***w* | сдвиг вправо | $E(S_w - 0>>S_w)$ | |
| | ***M* shr***w* | | $M_w= 0>>M_w$ | |
| | **ashr***w* | арифметический сдвиг вправо | $E(S_w - MSB>>S_w)$ | |
| | ***M* ash***w***r*** | | $M_w= MSB>>M_w$ | |
| Операции ветвления | | | | |
| | **beq***w A* | равно | $E(S1_w\ S_w - S1_w);\ S1_w=?S_w$ | |
| | ***M b*eq***w A* | | $E(S_w -);\ M_w=?S_w$ | |
| | **bgt***w A* | больше со знаком | $E(\pm S1_w \pm S_w - S1_w);\ \pm S1_w>?\pm S_w$ | |
| | ***M b*gt***w A* | | $E(S_w -);\ \pm M_w>?\pm S_w,$ | |
| | **bge***w A* | больше либо равно со знаком | $E(\pm S1_w \pm S_w - S1_w);\ \pm S1_w>?\pm S_w$ | |
| | ***M b*ge***w A* | | $E(S_w -);\ \pm M_w>?\pm S_w,$ | |
| | **blo***w A* | меньше без знака | $E(S1_w\ S_w - S1_w);\ S1_w<?S_w$ | |
| | ***M b*lo***w A* | | $E(S_w -);\ M_w<?S_w$ | |
| | **bhi***w A* | больше без знака | $E(S1_w\ S_w - S1_w);\ S1_w>?S_w,$ | |
| | ***M b*hi***w A* | | $E(S_w -);\ M_w>?S_w,$ | |
| | **bz***w A* | равно 0 | $E(S1_w\ S_w - S1_w);\ S1_w=?S_w$ | |
| | ***M bz****w A* | | $E(S_w -);\ M_w=?S_w$ | |
| | **bmi***w A* | знак минус | $E\ (S_w-- S_w),\ MSB\ S_w=?\ 0$ | |
| | ***M* bmi***w A* | | $MSB\ M_w=?\ 0$ | |
| Операции сравнения с константой | | | | |

| | | | | |
|---|---|---|---|---|
| | **beql***w A* | равно | E( $S_w - S_w$); $S_w$=?$L_w$ | |
| | ***M* beql***w A* | | $M_w$=?$L_w$ | |
| | **bgtl***w A* | больше со знаком | E(±$S_w - S_w$) ; $S_w$<?$L_w$ | |
| | ***M* bdlt***w A* | | $M_w$<?$L_w$ | |
| | **bgel***w A* | больше либо равно со знаком | E(±$S_w - S_w$); ±$S_w$>?±$L_w$ | |
| | ***M* bgel***w A* | | ±$M_w$>?±$L_w$, | |
| | **blol***w A* | меньше без знака | E($S_w - S_w$); $S_w$<?$L_w$ | |
| | ***M* blol***w A* | | $M_w$<?$L_w$ | |
| | **bhil***w A* | больше без знака | E($S_w - S_w$); $S_w$>?$L_w$, | |
| | ***M* bhil***w A* | | $M_w$>?$L_w$, | |
| Битовые операции в памяти | | | | |
| | **M,#N bix** | инверсия бита | M = M xor $2^N$ | |
| | **M,#N bis** | установка бита | M = M or $2^N$ | |
| | **M,#N bic** | очистка бита | M = M and $^-2^N$ | |
| Операции на стеке данных | | | | |
| | **swap***w* | обмен верхушки | E ($S1_w$ $S_w$ -- $S_w$ $S1_w$) | |
| | ***M* swap***w* | | E ($S_w$ -- $M_w$); $M_w$ = $S_w$ | |
| | **drop***w* | удаление элемента | E ($S_w$ -- ) | |
| | **dup***w* | копирование верхушки | E ($S_w$ -- $S_w$ $S_w$) | |
| | **over***w* | копирование второго элемента | E ($S1_w$ $S_w$ -- $S1_w$ $S_w$ $S1_w$) | |
| | **rot***w* | перестановка элементов | E ($S2_w$ $S1_w$ $S_w$ – $S1_w$ $S_w$ $S2_w$) | |
| Операции загрузки/выгрузки | | | | |
| | **M ld***w* | загрузка | E (-- $M_w$) | |
| | **M stw** | выгрузка | E ($S_w$ --); $M_w$=$S_w$ | |
| | ***Ms* Md stw** | | $Md_w$=$Ms_w$ | |
| | **ldlw L** | загрузка константы | E (-- $L_w$) | |
| | ***M* ldwl L** | | $M_w$ = $L_w$ | |
| | **(R)+ ldw** | косвенное чтение ПЗУ | E (-- @Rw ); R=+w | |
| Операции с флагами | | | | |
| | **cli** | сброс флага I | I=0 | |
| | **sti** | установка флага I | I=1 | |
| | **cle** | сброс флага E | E=0 | |
| | **ste** | установка флага E | E=1 | |
| Операции управления | | | | |
| | ***K*AAA jsr** | вызов подпрограммы | R (-- PC+2); | |
| | **AAA zjsr** | вызов подпрограммы в банке 0 | R (-- PC+2); | |
| | **jsr (R)** | косвенный вызов подпрограммы | R (R -- PC+2); | |
| | **ret** | выход из программы | R (R --); | |
| | **reti** | выход из прерывания | R (X R --); RS=X | |
| | **M,#N btz ±*K*.A** | проверка бита и переход | Если M and $2^N$=?0, то PC=±K..A | |
| Команды загрузки регистров | | | | |
| | **ldlA** | регистр IA | IA={@PC+,@PC+} | |
| | **ldlB** | регистр IB | IB={@PC+,@PC+} | |
| | **ldlR** | стек возвратов | R ( -- {@PC+,@PC+}) | |
| | **ldlP** | страница и параграф | PG P = @PC+ | |
| Операции со стеком возвратов | | | | |
| | **dropR** | удаление элемента | R (R -- ) | |
| | **RpopA** | восстановление регистра IA | R(R --); IA=R | |
| | **RpopB** | восстановление регистра IB | R(R --); IB=R | |
| | **RpopS** | выгрузка в стек данных | R (S1S -- ); E (-- S1 S) | |
| | **ApushR** | сохранение регистра IA | R (-- IA ) | |
| | **BpushR** | сохранение регистра IA | R (-- IB ) | |
| | **SpushR** | сохранение стека данных | E (S1 S -- ); R (-- S1S ) | |
| | **RmovS** | копирование в стек данных | R (S1S -- S1S); E( -- S1 S) | |

Примечания.
1. Жирным шрифтом выделены команды, в которых аргумент операции может задаваться префиксом операнда.