## Лабораторная работа № 2

**Тема:** форматы данных, система команд учебной ЭВМ;

**Цель:** изучить основные классы команд, формат данных и способы адресации; продолжить знакомство с интерфейсом программной модели учебной ЭВМ;

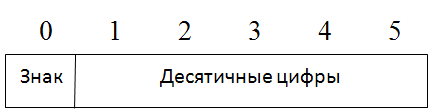
**Техническое обеспечение:** персональный компьютер, программная модель учебной ЭВМ;

**Основные понятия:** формат данных, система команд, формат команд, адресация, система операций.

Представление данных в модели учебной ЭВМ

Данные в учебной ЭВМ представляются в формате, показанном на рис. 4. Это целые десятичные цифры, изменяющиеся в диапазоне "-99 999...+99 999", содержащие знак и 5 десятичных цифр.

То есть, значение модулей целых десятичных цифр в 6-разрядной сетке не должны превышать значения 10n - 1 , где n – длина разрядной сетки. Соответственно, максимальное значение модуля целого числа равно 99 999.

Рисунок 4 - Формат десятичных данных учебной ЭВМ

Старший разряд данных используется для кодирования знака. Если число положительное, в знаковом разряде устанавливается 0, если отрицательное, то 1. Модуль числа располагается в младших разрядах, остающиеся свободными разряды, заполняют нулями.

Например: целое положительное (7) и отрицательное числа (-99), примут вид, рисунок 5:



Рисунок 5 - *а, б* – Представление десятичных чисел в разрядной сетке

Если результат арифметической операции выходит за пределы указанного диапазона, АЛУ вырабатывает сигнал переполнения OV = 1.

Например: рассмотрим операцию сложения, результат которой, вызывает переполнение

Единица находится в знаковом разряде, результат данной операции равен нулю. Флаг OV примет значение 1. Деление на ноль также вызывает переполнение. Результатом операции деления является целая часть частного. Например, 5/2=2 в данном случае, результат деления равен 2.

Общие сведения о форматах команд

Типовая команда, в общем случае, должна указывать:

* подлежащую выполнению операцию;
* адреса исходных данных (операндов), над которыми выполняется операция;
* адрес, по которому должен быть помещен результат операции;

Формат команд определяет ее структуру, т.е. количество разрядов, отводимых под всю команду, а также количество и расположение отдельных полей команды.

При создании вычислительной машины выбор формата команды влияет на многие характеристики будущей машины. Оценивая возможные форматы, нужно учитывать следующие факторы:

* общее число различных команд;
* общую длину команды;
* тип полей команды и их длина;
* адресуемость и способы адресации.

Длина команды – это важнейшее обстоятельство влияющее на организацию и емкость памяти, структуру шин, сложность и быстродействие процессора.

Система команд

Разнообразие типов данных, форм их представления и действий порождает необходимость использования различных команд – набора команд. Каждый процессор имеет собственный вполне определенный набор команд, называемой *системой команд,* приложение.

При рассмотрении системы команд ЭВМ обычно анализируют три аспекта: *форматы, способы адресации* и *систему операций*.

**Форматы команд.**Под форматом команд понимается состав, назначение и расположение отдельных полей команды. Большинство команд учебной ЭВМ являются одноадресными или безадресными, длиной в 6 разрядов.

В форматах команд выделяется три поля:

• два старших разряда **[0:1]** определяют код операции COP;

• разряд **2** может определять тип адресации (в одном случае (формат 5а) он определяет номер регистра);

• разряды **[3:5]** могут определять прямой или косвенный адрес памяти, номер регистра (в команде MOV номера двух регистров), адрес перехода или короткий непосредственный операнд. В двухсловных командах непосредственный операнд занимает поле **[6:11]**.

Для реализации одноадресного формата в процессоре предусмотрен аккумулятор (Асс): первый операнд и результат всегда размещаются в аккумуляторе, а второй операнд адресуется полем **[3:5]**, рисунок 6, формат 2. В безадресной команде отсутствует адресное поле, рисунок 6, формат 1. Исключение составляют двухсловные команды (ADI, DIVI и т.д.) с непосредственной адресацией и команда MOV, являющаяся двухадресной.

Например: команда HLT имеет формат 1; команда ADD @R3 – формат 3; команда MOV R3, R4 – формат 3а; RDI – формат 4; CALL 22 – формат 5; JNRZ R2, R4 – формат 5а.

На рисунке 6 представлен список форматов команд учебной ЭВМ, где:

• **COP** (Сode ОРeration) – код операции;

• **ADR** (Address Data Register) – адрес операнда в памяти;

• **ADC** (Address Data to Cell )– адрес перехода;

• **I** (Immediate operand) – непосредственный операнд;

• **R, R1, R2** – номер регистра;

• **TA** (Тype of Аddressing) - тип адресации;

• **X** – разряд не используется.

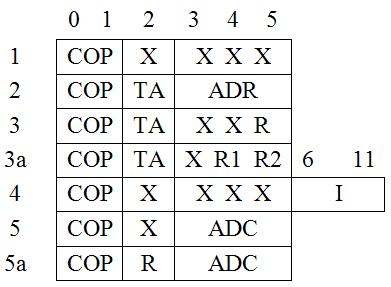


Рисунок 6 **-** Форматы команд учебной ЭВМ

В качестве примера, на рисунке 7, представлен формат команды ADD 30



Рисунок 7 **-** Формат команды add 30

Способы адресации

Большинство команд работают с операндами, расположение которых необходимо каким-то образом указать. Этот механизм называется адресацией.

В ЭВМ принято различать пять основных способов адресации: *прямая, косвенная, непосредственная, относительная, безадресная*. Некоторые способы адресации имеют свои разновидности. В модели учебной ЭВМ реализовано семь способов адресации, описание каждого, приведено ниже.

**1)****Прямая** *–* в адресном поле (ADR) располагается адрес операнда. Команда всегда имеет доступ только к одному и тому же адресу памяти. Т. е. значение может меняться, а адрес – нет, рисунок 8.

Разновидность - прямая регистровая, адресующая не ячейку памяти а РОН, рисунок 8.

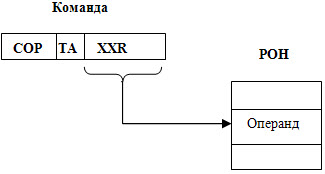
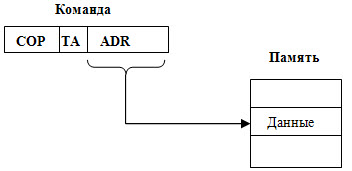


Рисунок 8 – Прямая адресация, Прямая регистровая адресация

**2)****Косвенная** – в поле адреса команды располагается адрес ячейки памяти, в которой хранится адрес операнда («адрес адреса»), рисунок 9.

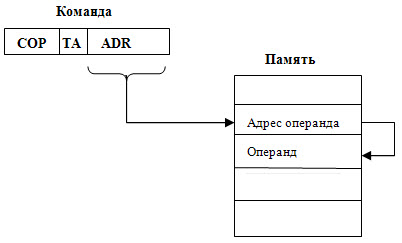


Рисунок 9 – Косвенная адресация

Разновидность – косвенно - регистровая адресация, при которой в поле команды размещается адрес РОН, хранящий адрес операнда, рисунок 10.

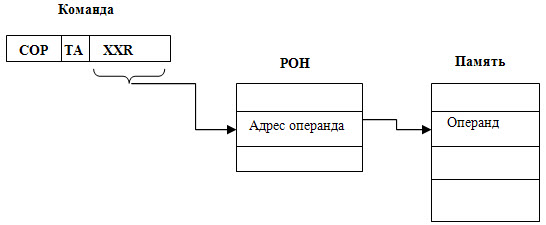


Рисунок 10 – Косвенно-регистровая адресация

**3)****Непосредственная** – самый простой способ указания операнда. В поле адреса команды располагается сам операнд, а не адрес операнда. Операнд автоматически вызывается из памяти вместе с командой, следовательно, сразу становится непосредственно доступным, рисунок 11. При непосредственной адресации не требуется дополнительного обращения к памяти.

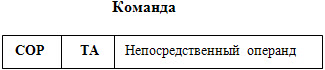


Рисунок 11 – Непосредственная адресация

**4)****Относительная** – адрес формируется как сумма двух слагаемых: базы, хранящейся в специальном регистре (RB) или одном из РОН, и смещения, извлекаемого из поля адреса команды.

Содержимое поля адреса команды складывается с содержимым регистра (RB). Таким образом формируется исполнительный адрес (Аисп), а адресный код в команде представляет собой смещение относительно адреса текущей команды, рисунок 12.

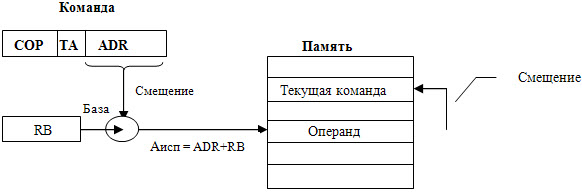


Рисунок 12 – Относительная адресация

**5)****Безадресная** – поле адреса в команде отсутствует, а адрес операнда или не имеет смысла для данной команды, или подразумевается по умолчанию. Часто безадресные команды подразумевают действия над содержимым аккумулятора (Асс), рисунок 13.

**6**) **Индексная с постинкрементом** – содержимое регистра используется как указатель ячейки памяти, в которой находится адрес операнда. Затем, после формирования исполнительного адреса, содержимое регистра увеличивается на 1, рисунок 13. Этот способ удобен для обработки данных, хранящихся в последовательно расположенных ячейках памяти.

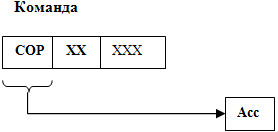
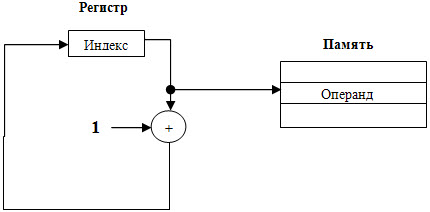


Рисунок 13 – Безадресная адресация, Индексная адресация с постинкрементом

**7)****Индексная с преддекрементом** – содержимое регистра уменьшается, а затем используется как указатель ячейки памяти, в которой находится адрес операнда, рисунок 14.

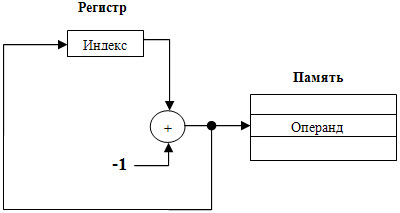


Рисунок 14 – Индексная адресация с преддекрементом

Система операций

Система команд учебной ЭВМ включает команды следующих классов:

* арифметико-логические и специальные: сложение, вычитание, умножение, деление;
* пересылки и загрузки: чтение, запись, пересылка, помещение в стек, извлечение из стека, загрузка указателя стека, загрузка базового регистра;
* ввода/вывода: ввод, вывод;
* передачи управления: безусловный и шесть условных переходов, вызов подпрограммы, возврат из подпрограммы, цикл, программное прерывание, возврат из прерывания;
* системные: пустая операция, разрешить прерывание, запретить прерывание, стоп.

Для решения с помощью ЭВМ некоторой задачи должна быть разработана программа. Программа на языке ЭВМ представляет собой последовательность команд. Код каждой команды определяет выполняемую операцию, тип адресации и адрес. Выполнение программы, записанной в памяти ЭВМ, осуществляется последовательно по командам в порядке возрастания адресов команд или в порядке, определяемом командами передачи управления.

Команды в память учебной ЭВМ вводятся в виде шестиразрядных десятичных чисел (см. форматы команд на рисунке 6, коды команд и способы адресации в приложение).

Порядок выполнения работы

Дана последовательность мнемокодов, которые необходимо преобразовать в машинные коды, занести в ОЗУ ЭВМ. Для каждого варианта, последовательность мнемокодов приведена в таблице 3. Для того, чтобы ввести команды в ОЗУ, необходимо:

1. Ввести последовательно команды в окно **Текст программы**. Запустить процедуру компиляции, рис. 15.

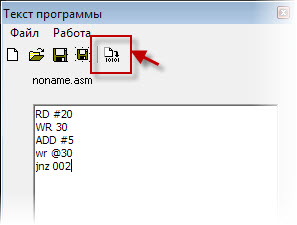
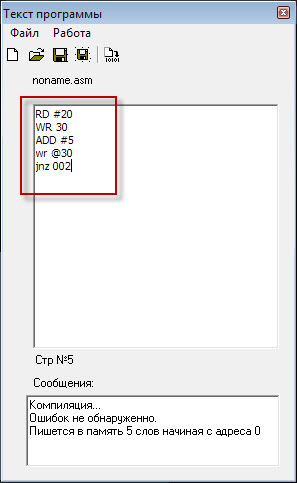


Рисунок 15 – Текстовый редактор, Компиляция

При этом, дизассемблированные команды запишутся в ячейки ОЗУ, начиная с адреса 000, и автоматически появятся в окне **Программа,** рисунок 16.

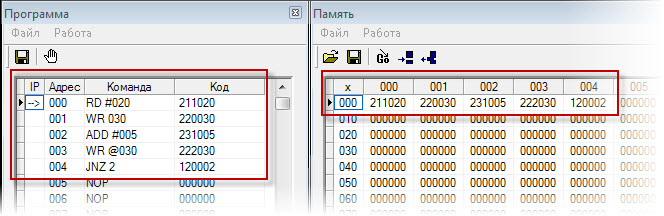


Рисунок 16 – Компиляция программы

Также возможна работа и непосредственно с кодом команд. Для этого необходимо: внести последовательно в ОЗУ, начиная с адреса 000, код команд; при переходе на следующую ячейку, команда автоматически ассемблируется и отобразится в окне **Текст программы,** рисунок 17. Например: код операции   
RD – 21, нам необходимо непосредственно адресовать операнд (20). Код непосредственной операции – 1. Заносим код (211020) команды RD #20 в ячейку памяти, начиная с адреса 000 и нажимаем кнопку **Go** в окне **Память**.

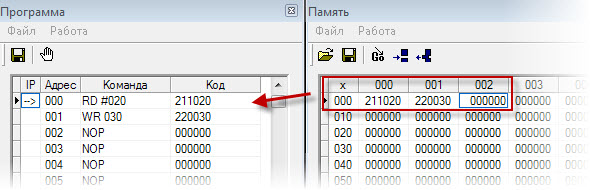


Рисунок 17 – Ввод кодов команд непосредственно в ОЗУ

2. В соответствии с вариантом задания, таблица 3, установить начальное значение в устройство ввода IR. Выполнить в режиме Шаг последовательность команд, рисунок 18.

3. Оформить последовательность мнемокодов и соответствующие машинные коды в форме таблицы 3.

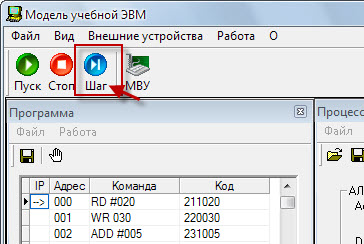
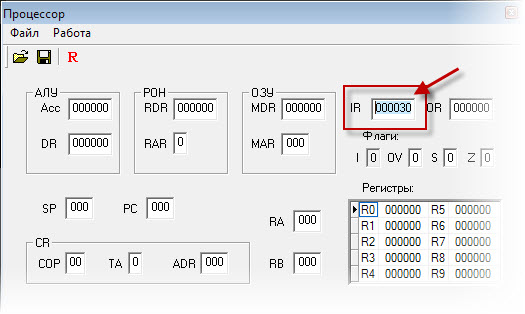


Рисунок 18 – Установленное значение в IR, Выполнение команд в режиме Шаг

Таблица 3 – Команды и коды

|  |  |  |  |
| --- | --- | --- | --- |
| **Адрес** | **Мнемокод** | **Код** | **Примечание** |
| 000 | RD #20 | 211020 | АСС ← 20 |
| 001 | WR 30 | 220030 | М(30) ← ACC |
| 002 | ADD #5 | 231005 | АСС ← АСС + 5 |
| 003 | WR @30 | 222030 | АСС ← М(М(30)) |
| 004 | JNZ 002 | 120002 | Переход к команде по адресу 002, если ≠ 0 |

**4.** Выполняя команды в режиме **Шаг**, зафиксировать изменения программно-доступных объектов (в данном случае это Асс, PC и ячейки ОЗУ  
М (30), М (М(30)), таблица 4. Если в программе образуется цикл, необходимо просмотреть не более двух повторений каждой команды, входящей в тело цикла;

5. В соответствии с вариантом задания - таблица 6, записать состояние учебной модели ЭВМ на уровне микрокоманд, таблица 5.

Таблица 4 – Содержимое регистров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PC** | **Асc** | **М(30)** | **М(М(30)** |  |
| 000 | 000000 | 000000 | 000000 |  |
| 001 | 000020 |  |  |  |
| 002 |  | 000020 |  |  |
| 003 | 000025 |  |  |  |
| 004 |  |  | 000025 |  |
| 002 |  |  |  |  |
| 003 | 000030 |  |  |  |
| 004 |  |  | 000030 |  |

6. Сохранить программу, для этого необходимо: в окне Текст программы выбрать меню Файл → Сохранить как. В открывшемся окне, выбрать расположение файла на диске и задать имя файла. Тип файла указать **.asm** и нажать кнопку **Сохранить**.

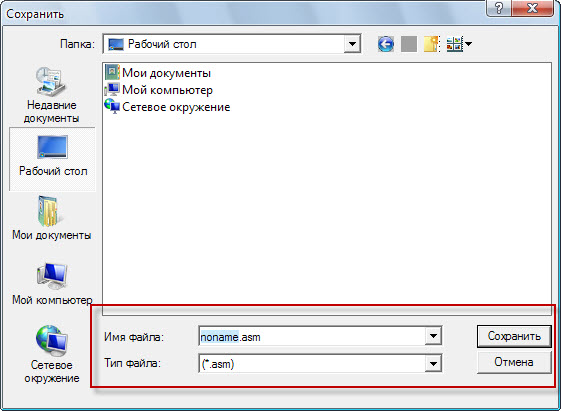
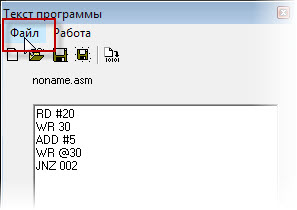


Рисунок 19 – Сохранение программы

Таблица 5 – Состояние модели в режиме моделирования на уровне микрокоманд

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Адрес (РС)** | **Мнемокод** | **Микрокоманда** | **ОЗУ** | **CR** | **АУ** | **Ячейки** |
| **MAR** | **MDR** | **COP** | **TA** | **ADR** | **Acc** | **DR** | **020** | **030** |
| **000** | **RD #20** | MAR := PC | 000 | 000000 | 00 | 0 | 000 | 000000 | 000000 | 000000 | 000000 |
|  |  | MRd | 000 |  |  |  |  |  |  |  |  |
|  |  | CR := MDR |  | 211020 |  |  |  |  |  |  |  |
|  |  | PC := PC+1 |  |  | 21 | 1 | 020 |  |  |  |  |
| **001** |  | Acc := ADR |  |  |  |  |  |  |  |  |  |
| **002** | **WR 30** | MAR := PC |  |  |  |  |  | 000020 |  |  |  |
|  |  | MRd | 001 |  |  |  |  |  |  |  |  |
|  |  | CR := MDR |  | 220030 |  |  |  |  |  |  |  |
|  |  | PC := PC+1 |  |  | 22 | 0 | 030 |  |  |  |  |
|  |  | MAR := ADR |  |  |  |  |  |  |  |  |  |
|  |  | MDR:=Acc | 030 |  |  |  |  |  |  |  |  |
|  |  | MWr |  | 000020 |  |  |  |  |  |  |  |
| **003** | **ADD #5** | MAR := PC |  |  |  |  |  |  |  |  | 000020 |
|  |  | MRd | 002 |  |  |  |  |  |  |  |  |
|  |  | CR := MDR |  | 231005 |  |  |  |  |  |  |  |
|  |  | PC := PC+1 |  |  | 23 | 1 | 005 |  |  |  |  |
|  |  | DR: = ADR |  |  |  |  |  |  |  |  |  |
|  |  | Fау:= ALI |  |  |  |  |  |  | 000005 |  |  |
| **004** | **WR @30** | MAR := PC |  |  |  |  |  | 000025 |  |  |  |

Таблица 6 – Список последовательности команд по вариантам

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Вариант** | **IR** | **Команда 1** | **Команда 2** | **Команда 3** | **Команда 4** | **Команда 5** |
| 1 | 000007 | IN | MUL #2 | WR 10 | WR @10 | JNS 001 |
| 2 | x | RD #17 | SUB #9 | WR 16 | WR @16 | JNS 001 |
| 3 | 100029 | IN | ADD #16 | WR 8 | WR @8 | JS 001 |
| 4 | x | RD #2 | MUL #6 | WR 11 | WR @11 | JNZ 00 |
| 5 | 000016 | IN | WR 8 | DIV #4 | WR @8 | JMP 002 |
| 6 | x | RD #4 | WR 11 | RD @11 | ADD #330 | JS 000 |
| 7 | 000002 | IN | MUL #2 | WR R2 | WR @R2 | JNS 001 |
| 8 | x | RD #10 | DIV #5 | WR 20 | WR @20 | JMP 000 |
| 9 | 000005 | IN | WR 30 | MUL 30 | WR R2 | JMP 001 |
| 10 | x | RD #5 | MUL #9 | WR 10 | WR @10 | JNZ 001 |
| 11 | 000017 | IN | WR 11 | ADD 11 | WR @11 | JMP 002 |
| 12 | x | RD #998 | ADD #19 | WR 9 | WR @9 | JNZ 001 |
| 13 | 000315 | IN | SUB #308 | WR 11 | WR @11 | JMP 001 |
| 14 | x | RD 4 | ADD #15 | WR 13 | WR @13 | JMP 001 |
| 15 | 100005 | IN | ADD #12 | WR 10 | WR @10 | JS 004 |
| 16 | x | RD 4 | SUB 8 | WR 8 | WR @8 | JNZ 001 |
| 17 | 000000 | IN | WR 9 | RD @9 | SUB #1 | JS 001 |
| 18 | x | RD #8 | WR R2 | SUB #1 | WR R3 | JMP 000 |
| 19 | 000015 | IN | SUB #5 | WR 30 | WR @30 | JMP 000 |
| 20 | x | RD #20 | MUL #2 | WR 10 | WR @10 | JMP 000 |
| 21 | 100005 | IN | MULI 100005 | WR 40 | WR @40 | JMP 000 |
| 22 | x | RDI 100002 | WR 7 | RD 7 | ADD 7 | JS 004 |
| 23 | 000010 | IN | ADD #10 | WR 9 | WR @9 | JMP 001 |
| 24 | x | RD #540 | ADD #60 | WR 10 | WR @10 | JNZ 001 |
| 25 | 100015 | IN | WR 15 | ADD 15 | WR @15 | JMP 002 |
| 26 | x | RD #30 | MULI 100030 | WR 20 | WR @20 | JMP 001 |
| 27 | 000100 | IN | WR @30 | ADD 30 | SUB #5 | JMP 001 |
| 28 | x | RD #120 | MUL #2 | WR 6 | WR @6 | JNO 001 |
| 29 | 000035 | IN | WR 8 | DIV #5 | WR @8 | JMP 001 |
| 30 | x | RD #10 | DIV #2 | WR R1 | WR @R1 | JMP 000 |

Форма и содержание отчета

1. Титульный лист;
2. Номер и тема лабораторной работы;
3. Формулировка и вариант задания, таблица 2.8;
4. Последовательность мнемокодов в форме таблицы 2.5;
5. Содержимое регистров в форме таблицы 2.6;
6. Состояние процессора и памяти в форме таблицы 2.7.

Для тематического контроля, в рамках лабораторной работы №2, разработан Тест (приложение А).

## Приложение

**Система команд учебной ЭВМ**

| **КОП** | **Мнемокод** | **Название** | **Действие** | |
| --- | --- | --- | --- | --- |
| 00 | nop | Пустая операция | Нет | |
| 01 | in | Ввод | Асе ← IR | |
| 02 | out | Вывод | OR ← Асc | |
| 03 | iret | Возврат из прерывания | FLAGS.PC ← M(SP); INC(SP) | |
| 04 | wrrb | Загрузка базового регистра RB | RB ← CR[ADR] | |
| 05 | wrsp | Загрузка указателя стека SP | SP ← CR[ADR] | |
| 06 | push | Поместить в стек | DEC(SP); M(SP) ← R | |
| 07 | pop | Извлечь из стека | R → M(SP); INC(SP) | |
| 08 | ret | Возврат | PC → M(SP); INC(SP) | |
| 09 | hlt | Стоп | Конец командных циклов | |
| 10 | jmp | Безусловный переход | PC ← CR[ADR] | |
| 11 | jz | Переход, если 0 | if Асc = 0 then PC ← CR[ADR] | |
| 12 | JNZ | Переход, если не 0 | if Асc ≠ 0 then PC ← CR[ADR] | |
| 13 | JS | Переход, если отрицательно | if Асc < 0 then PC ← CR[ADR] | |
| 14 | JNS | Переход, если положительно | if Асc > 0 then PC ← CR[ADR] | |
| 15 | JO | Переход, если переполнение | if |Acc| ≥ 99999 then PC ← CR[ADR] | |
| 16 | JNO | Переход, если нет переполнения | if |Acc| ≤ 99999 then PC ← CR[ADR] | |
| 17 | JRNZ | Цикл | | DEC(R); if R > 0 then PC ← CR[ADR] |
| 18 | INT | Программное прерывание | | DEC(SP); M(SP) ← FLAGS.PC; PC ← M(V) |
| 19 | CALL | Вызов подпрограммы | | DEC(SP); M(SP) ← PC; PC ← CR(ADR) |
| 20 | нет |  | |  |
| 21 | RD | Чтение | | Acc ← DD |
| 22 | WR | Запись | | M(\*) ← Acc |
| 23 | ADD | Сложение | | Acc ← Acc + DD |
| 24 | SUB | Вычитание | | Acc ← Acc - DD |
| 25 | MUL | Умножение | | Acc ← Acc x DD |
| 26 | DIV | Деление | | Acc ← Acc/DD |
| 27 | нет |  | |  |
| 28 | EI | Разрешить прерывание | | IF ← 1 |
| 29 | DI | Запретить прерывание | | IF ← 0 |
| 30 | MOV | Пересылка | | Rl ← R2 |
| 31 | RD | Чтение | | Acc ← R\* |

| **КОП** | **Мнемокод** | **Название** | **Действие** | |
| --- | --- | --- | --- | --- |
| 32 | WR | Запись | | R\* ← Acc |
| 33 | ADD | Сложение | | Acc ← Acc + R\* |
| 34 | SUB | Вычитание | | Acc ← Acc - R\* |
| 35 | MUL | Умножение | | Acc ← Acc x R\* |
| 36 | DIV | Деление | | Acc ← Acc/R\* |
| 37 | IN | Ввод | | Acc ← BУ(CR[ADR\*]) |
| 38 | OUT | Вывод | | BУ(CR[ADR\*]) ← Асс |
| 39 | нет |  | |  |
| 40 | нет |  | |  |
| 41 | RDI | Чтение | | Асс ←I |
| 42 | нет |  | |  |
| 43 | ADI | Сложение | | Асс ← Асc + I |
| 44 | SBI | Вычитание | | Асс ← Асc -I |
| 45 | MULI | Умножение | | Асс ← Асc х I |
| 46 | DIVI | Деление | | Асс ← Асс/I |

Приняты следующие обозначения:

* Acc – аккумулятор
* PC – счётчик команд
* SP – указатель стека
* RB – регистр базы
* IR – регистр ввода
* OR – регистр вывода
* FLAGS – вектор флагов: IF,OV,S,Z
* IF - флаг разрешения прерывания
* DD – данные, формируемые командой в качестве (второго) операнда: прямо или косвенно адресуемая ячейка памяти или трехразрядный непосредственный операнд;
* R – содержимое регистра общего назначения (РОН)
* R\* – содержимое РОН или косвенно адресуемой через регистр ячейки памяти;
* M( ) – содержимое ячейки памяти
* M(\*) – содержимое ячейки памяти, прямо или косвенно адресуемой в команде;
* CR – регистр команды
* CR[ADR] – трёхразрядное поле ADR регистра CR;
* CR[ADR\*] – два младших разряда поля ADR регистра CR;
* V – адрес памяти, соответствующий вектору прерывания;
* I – пятиразрядный непосредственный операнд со знаком.

## Приложение

**Типы адресации, их коды и обозначения**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обозначение** | **Код ТА** | **Тип адресации** | **Исполнительней адрес** | **Пример команды** |
|  | 0 | Прямая (регистровая) | ADR (R) | add 23 (add r3) |
| **#** | 1 | Непосредственная | - | add #33 |
| @ | 2 | Косвенная | ОЗУ(ADR)[3:5] | add @33 |
| [ ] | 3 | Относительная | ADR + RB | add [33] |
| @R | 4 | Косвенно-регистровая | РОН(R)[3:5] | add @r3 |
| @R+ | 5 | Индексная с постинкрементом | РОН(R)[3:5], R:=R+1 | add @r3+ |
| -@R | 6 | Индексная с преддекрементом | R:=R-1, РОН(R)[3:5] | add -@r3 |