**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ**
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Кафедра компьютерной инженерии

---

# Методические указания и задания к выполнению курсовой работе по программированию

часть 2

*для студентов специальности*
*6.050102 «Компьютерная инженерия»*
*дневной и заочной форм обучения*

Составители
Назаренко В.И., Иванов А.Ю., Демеш Н.С.

Рассмотрено
на заседании кафедры КИ
протокол № 3 от 23.11.2009 г.

Утверждено на заседании
учебно-издательского совета ДонНТУ
протокол № 1 от 01.03.2010 г.

Учебное издание

Донецк - 2010 г.

*УДК 681.3(07)*

Методические указания и задания к выполнению курсовой работы по программированию, часть 2 (для студентов специальности 6.050102 дневной и заочной форм обучения). Сост. В.И.Назаренко, Иванов А.Ю., Н.С.Демеш – Донецк, ДонНТУ, 2010. – 68 с.

Содержанием курсовой работы по программированию является разработка программы-эмулятора для заданной гипотетической (учебной) ЭВМ. В сборнике приведены описания, функциональные схемы, системы команд и методика программирования на машинном языке ЭВМ C1, C2, D1, D2, что является составной частью заданий к курсовой работе по программированию.

Составители:
доц. Назаренко В.И.,
ст.пр.Иванов А.Ю.,
асс. Демеш Н.С.

Ответственный
за выпуск
проф. Святный В.А.

Рецензент
доц. Теплинский С.В.
доц. Губенко Н.Е.

# 1. УЧЕБНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА С1

## 1.1 Архитектура ЭВМ С1

Учебная ЭВМ С1 представляет собой двухадресную ЭВМ. Ее структурная схема приведена на рис.5. В состав ЭВМ С1 входят следующие основные узлы:

- оперативная память MEMORY емкостью 256 22-разрядных ячеек;
- восьмиразрядный регистр адреса RA, определяющий номер ячейки памяти, с которой происходит взаимодействие в данный момент времени;
- двадцатидвухразрядный регистр слова RS для временного хранения читаемой или записываемой информации;
- восьмиразрядный счетчик адреса команд SAK, предназначенный для хранения адреса следующей команды;
- двадцатидвухразрядный регистр команд RK, хранящий выполняемую команду;
- двадцатидвухразрядный регистр-аккумулятор AC для приема и хранения первого операнда;
- двадцатидвухразрядные операционные регистры OR1 и OR2, непосредственно используемые при выполнении операции в АЛУ по отношению к заданным двум операндам;
- арифметико-логическое устройство АЛУ, выполняющее операцию, заданную кодом операции команды;
- двадцатидвухразрядный регистр RR, сохраняющий результат выполнения операции в АЛУ;
- дешифратор кода операции DC;
- трехразрядный регистр признаков завершения операции RP.

При выполнении арифметической операции первый бит RP устанавливается в «1», если ее результат отрицательный; второй бит устанавливается в «1» при нулевом результате этой операции; третий бит принимает значение «1» в случае некорректности выполнения операции (переполнение регистра RR или попытка деления на нуль). Эти биты на схеме ЭВМ обозначены как признаки S, Z и C.

Для логических операций, команд сдвига и команды пересылки производится лишь анализ результата на равенство нулю: RP[1] = 1, если результат операции не равен нулю; RP[2] = 1, если этот результат равен нулю.

Регистр команды разделяется на следующие поля: код выполняемой операции КОР (биты 1 – 4), модификатор адреса МР (биты 5 – 6), адреса А1 (биты 7 – 14) и А2 (биты 15 – 22) первого и второго операндов.

Адреса ячеек памяти изменяются от 0 до 255 (от 00000000 до 11111111) в двоичной системе счисления). В ячейке с нулевым адресом постоянно находится нуль, который можно только считывать.

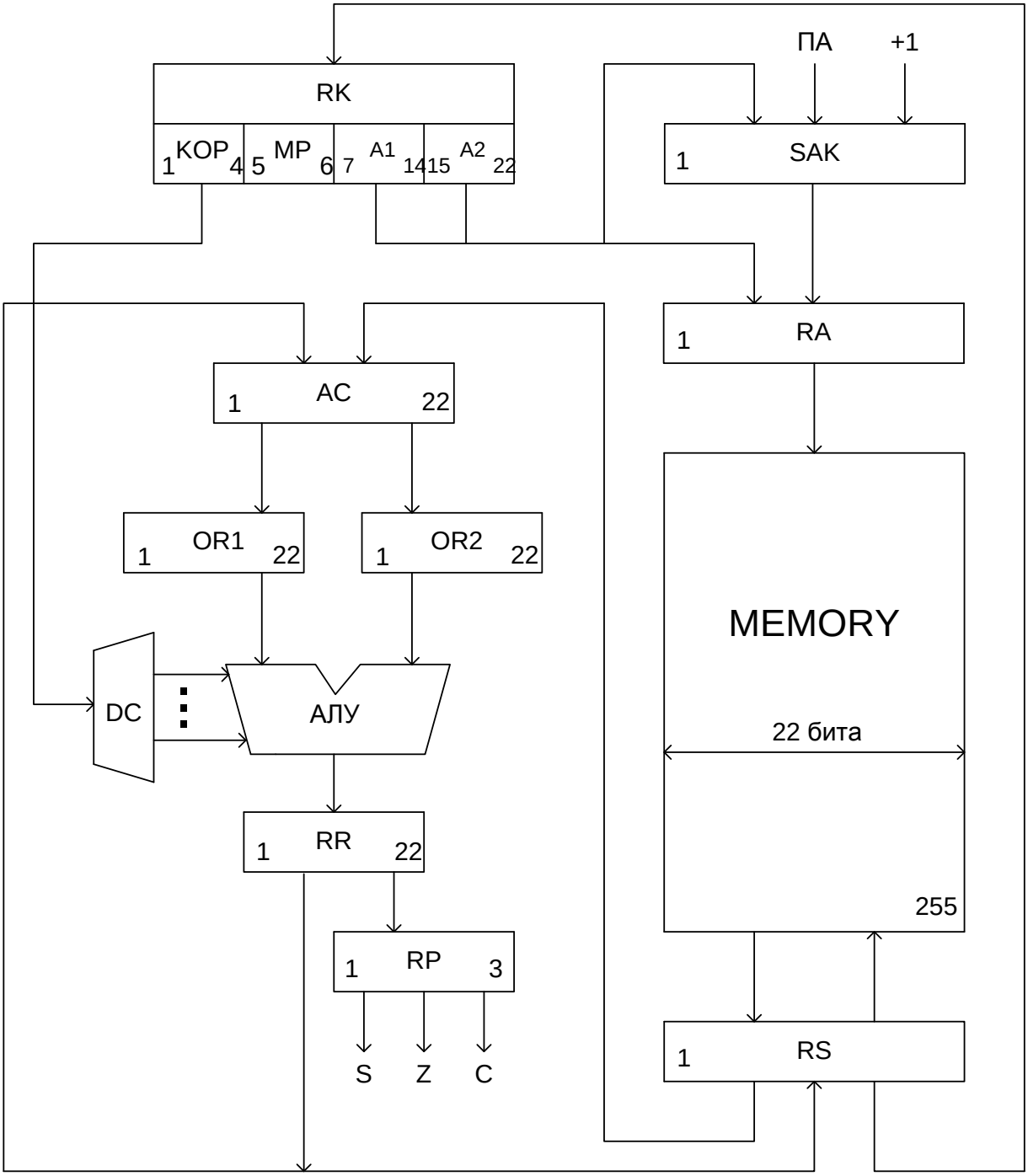


Рисунок 1 - Структурная схема ЭВМ С1

Система команд ЭВМ С1, приведенная в табл.1, включает в себя:

- арифметико-логические команды;
- команды сдвига;
- команду пересылки;
- команды управления порядком выполнения программы.

К арифметико-логическим командам относятся: сложение, вычитание, умножение, деление, конъюнкция, дизъюнкция, сумма по модулю 2.

К командам сдвига относятся сдвиг влево арифметический и сдвиг вправо арифметический.

В состав команд управления порядком выполнения программы входят: вызов подпрограммы, возврат из подпрограммы, безусловный переход, условный переход по S и Z, условный переход по C и останов.

В команде «Останов» разряды 5 – 22 не используются.

Таблица 1. Система команд ЭВМ С1

| Код опер. | Наименование | Признаки | | | Выполнение команды |
|---|---|---|---|---|---|
| | | S | Z | C | |
| 0000 | Останов | – | – | – | |
| 0001 | Пересылка | + | + | – | $(A1) \to A2$ |
| 0010 | Сложение | + | + | + | $MP = 00$<br>$(A1) \otimes (A2) \to AC, A2$<br>$MP = 01$<br>$(A1) \otimes (A2) \to AC$<br>$MP = 10$<br>$(A1) \otimes (AC) \to AC, A2$<br>$MP = 11$<br>$(A1) \otimes (AC) \to AC$ |
| 0011 | Вычитание | + | + | + | |
| 0100 | Умножение | + | + | + | |
| 0101 | Деление | + | + | + | |
| 0110 | Конъюнкция | + | + | – | |
| 0111 | Дизъюнкция | + | + | – | |
| 1000 | Сумма по модулю 2 | + | + | – | |
| 1001 | Сдвиг влево арифм. | + | + | – | $(A1) \Leftarrow A2$ *или* $(AC)$ |
| 1010 | Сдвиг вправо арифм. | + | + | – | $(A1) \Rightarrow A2$ *или* $(AC)$ |
| 1011 | Вызов подпрограммы | – | – | – | $(SAK) \to A2;\ A1 \to SAK$ |
| 1100 | Возврат из подпр-мы | – | – | – | $(A2) \to SAK$ |
| 1101 | Безусловный переход | – | – | – | $A1 \to SAK$ |
| 1110 | Усл. переход по S и Z | – | – | – | См.описание |
| 1111 | Усл. переход по C | – | – | – | См.описание |

Круглые скобки в графе «Выполнение команды» означают содержимое регистра или ячейки памяти. Например, запись A1 – это адрес ячейки, а (A1) – содержимое ячейки с адресом A1.

## 1.2 Выполнение команд в ЭВМ С1

В ЭВМ С1 используются 22-разрядные числа с фиксированной запятой (целые числа со знаком). Отрицательные значения чисел записываются в дополнительном коде.

Адресность выполняемой команды определяется модификатором адреса МР (разряды 5 и 6 регистра команд RK). Модификатор МР может принимать следующие значения: 00, 01, 10, 11.

В арифметико-логических операциях первый операнд содержится в аккумуляторе АС, второй – в операционном регистре OR2. Непосредственно

перед выполнением операции в АЛУ содержимое аккумулятора переписывается в регистр OR1. Результат операции из регистра RR пересылается в аккумулятор.

Если MP = 00, то содержимое ячейки с адресом A1 пересылается в аккумулятор, а ячейки с адресом A2 – в регистр OR2. Результат операции записывается, кроме аккумулятора, также в память по адресу A2.

Если MP = 01, то исходные операнды, как и ранее, содержатся в AC и OR2, но результат операции в память не передается.

Если MP = 10, то содержимое аккумулятора переписывается в регистр OR2, а из памяти читается ячейка с адресом A1 и ее содержимое пересылается в аккумулятор AC. Результат операции записывается в аккумулятор и в память по адресу A2.

При MP = 11 выполняются аналогичные действия, но результат операции в память не записывается.

Выполнение команды начинается с загрузки содержимого ячейки памяти, адрес которой задан в SAK, в регистр команд RK. При этом значение адреса выполняемой команды из SAK переписывается в регистр адреса RA, по этому адресу читается содержимое соответствующей ячейки памяти MEMORY и через буферный регистр слова RS пересылается в регистр RK. В последующем из RK выделяются поля KOP, MP, A1 и A2. Для подготовки выборки следующей команды значение счетчика SAK увеличивается на единицу. Дешифратор DC анализирует значение KOP и определяет команду, которая должна выполняться в данный момент (дешифратор имеет 16 выходов по количеству дешифрируемых команд).

В зависимости от значения кода операции KOP дешифратор DC вырабатывает управляющие сигналы, которые определяют тип выполняемой операции.

Далее в арифметико-логических командах осуществляется формирование значений операндов. Первый операнд определяется следующим образом: значение адреса A1 из регистра команд RK передается в регистр адреса RA, содержимое ячейки памяти с адресом, который задан регистром RA, передается в регистр слова RS, откуда пересылается в аккумулятор AC. Если модификатор MP равен «00» или «01», то значение второго операнда определяется из памяти по адресу A2 и переписывается в регистр OR2. Если модификатор MP равен «10» или «11», то содержимое аккумулятора предварительно переписывается в регистр OR2, а затем уже аккумулятор заполняется значением первого операнда, содержащегося в памяти по адресу A1.

В арифметико-логических операциях аккумулятор переписывается в регистр OR1, содержимое регистров OR1 и OR2 поступает на входы арифметико-логического устройства. Следовательно, операндами в арифметико-логической операции всегда является содержимое регистров OR1 и OR2. В дальнейшем АЛУ выполняет операцию, заданную дешифратором DC, ее результат поступает в выходной регистр RR и пересылается в аккумулятор AC. Кроме того, в командах с модификатором MP, равным «00» или «10», результат переписывается в память по адресу A2.

Результат арифметической операции используется для формирования в регистре RP признака S (S = 1, если результат меньше нуля), Z (Z = 1, если результат равен нулю) или C (C = 1, если отмечена некорректность выполнения операции – переполнение регистра RR или попытка деления на нуль). Требуемые изменения регистра RP отражены в табл.5.

В регистре RP нет бита, устанавливающегося в 1, если результат арифметической операции положительный. Здесь подразумевается, что если результат такой операции неотрицательный и ненулевой, то он положительный. Следовательно, в этом случае первые три бита регистра RP должны быть установлены в нулевое положение.

Для логических операций, команд сдвига и команды пересылки производится лишь анализ результата на равенство нулю: S = 1, если результат операции не равен нулю; Z = 1, если этот результат равен нулю.

Символ “–” в графе признаков табл.1 определяет ситуацию, которая не может возникнуть в данной операции (например, некорректность при выполнении операции сдвига), или то, что данная операция не изменяет регистр RP (например, команда безусловного перехода).

Сущность изменения регистра RP рассмотрим на конкретном примере. Предположим, что в программе выполнена команда сложения и при этом получен отрицательный результат. Тогда в состояние «1» должен быть установлен лишь первый бит регистра RP, остальные биты должны быть сброшены на нуль.

*Примечание*. Если в команде, приведенной в табл.1, все три графы признаков отмечены символом “–”, то это означает лишь то, что при этом сохраняются предыдущие значения признаков (но не выполняется сброс их на нуль).

В командах сдвига осуществляется сдвиг разрядов первого операнда, а количество сдвигов определяет второй операнд. Следовательно, вне зависимости от значения модификатора производится сдвиг разрядов операционного регистра OR1 влево или вправо на то количество разрядов, которое определено содержимым регистра OR2.

Заполнение операционных регистров OR1 и OR2 в основном аналогично арифметико-логическим операциям, кроме одного обстоятельства: при MP = 00 и MP = 01 в регистр OR2 пересылается не содержимое ячейки памяти с адресом A2, а непосредственно значение адреса A2 из регистра команд RK. Следовательно, при MP = 00 и MP = 11 в регистр OR2, как и в арифметико-логической операции, пересылается содержимое аккумулятора. В этом случае поле A2 регистра команд RK игнорируется и может быть любым (обычно равным нулю).

Как записано в табл.1, в ЭВМ С1 выполняется арифметический сдвиг. Его особенностью, в отличие от логического сдвига, является то, что при сдвиге влево знаковый разряд не затрагивается, при этом освобождаемые в правой части регистра OR1 разряды заполняются нулями; при сдвиге вправо разряд знака (0 или 1) распространяется на сдвигаемые разряды. Разряды, выходящие при сдвиге за пределы OR1, теряются.

Если (OR2) > 14, то сдвиг разрядов регистра OR1 не выполняется; в этом случае по команде сдвига влево разряды 1..15 заполняются нулями, по команде сдвига вправо – значением разряда знака (0 или 1).

По команде безусловного перехода содержимое поля A1 регистра команд RK пересылается в счетчик SAK, после чего происходит выборка новой команды и ее выполнение. Адрес A2 в этой команде не используется.

При выполнении команды условного перехода по S и Z производится анализ значений битов регистра признаков RP. Если S = 1 (для арифметической операции это означает, что (AC) < 0, поскольку результат операции всегда записывается в аккумулятор), то происходит передача управления по адресу A1; если Z = 1, т.е. (AC) = 0, то в SAK передается адрес A2; если S = 0 и Z = 0, т.е. (AC) > 0, то управление передается следующей команде. Если в регистре RP установлен признак C = 1, то команда условного перехода по C производит передачу управления по адресу A1, в противном случае – следующей по порядку команде. При отсутствии команды условного перехода по C никакой реакции на признак C = 1 не происходит.

В реальных ЭВМ команда условного перехода по C определяла бы переход на подпрограмму обработки аварийного прерывания. В ЭВМ С1 по адресу A1 достаточно вывести на экран сообщение об аварийном прерывании (код операции и адрес команды), после чего произвести останов работы ЭВМ.

*Примечание.* Вполне очевидно, что признаки S и Z не могут быть одновременно равными единице.

При обращении к подпрограмме должны быть выполнены два действия: передача управления на начальный участок подпрограммы и возврат в вызывающую программу после отработки подпрограммы. Эта работа в ЭВМ С1 выполняется двумя командами: «Вызов подпрограммы» и «Возврат из подпрограммы».

В команде «Вызов подпрограммы» вначале в регистр RA заносится адрес A2 и содержимое счетчика SAK через регистр RS записывается по этому адресу. После этого адрес A1 переписывается в счетчик SAK, чем достигается переход к выполнению первой команды подпрограммы.

В команде «Возврат из подпрограммы» в регистр RA записывается адрес A2, содержимое ячейки памяти с этим адресом выбирается в регистр RS, а затем пересылается в счетчик SAK; тем самым восстанавливается адрес команды, сохраненной ранее при обращении к подпрограмме, т.е. команды, записанной в вызывающей программе после команды вызова подпрограммы. Адрес A1 в данной команде не используется.

В команде пересылки и в командах управления порядком выполнения программы модификатор MP не используется.

## 1.3 Программирование в кодах ЭВМ С1

На начальном этапе разработки машинной программы целесообразно каждую команду заменить ее условным обозначением.

Машинная команда ЭВМ С1 состоит из четырех частей: код операции, модификатор адреса и адреса первого и второго операндов.

Для кода операции будем использовать следующие обозначения:

Halt – останов;
Mov – загрузка акумулятора;
Add – сложение;
Sub – вычитание;
Mul – умножение;
Div – деление;
And – конъюнкция;
Or – дизъюнкция;
Xor – сумма по модулю 2;
Shl – сдвиг влево арифметический;
Shr – сдвиг ввправо арифметический;
Call – вызрв подпрограммы;
Ret – возврат из подпрограммы;
Jump – безусловный переход;
JumSZ – условный переход по S и Z;
JumC – условный переход по С.

Для модификатора адреса будем использовать непосредственно его численное значение: 0, 1, 2 или 3.

В адресной части команды может быть:

- условное обозначение адреса;
- переменная или константа, записанные в ячейки памяти.

В последнем случае имя переменной или значение константы будем заключать в угловые скобки.

*Примеры.*

а) Mov 0 <x> <y>

Пересылка содержимого адреса, в котором находится переменная *x*, и запись этого значения в ячейку, предназначенную для хранения переменной *y*.

б) And 1 <x> <y>

Сложение переменных *x* и *y* с записью результата в аккумулятор.

в) Jump 0 Met 0

Безусловный переход по адресу Met (на метку Met).

г) JumSZ 0 Met1 Met2

Условный переход по адресу Met1, если (АС) < 0, или по адресу Met2, если (АС) = 0.

### 1.3.1 Программирование арифметического выражения

$$y = \frac{338 + 25a}{2a - 1}; \quad a = 10$$

В системе команд ЭВМ С1 непосредственная адресация не применяется. Поэтому все константы, входящие в состав расчетной формулы, должны быть записаны в ячейки памяти.

*Примечание.* Поскольку ячейка памяти ЭВМ С1 имеет размер 22 разряда, то максимальное значение переменной может быть $2^{21} - 1 = 2\,097\,151$.

```
14   Mov    0   <a>    R1        a → R1
15   Mul    0   <2>    R1        2 R1 → R1 + Acc
16   Sub    1   R1     <1>       (Acc) − 1 → Acc
17   Add    3    0     R1        2a − 1 → R1
18   Mov    0   <a>    R2        a → R2
19   Mul    1   <25>   R2        25a → Acc
1A   Add    2   <338>  R2        25a + 338 → R2
1B   Div    2    R2    <y>       (338 + 25a)/(2a − 1) → < y >
1C   Halt
1D   a
1E   1
1F   2
20   25
21   338
22   y
23   R1
24   R2
```

Здесь в первой колонке – адрес ячейки памяти, содержащей машинную команду, значение переменной или константу. Адрес записан в шестнадцатеричной системе счисления. В данном случае принято, что машинная программа имеет пусковой адрес $14_{16} = 20_{10}$ .

Второй этап – запись машинной команды в двоичном коде (адреса ячеек памяти по-прежнему остаются шестнадцатеричными).

```
14   0001 00 00011101 00100011
15   0100 00 00011111 00100011
16   0011 01 00100011 00011110
17   0010 00 00000000 00100011
18   0001 00 00011001 00100100
19   0100 01 00100000 00100100
1A   0010 10 00100001 00100100
1B   0101 10 00100100 00100010
1C   0000 00 00000000 00000000
1D   0000 00 00000000 00001010
```

```
1E   0000 00 00000000 00000001
1F   0000 00 0000000 00000010
20   0000 00 00000000 00011001
21   0000 00 00000001 01010010
```

Здесь адреса 22, 23 и 24 не отображаются, поскольку значение переменной *y* и содержимое буферных ячеек *R1* и *R2* формируются в процессе работы программы.

Третий этап – подготовка файла в заданной входной системе счисления (например, восьмеричной).

В первой строке текстового файла записывается пусковой адрес (в десятичной системе). В остальных строках – содержимое ячеек памяти, предварительно разделенное на триады справа налево. Адреса ячеек памяти не отображаются.

```
   20
01016443
04017423
03221036
02000043
02016444
04220044
02420444
05424042
00000000
00000012
00000001
00000010
00000031
00000522
```

Значение переменной *y* как результат вычислений должно быть записано в ячейку с адресом $22_{16} = 34_{10}$ . Этот результат равен $33_{10} = 21_{16} = 41_8$

В п. 1.3.1 проверены машинные команды Mov, Add, Sub, Mul, Div и Halt.

### 1.3.2 Программирование разветвляющихся процессов

$$y = \begin{cases} 1+x & при\ x > 0 \\ -1 & при\ x = 0 \\ 1-x^2 & при\ x < 0 \end{cases}$$

С методической целью отобразим решение этой задачи в виде блок-схемы (рис.2).

В том виде, как это отображено на рис.2, блок-схему удобно реализовать на Паскале с помощью оператора **if,** но не на машинном языке или на Ассемблере. Изображенная здесь блок-схема имеет пространственную конфигурацию, в то время как программа на машинном языке имеет чисто линейную структуру. В данном случае удобно применить линейное изображение блок-схемы (рис.3).

Обход линейной последовательности блоков в машинной программе осуществляется с помощью команд перехода.

На первом этапе в машинной программе с разветвлениями не рекомендуется сразу же записывать шестнадцатеричные адреса ячеек памяти, поскольку в процессе разработки велика вероятность добавления и удаления команд. Адреса перехода отмечаются метками Met2..Met4.

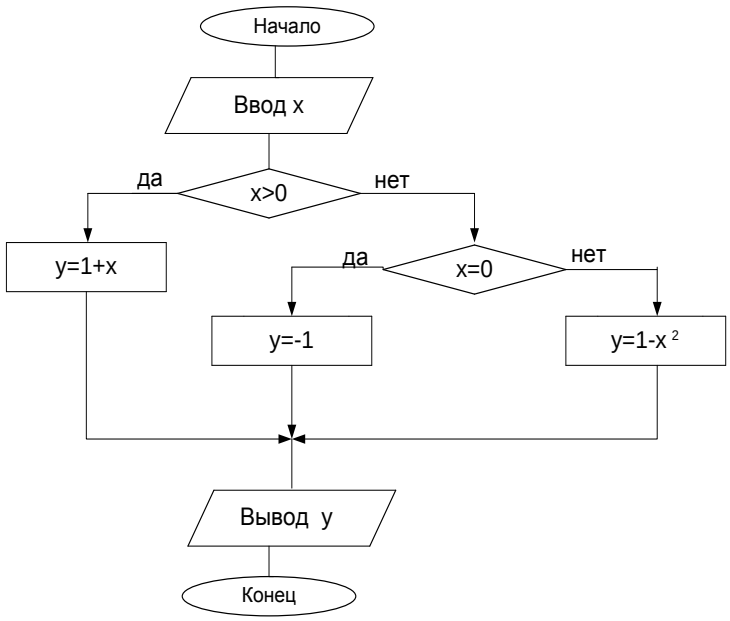


Рисунок 2 – Двухмерная блок-схема задачи с разветвлениями

```
        Mov     0   <x>    R1      x → R1
        Add     1    R1    0       x + 0 → Acc
        JumSZ   0   Met1   Met2    Переход на 2 или 3 (x ≤ 0)
        Add     2   <1>    <y>     x + 1 →< y >
        Jump    0    Met3   0      Переход на Met3
Met1    Mov     0   <x>    R2      x → R2
        Mul     2   <x>    R2      x^2 → R2 + Acc
        Sub     3   <1>    R2      1 − x^2 → Acc
        Add     3    0     <y>     1 − x^2 →< y >
        Jump    0   Met3    0      Переход на Met3
Met2    Mov     0   <-1>   <y>     −1 →< y >
Met3    Halt
```

```
 1
-1
x
y
R1
R2
```

Маски в командах перехода представлены в шестнадцатеричном виде.

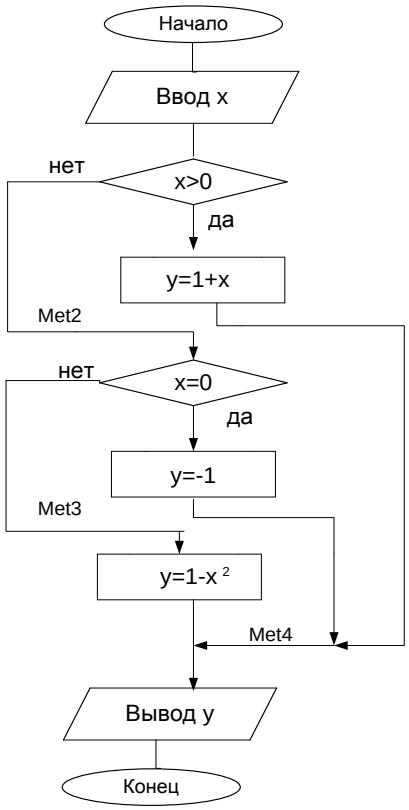


Рисунок 3 – Линейная блок-схема задачи с разветвлениями

Второй этап – запись адресов ячеек памяти.

```
55        Mov    0  <x>   R1
56        Add    1   R1   0
57        JumSZ  0  Met1  Met2
58        Add    2  <1>   <y>
59        Jump  0   Met3   0
5A Met1   Mov    0  <x>    R2
5B        Mul    2  <x>    R2
5C        Sub    3  <1>    R2
5D        Add    3   0     <y>
5E        Jump   0  Met3    0
5F Met2   Mov    0  <-1>   <y>
60 Met3   Halt
61         1
```

```
62        -1
63         x
64         y
65         R1
66         R2
```

Этапы 3 и 4 аналогичны этапам 2 и 3 предыдущей программы.

В п.1.3.2 дополнительно проверены команды Jump и JumSZ.

### 1.3.3 Организация циклической программы

В системе команд ЭВМ С1 нет отдельной команды управления циклом, а для доступа к ячейкам памяти используется лишь прямая адресация. В связи с этим для формирования цикла будем использовать команду условной передачи по S и Z, а модификацию изменяемого адреса в команде будем выполнять принудительным образом.

В качестве примера реализации циклической программы рассмотрим вычисление выражения

$$y = 8a - \sum_{i=1}^{5} x_i$$

При этом предполагается, что элементы массива *X* расположены в смежных ячейках памяти.

Представим заданное выражение в следующем виде:

$$y = 8a - R; \quad R = \sum_{i=1}^{5} x_i$$

Тогда фрагмент программы может иметь следующий вид:

| | | | |
|---|---|---|---|
| k+0 | 0001 00 00000000 < R > | $0 \rightarrow < R >$ |
| k+1 | 0001 00 00000000 < сч-к > | $0 \rightarrow <$ *счетчик* $>$ |
| k+2 | 0001 00 k+11 k+3 | *восстановл ение команды* |
| k+3 | 0010 00 $< x_1^* >$ < R > | $y := y + x_i$ |
| k+4 | 0010 00 k+12 k+3 | *модификаци я команды* |
| k+5 | 0010 00 < 1 > < сч-к > | *счетчик* $+1 \rightarrow <$ *счетчик* $>$ |
| k+6 | 0011 01 < сч-к > < 5 > | *счетчик* $-5 \rightarrow AC$ |
| k+7 | 1110 00 k+3 k+8 | *управление циклом* |
| k+8 | 0100 01 < 8 > < a > | $8 \cdot a \rightarrow AC$ |
| k+9 | 0011 10 < R > < y > | $y \rightarrow < y >$ |
| k+10 | 0000 00 00000000 00000000 | *останов* |
| k+11 | 0010 00 $< x_1 >$ < R > | *исходный вид команды* |
| k+12 | 0000 00 00000001 00000000 | *константа переадреса ции* |

```
k+13  0000 00 00000000 00000101  5
k+14  0000 00 00000000 00000001  1
k+15  0000 00 00000000 00001000  8
k+16           a = 31
k+17           x1 = 10
k+18           x2 = -15
k+19           x3 = 25
k+20           x4 = 40
k+21           x5 = 50
k+22           y
k+23        счетчик
k+24           R
```

Здесь *k, k*+1, … - условное обозначение адреса команды, < *y* > - обозначение адреса ячейки, отведенной для переменной *y* (в данном случае *k*+22).

Первая команда программы обнуляет ячейку, предназначенную для накапливаемой суммы *R*. Обнуляется также счетчик повторений цикла (ячейка *k*+23).

В команде суммирования элементов массива *X* (команда с адресом *k*+3) должен изменяться первый адрес, что приводит к соответствующему изменению самой команды. Поэтому до начала цикла производится восстановление этой команды путем засылки в ячейку *k*+3 первоначального вида команды из ячейки *k*+11.

После добавления к переменной *R* значения элемента $x_i$ первый адрес команды *k*+3 увеличивается на единицу, для чего используется константа переадресации из ячейки *k*+12. Счетчик циклов также увеличивается на единицу, после чего от текущего значения счетчика отнимается его конечное значение 5. Если при этом получен отрицательный результат, то команда условного перехода передает управление на начало цикла, в противном случае – следующей команде.

Из мнемонических соображений изменяемый адрес в команде k+3 обозначен символом «*» с тем, чтобы четко организовать модификацию и восстановление изменяемой команды.

Предположим, что пусковой адрес программы задан равным $k = 20_{10} = 14_{16}$. Тогда текстовый файл, подготовленный для загрузки машинных команд и обрабатываемых данных в память MEMORY, будет иметь следующий вид (пусковой адрес представлен в 10 с/с, остальная информация – в 2 с/с и параллельно в 16 с/с):

```
          20                  20
0001000000000000101100    04002C
```

```
0001000000000000101011    04002B
0001000001111100010111    041F17
0010000010010100101100    08252C
0010000010000000010111    082017
0010000010001000101011    08222B
0011010010101100100001    0D2B21
1110000001011100011100    38171C
0100010010001100100100    112324
0011100010110000101010    0E2C2A
0000000000000000000000    000000
0010000010010100101100    08252C
0000000000000100000000    000100
0000000000000000000101    000005
0000000000000000000001    000001
0000000000000000001000    000008
0000000000000000011111    00001F
0000000000000000001010    00000A
1111111111111111110001    3FFFF1
0000000000000000011001    000019
0000000000000000101000    000028
0000000000000000110010    000032
```

При формировании шестнадцатеричного представления программы двоичная запись каждой команды разделена на тетрады справа налево. Неполная левая тетрада дополнена двумя незначащими нулями.

*Примечание.* В состав текстового файла не включено содержимое ячеек <y>, <счетчик> и <R>, поскольку это содержимое формируется в программе, а не вводится извне.

### 1.3.4 Программирование подпрограмм

В систему команд ЭВМ С1 входят вызов подпрограммы и возврат из подпрограммы, поэтому реализация подпрограммы выполняется сравнительно просто.

После реализации пп. 1.3.1, 1.3.2 и 1.3.3 остались непроверенными команды And, Or, Xor, Shr, Shl, Call и Ret.

Пусть нам требуется вычислить

$$u = (a \leftarrow^{3} \wedge b) \rightarrow^{2} \vee c \oplus d$$

$$v = (e \leftarrow^{3} \wedge f) \rightarrow^{2} \vee g \oplus h$$

Вычисления будем производить в подпрограмме, реализующий опера-

тор

$$w = (l \leftarrow^{3} \wedge m) \rightarrow^{2} \vee n \oplus p$$

```
        Mov   0  <a>  <l>
        Mov   0  <b>  <m>
        Mov   0  <c>  <n>
        Mov   0  <d>  <p>
        Call  0  Met1  R1  Обращение к подпрограмме
        Mov   0  <w>  <u>
        Mov   0  <e>  <l>
        Mov   0  <f>  <m>
        Mov   0  <g>  <n>
        Mov   0  <h>  <p>
        Call  0  Met1  R1  Обращение к подпрограмме
        Mov   0  <w>  <v>
        Halt
Met1    Mov   0  <l>   R2  l → R2
        Shl   0   R2   3   (R2) ←3
        And   3   <m>  R2  (R2)∧ < m >→ R2
        Shr   0   R2   2   (R2) →2→ Acc
        Or    3   <n>  R2  (Acc)∨ < n >→ R2
        Xor   3   <p>  R2  (Acc)⊕ < p >→ R2
        And   2    0  <w>  (Acc) →< w >
        Ret   0   R1   0   Возврат из подпрограммы
            a
            b
            c
            d
            e
            f
            g
            h
            l
            m
            n
            p
            u
            v
            w
            R1
            R2
```

Дальнейшие этапы выполняются аналогично предыдущему.

Для контроля правильности работы рассматриваемой программы выполним расчет значения параметра *u* для конкретных величин переменных *a*, *b*, *c*, *d*.

$a = 00\ 0001\ 0010\ 0011\ 0100\ 0110_2 = 012346_{16}$

$b = 10\ 0111\ 1000\ 1001\ 1010\ 1100_2 = 2789AC_{16}$

$c = 01\ 1101\ 1110\ 1111\ 1110\ 1100_2 = 1DEFEC_{16}$

$d = 11\ 1100\ 1111\ 1110\ 1100\ 1010_2 = 3EFEDA_{16}$

$a \leftarrow^3 = 00\ 1001\ 0001\ 1010\ 0011\ 0000_2 = 091A30_{16}$

$(a \leftarrow^3) \wedge b = 00\ 0001\ 0000\ 1000\ 0010\ 0000_2 = 010820_{16}$

$(a \leftarrow^3) \wedge b \rightarrow^2 = 00\ 0000\ 0100\ 0010\ 0000\ 1000_2 = 004208_{16}$

$(a \leftarrow^3) \wedge b \rightarrow^2 \vee c = 01\ 1101\ 1110\ 1111\ 1110\ 1100_2 = 1DEFEC_{16}$

$u = (a \leftarrow^3) \wedge b \rightarrow^2 \vee c \oplus d = 10\ 0010\ 0001\ 0001\ 0011\ 0110_2 = 221136_{16}$

# 2. УЧЕБНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА С2

## 2.1 Архитектура ЭВМ С2

Учебная ЭВМ С2 может быть одноадресной или двухадресной ЭВМ. Адресность определяется модификатором команд МК регистра команд RK. Операнды располагаются в аккумуляторе AC и операционном регистре OR. Результат операции всегда сохраняется в аккумуляторе, а в некоторых случаях может также записываться в память.

Структурная схема ЭВМ С2 приведена на рис.4. В состав ЭВМ С2 входят следующие основные узлы:

- оперативная память MEMORY емкостью 256 24-разрядных слов;
- восьмиразрядный регистр адреса RA, определяющий номер ячейки памяти, с которой происходит взаимодействие в данный момент времени;
- двадцатичетырехразрядный регистр слова RS для временного хранения читаемой или записываемой информации;
- восьмиразрядный счетчик адреса команд SAK, предназначенный для хранения адреса следующей команды;
- двадцатичетырехразрядный регистр команды RK, хранящий выполняемую команду;
- двадцатичетырехразрядный регистр-аккумулятор AC для приема и хранения первого операнда;
- двадцатичетырехразрядный операционный регистр OR, непосредственно используемый при выполнении операции в АЛУ по отношению к заданному второму операнду;
- арифметико-логическое устройство АЛУ, выполняющее операцию, заданную кодом операции команды;
- двадцатичетырехразрядный регистр RR, сохраняющий результат выполнения операции в АЛУ;
- дешифратор кода операции DC;
- сумматор адресов SA, выполняющий модификацию адресной части команды и изменение модификаторов;
- четырехразрядный регистр признаков завершения операции RP.

При выполнении арифметической операции первый бит RP устанавливается в «1», если ее результат отрицательный; второй бит устанавливается в «1» при нулевом результате этой операции; третий бит принимает значение «1» в случае некорректности выполнения операции (переполнение регистра RR или попытка деления на нуль). Эти биты на схеме ЭВМ обозначены как признаки S, Z и C. Изменение состояния четвертого бита регистра RP (признак E) производится только командой «Конец цикла».

Для логических операций и команды вычитания модулей производится лишь анализ результата на равенство нулю: RP[1] = 1, если результат опера-

ции не равен нулю; RP[2] = 1, если этот результат равен нулю.

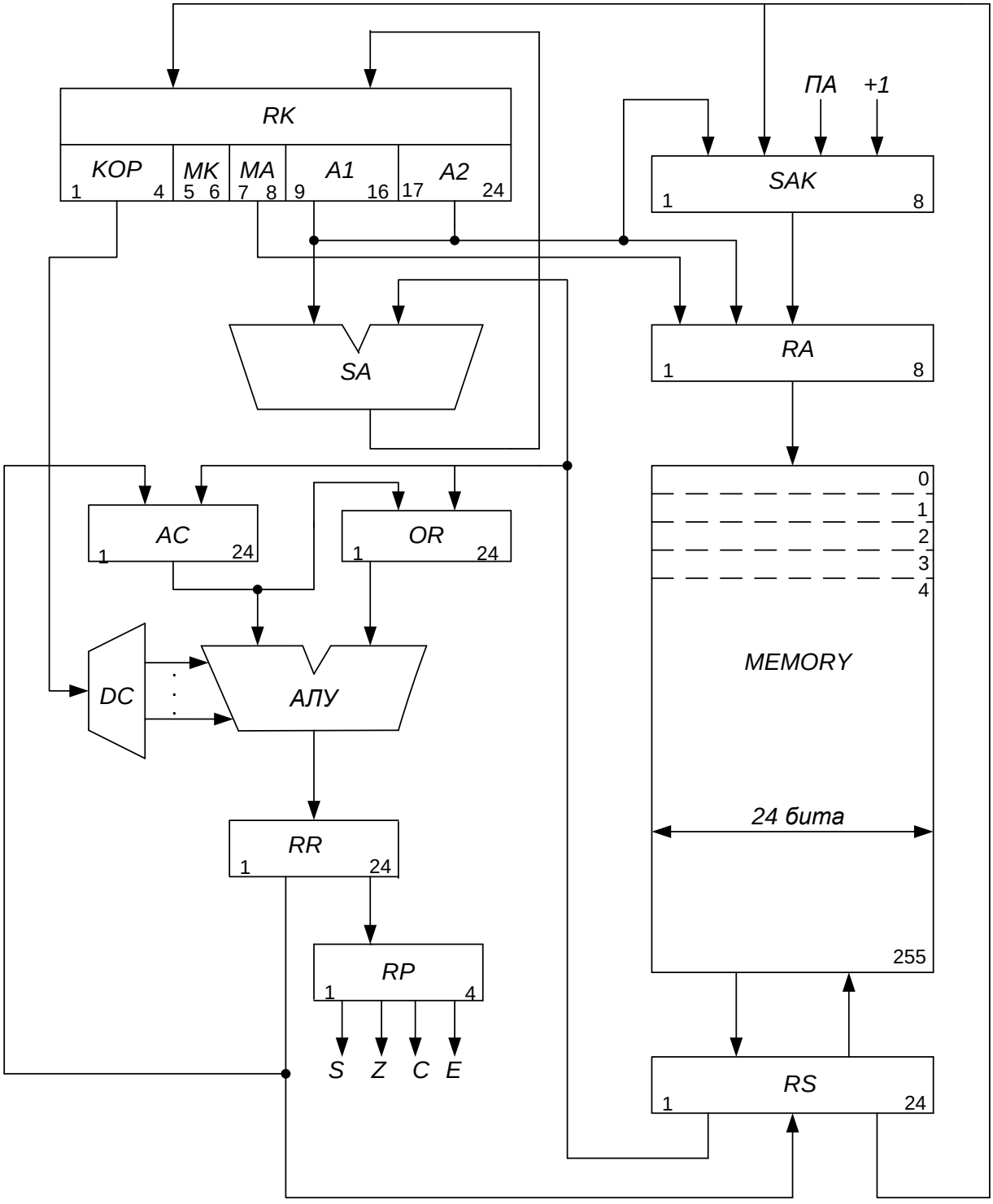


*Рисунок 4 - Структурная схема ЭВМ С2*

Регистр команды разделяется на следующие поля:

- код выполняемой операции КОР (биты 1 – 4),
- модификатор команды МК, определяющий ее адресность (биты 5 – 6);

- модификатор адреса МА (биты 7 – 8),
- исходный адрес первого операнда А1 (биты 9 – 16);
- исходный адрес второго операнда А2 (биты 17 – 24).

Адреса ячеек памяти изменяются от 0 до 255 (от 00000000 до 11111111 в двоичной системе счисления). В ячейке с нулевым адресом постоянно находится нуль, который можно только считывать.

Ячейки памяти с адресами 1, 2, 3 используются для модификации адресов в командах. В битах 1 – 8 этих ячеек записан счетчик повторений цикла SP, в разрядах 9 – 16 и 17 – 24 – модификаторы адресов. Исполнительные адреса формируются путем сложения в сумматоре SA содержимого полей А1 и А2 с соответствующими полями ячейки модификации, имеющей адрес МА:

$$A1_{исп} = A1 + MEMORY\ [MA](9-16);$$
$$A2_{исп} = A2 + MEMORY\ [MA](17-24);$$

Если МА = 0, то $A1_{исп} = A1;\ A2_{исп} = A2.$

Таблица 2. Система команд ЭВМ С2

| Код опер. | Наименование | Признаки | | | | Выполнение команды |
|---|---|---|---|---|---|---|
| | | S | Z | C | E | |
| 0000 | Останов | – | – | – | – | |
| 0001 | Конец цикла | – | – | – | + | См.описание |
| 0010 | Пересылка | – | – | – | – | $(A1_{исп}) \to A2_{исп}$ |
| 0011 | Возврат из подпрогр-ы | – | – | – | – | $(A2_{исп}) \to SAK$ |
| 0100 | Условный переход по S и Z | – | – | – | – | См.описание |
| 0101 | Условный переход по C | – | – | – | – | См.описание |
| 0110 | Безусловный переход | – | – | – | – | $A1_{исп} \to SAK$ |
| 0111 | Вызов подпрограммы | – | – | – | – | $(SAK) \to A2_{исп};\ A1_{исп} \to SAK$ |
| 1000 | Сложение | + | + | + | – | |
| 1001 | Вычитание | + | + | + | – | $MK = 00$ |
| 1010 | Умножение | + | + | + | – | $(A1_{исп}) \otimes (A2_{исп}) \to AC, A2_{исп}$ |
| 1011 | Деление | + | + | + | – | $MK = 01$ |
| 1100 | Вычитание модулей | + | + | – | – | $(A1_{исп}) \otimes (A2_{исп}) \to AC$ |
| 1101 | Сумма по модулю 2 | + | + | – | – | $MK = 10$ |
| 1110 | Конъюнкция | + | + | – | – | $(A1_{исп}) \otimes (AC) \to AC, A2_{исп}$ |
| 1111 | Дизъюнкция | + | + | – | – | $MK = 11$ $(A1_{исп}) \otimes (AC) \to AC$ |

Круглые скобки в графе «Выполнение команды» означают содержимое

регистра или ячейки памяти. Например, запись $A1_{исп}$ – это адрес ячейки, а ($A1_{исп}$) – содержимое ячейки с адресом $A1_{исп}$.

Система команд ЭВМ С2, приведенная в табл.2, включает в себя:
- арифметико-логические команды;
- команду пересылки;
- команды управления порядком выполнения программы.

К арифметико-логическим командам относятся: сложение, вычитание, умножение, деление, вычитание модулей, конъюнкция, дизъюнкция и сумма по модулю 2.

В состав команд управления порядком выполнения программы входят: вызов подпрограммы, возврат из подпрограммы, безусловный переход, условный переход по S и Z, условный переход по С, конец цикла и останов.

В команде «Останов» разряды 5 – 24 не используются.

## 2.2 Выполнение команд в ЭВМ С2

В ЭВМ С2 используются 24-разрядные числа с фиксированной запятой (целые числа со знаком). Отрицательные значения чисел записываются в дополнительном коде.

Адресность выполняемой команды определяется модификатором команды МК (разряды 5 и 6 регистра команд RK). Модификатор МК может принимать следующие значения: 00, 01, 10, 11.

В арифметико-логических операциях первый операнд содержится в аккумуляторе АС, второй – в операционном регистре OR. Результат операции из выходного регистра RR пересылается в аккумулятор, а при значениях МК = 00 и МК = 10 – также в память.

Если МК = 00, то содержимое ячейки с адресом $A1_{исп}$ пересылается в аккумулятор, а ячейки с адресом $A2_{исп}$ – в регистр OR. Результат операции записывается, кроме аккумулятора, также в память по адресу $A2_{исп}$.

Если МК = 01, то исходные операнды, как и ранее, содержатся в АС и OR, но результат операции в память не передается.

Если МК = 10, то содержимое аккумулятора переписывается в регистр OR, а из памяти читается ячейка с адресом $A1_{исп}$ и ее содержимое пересылается в аккумулятор АС. Результат операции записывается в аккумулятор и в память по адресу $A2_{исп}$.

При МК = 11 выполняются аналогичные действия, но результат операции в память не записывается.

Выполнение команды начинается с загрузки содержимого ячейки памяти, адрес которой задан в SAK, в регистр команд RK. При этом значение адреса выполняемой команды из SAK переписывается в регистр адреса RA, по этому адресу читается содержимое соответствующей ячейки памяти

MEMORY и через буферный регистр слова RS пересылается в регистр RK. В последующем из RK выделяются поля КОР, МК, МА, А1 и А2. Для подготовки выборки следующей команды значение счетчика SAK увеличивается на единицу. Дешифратор DC анализирует значение КОР и определяет команду, которая должна выполняться в данный момент (дешифратор имеет 16 выходов по количеству дешифрируемых команд).

В зависимости от значения кода операции КОР дешифратор DC вырабатывает управляющие сигналы, которые определяют тип выполняемой операции.

Далее во всех командах, кроме команд «останов» и «конец цикла», формируются исполнительные адреса операндов. Для этого значение модификатора адресов МА, записанное в исполняемой команде, передается в регистр адреса RA, содержимое ячейки памяти с адресом, заданным в RA, посылается в регистр слова RS, откуда поступает на второй вход сумматора SA. На первый вход сумматора поступает содержимое полей А1 и А2. В сумматоре SA производится суммирование содержимого ячейки модификации, выбранной из памяти, и исходных адресов А1 и А2, при этом перенос из разряда 17 блокируется. Результат суммирования переписывается снова в регистр команды RK в разряды 9 – 24. Таким образом, в полях А1 и А2 регистра команды RK получаются исполнительные адреса операндов.

Если МК = 01 или МК = 11, то сформированный адрес $A2_{исп}$, каким бы он ни был, в данной команде не будет использован.

Если МА = 00, то в этом случае, в отличие от других значений модификатора МА, чтение нулевой ячейки модификации не производится, поскольку в ней находится заранее известное нулевое значение. В этом случае в качестве исполнительных адресов принимаются исходные адреса операндов:

$$A1_{исп} = A1;\ A2_{исп} = A2.$$

На следующем этапе в арифметико-логических командах по значению модификатора команды МК определяется адресность анализируемой команды. Если МК равен 00 или 01, то ЭВМ работает как двухадресная машина. В этом случае используются два исполнительных адреса для определения соответствующих операндов. Если МК принимает значения 10 или 11, то ЭВМ работает как одноадресная машина. В этом случае используется только один исполнительный адрес, по которому определяется первый операнд двухместной операции, вторым операндом выступает содержимое аккумулятора АС. Результат операции записывается в выходной регистр RR и пересылается обратно в аккумулятор АС. Кроме того, при значении МК, равном 00 или 10, результат выполнения операции записывается в ячейку памяти с адресом $A2_{исп}$.

Для выполнения операции, заданной кодом КОР, значения двух операндов должны быть загружены в аккумулятор АС и в операционный регистр OR. С этой целью значение исполнительного адреса первого операнда $A1_{исп}$ передается в регистр RA, затем содержимое ячейки памяти с адресом, определенным в RA, поступает в регистр RS, откуда переписывается в аккумуля-

тор АС. В двухадресной команде значение $A2_{исп}$ передается в RA, содержимое ячейки с соответствующим адресом поступает в RS, откуда переписывается в регистр OR. Если выполняемая команда является одноадресной, то в регистр OR предварительно переписывается содержимое аккумулятора АС.

В арифметико-логических командах содержимое аккумулятора АС и операционного регистра OR поступает на вход арифметико-логического устройства АЛУ. АЛУ выполняет операцию, заданную дешифратором DC, результат операции записывается в выходной регистр RR и в аккумулятор АС, а также в ячейку памяти с адресом $A2_{исп}$, если модификатор команды МК равен 00 или 10.

Результат арифметической операции используется для формирования в регистре RP признака S (S = 1, если результат меньше нуля), Z (Z = 1, если результат равен нулю) или C (C = 1, если отмечена некорректность выполнения операции – переполнение регистра RR или попытка деления на нуль). Требуемые изменения регистра RP отражены в табл.2.

В регистре RP нет бита, устанавливающегося в 1, если результат арифметической операции положительный. Здесь подразумевается, что если результат такой операции неотрицательный и ненулевой, то он положительный. Следовательно, в этом случае первые три бита регистра RP должны быть установлены в нулевое положение. Состояние последнего бита (признак E) изменяет лишь команда «Конец цикла».

Для логических операций и операции вычитания модулей формируются лишь признаки S (S = 1, если результат не равен нулю) и Z (Z = 1, если результат равен нулю).

Символ “–” в графе признаков табл.2 определяет ситуацию, которая не может возникнуть в данной операции (например, некорректность при выполнении операции пересылки), или то, что данная операция не изменяет регистр RP (например, команда безусловного перехода).

Сущность изменения регистра RP рассмотрим на конкретном примере. Предположим, что в программе выполнена команда сложения и при этом получен отрицательный результат. Тогда в состояние «1» должен быть установлен лишь первый бит регистра RP, остальные биты должны быть сброшены на нуль.

*Примечание*. Если в команде, приведенной в табл.6, все три графы признаков отмечены символом “–”, то это означает лишь то, что при этом сохраняются предыдущие значения признаков (но не выполняется сброс их на нуль).

По команде безусловного перехода содержимое поля A1 регистра команд RK, т.е. значение адреса $A1_{исп}$, пересылается в счетчик SAK, после чего происходит выборка новой команды и ее выполнение. Адрес A2 в этой команде не используется.

При выполнении команды условного перехода по S и Z производится анализ значений битов регистра признаков RP. Если S = 1 (для арифметичес-

кой операции это означает, что (АС) < 0, поскольку результат операции всегда записывается в аккумулятор), то происходит передача управления по адресу $A1_{исп}$; если Z = 1, т.е. (АС) = 0, то в SAK передается адрес $A2_{исп}$; если S = 0 и Z = 0, т.е. (АС) > 0, то управление передается следующей команде.

Если в регистре RP установлен признак С = 1, то команда условного перехода по С производит передачу управления по адресу $A1_{исп}$, в противном случае – следующей по порядку команде. При отсутствии команды условного перехода по С никакой реакции на признак С = 1 не происходит.

В реальных ЭВМ команда условного перехода по С определяла бы переход на подпрограмму обработки аварийного прерывания. В ЭВМ С2 по адресу $A1_{исп}$ достаточно вывести на экран сообщение об аварийном прерывании (код операции и адрес команды), после чего произвести останов работы ЭВМ.

*Примечание.* Вполне очевидно, что признаки S и Z не могут быть одновременно равными единице.

При обращении к подпрограмме должны быть выполнены два действия: передача управления на начальный участок подпрограммы и возврат в вызывающую программу после отработки подпрограммы. Эта работа в ЭВМ С2 выполняется двумя командами: «Вызов подпрограммы» и «Возврат из подпрограммы».

В команде «Вызов подпрограммы» вначале в регистр RA заносится адрес $A2_{исп}$ и содержимое счетчика SAK через регистр RS записывается по этому адресу. После этого адрес $A1_{исп}$ переписывается в счетчик SAK, чем достигается переход к выполнению первой команды подпрограммы.

В команде «Возврат из подпрограммы» в регистр RA записывается адрес $A2_{исп}$, содержимое ячейки памяти с этим адресом выбирается в регистр RS, а затем пересылается в счетчик SAK; тем самым восстанавливается адрес команды, сохраненной ранее при обращении к подпрограмме, т.е. команды, записанной в вызывающей программе после команды вызова подпрограммы. Адрес $A1_{исп}$ в данной команде не используется.

В командах управления порядком выполнения программы модификатор МК не используется.

Команда «Конец цикла» применяется для реализации циклических алгоритмов с переадресацией. В этой команде формирование исполнительных адресов не производится. При ее выполнении по адресу МА, принимающему значение 1, 2 или 3, выбирается из памяти MEMORY ячейка модификации, содержимое этой ячейки через регистр RS передается на первый вход сумматора SA. На второй вход сумматора поступает содержимое ячейки памяти с адресом А1 (биты 9 – 24), в разряды 1 – 8 по второму входу поступают единицы. Следовательно, на входы сумматора SA при выполнении команды «Конец цикла» поступают два операнда:

- содержимое ячейки модификации, имеющей адрес МА;
- содержимое ячейки памяти с адресом А1, из которого используются

лишь биты 9 – 24, а биты 1 – 8 заполнены единицами.

В ячейке памяти с адресом A1, задаваемой в команде «Конец цикла», обычно указывают константу переадресации, т.е. шаг изменения адресов операндов.

При осуществлении суммирования в SA переносы из разрядов 9 и 17 блокируются, при этом происходит вычитание единицы из счетчика SP (разряды 1 – 8 модификатора) и увеличение адресной части модификатора. Полученный результат возвращается в память по адресу MA. Если значение SP стало отрицательным, то вырабатывается признак E = 1 и управление передается следующей команде, в противном случае адрес A2 пересылается в счетчик SAK.

Следует отметить, что если в команде «Конец цикла» модификатор MA равен нулю, то при этом также осуществляется работа цикла, но переадресация операндов не выполняется. Этот частный случай может быть использован для программирования циклических задач, в которых не производится обработка массивов и, следовательно, адреса операндов в теле цикла остаются неизменными. Простым примером такой задачи является возвышение в целочисленную степень.

## 2.3 Программирование в кодах ЭВМ С2

На начальном этапе разработки машинной программы целесообразно каждую команду заменить ее условным обозначением.

Машинная команда ЭВМ С2 состоит из следующих частей:
- код операции;
- модификатор команды МК;
- модификатор адреса МА;
- первый адрес А1;
- второй адрес А2.

Для кода операции будем использовать следующие обозначения:

Halt – останов;
Cikl – конец цикла;
Mov – пересылка;
Ret – возврат из подпрограммы;
JumSZ – условный переход по S и Z;
JumC – условный переход по C;
Jump – безусловный переход;
Call – обращение к подпрограмме;
Add – сложение;
Sub – вычитание;
Mul – умножение;
Div – деление;

Smod – вычитание модулей;
Xor – сумма по модулю 2 (исключающее ИЛИ);
And – конъюнкция;
Or – дизъюнкция.

Для модификатора команды, как и для модификатора адреса, будем использовать их непосредственное численное значение: 0, 1, 2, 3.

В адресной части команды может быть:

- условное обозначение адреса;
- переменная или константа, записанные в ячейки памяти.

В последнем случае имя переменной или значение константы будем заключать в угловые скобки.

*Примеры.*

```
а) Mov  0  0  <x>  <y>
```

Пересылка переменной x, расположенной по адресу <x>, в ячейку с адресом <y>.

```
б) Add  1  1  <x>  <y>
```

$(A1_{исп}) + (A2_{исп}) \to Acc$

```
в) JumSZ  9  1  M1  M2
```

Условный переход по адресу Met1 (на метку Met1), если (Acc) < 0, или по адресу Мет2 (на метку Мет2), если (Acc) = 0.

### 2.3.1 Программирование арифметического выражения

$$y = \frac{338 + 25a}{2a - 1}; \quad a = 10$$

В системе команд ЭВМ С2 непосредственная адресация не применяется. Поэтому все константы, входящие в состав расчетной формулы, должны быть записаны в ячейки памяти.

*Примечание.* Поскольку ячейка памяти ЭВМ С2 имеет размер 24 бита, то максимальное значение переменной может быть $2^{23} - 1 = 8\,388\,607$.

```
14   Mul    1  0   <2>    <a>     2a → Acc
15   Add    2  0    0     R1      2a → R1
16   Sub    2  0   <1>    R1      2a − 1 → R1
17   Mul    1  0   <25>   <a>     25a → Acc
18   Add    2  0   <338>  R2      25a + 338 → R2
19   Div    1  0    R1    R2      (25a + 338)/(2a − 1) → Acc
1A   Add    2  0    0     <y>     y → < y >
1B   Halt
1C        1
1D        2
1E        25
1F        338
```

```
20        a = 10
21        R1
22        R2
23        y
```

Здесь в первой колонке – адрес ячейки памяти, содержащей машинную команду, значение переменной или константу. Адрес записан в шестнадцатеричной системе счисления. В данном случае принято, что машинная программа имеет пусковой адрес $14_{16} = 20_{10}$ . Запись (R1) означает содержимое ячейки с номером R1.

Следующий этап – запись машинной команды в двоичном коде (адреса ячеек памяти по-прежнему остаются шестнадцатеричными).

```
14   1010 0100 0001 1101 0010 0000
15   1000 1000 0000 0000 0010 0001
16   1001 1000 0001 1100 0000 0001
17   1010 0100 0001 1110 0010 0000
18   1000 1000 0001 1111 0010 0010
19   1011 0100 0010 0001 0010 0010
1A   1000 1000 0000 0000 0010 0011
1B   0000 0000 0000 0000 0000 0000
1C   0000 0000 0000 0000 0000 0001
1D   0000 0000 0000 0000 0000 0010
1E   0000 0000 0000 0000 0001 1001
1F   0000 0000 0000 0001 0101 0010
20   0000 0000 0000 0000 0000 1010
```

Последний этап – подготовка файла в заданной входной системе счисления (например, восьмеричной).

В первой строке текстового файла записывается пусковой адрес (в десятичной системе). В остальных строках – содержимое ячеек памяти, предварительно разделенное на триады справа налево. Адреса ячеек памяти не отображаются.

```
   20
51016440
42000041
46016041
51017040
42017442
51020442
42000043
00000000
00000001
```

```
00000002
00000031
00000522
00000012
```

Значение переменной *y* как результат вычислений должен быть записан в ячейку с адресом $23_{16} = 35_{10}$ . Этот результат равен $33_{10} = 21_{16} = 41_8$.

В п. 2.3.1 проверены машинные команды Add, Sub, Mul, Div и Halt.

### 2.3.2 Программирование разветвляющихся процессов

$$y = \begin{cases} 1 + x & при\ x > 0 \\ -1 & при\ x = 0 \\ 1 - x^2 & при\ x < 0 \end{cases}$$

Блок-схемное решение рассматриваемой задачи приведено на рис. 2 и 3 в описании ЭВМ С1.

На первом этапе в машинной программе с разветвлениями не рекомендуется сразу же записывать шестнадцатеричные адреса ячеек памяти, поскольку в процессе разработки велика вероятность добавления и удаления команд. Адреса перехода отмечаются метками Met1..Met3.

```
        Mov    0    0    <x>    R      0 → R
        Add    3    0     0     0      x + 0 → Acc
        JumSZ  0    0   Met1   Met2  Переход на Met1 или Met2
        Add    2    0    <1>    <y>    x + 1 →< y >
        Jump   0    0   Met3     0     Переход на Met3
Met1    Mul    2    0    <x>     R      x² → R1 + Acc
        Sub    2    0    <1>    <y>    1 − x² →< y >
        Jump   0    0   Met3     0     Переход на Met3
Met2    Mov    0    0   <-1>    <y>    −1 →< y >
Met3    Halt
            1
            -1
            x
            y
            R
```

В ЭВМ С2 команды пересылки не вырабатывают признаков их выполнения. Поэтому на начальном участке значение *x* складывается с нулем (ко-

манда сложения вырабатывает необходимые признаки).

Второй этап – запись адресов ячеек памяти.

```
51       Mov   0   0   <x>   R
52       Add   3   0    0    0
53       JumSZ 0   0   Met1 Met2
54       Add   2   0   <1>  <y>
55       Jump  0   0   Met3  0
56 Met1  Mul   2   0   <x>   R
57       Sub   2   0   <1>  <y>
58       Jump  0   0   Met3  0
59 Met2  Mov   0   0   <-1>  <y>
5A Met3  Halt
5B           1
5C           -1
5D           x
5E           y
5F           R
```

Этапы 3 и 4 аналогичны этапам 2 и 3 предыдущей программы.

В п.2.3.2 дополнительно проверена команда Mov, Jump и JumSZ.

### 2.3.3 Организация циклической программы

В качестве примера реализации циклической программы рассмотрим вычисление выражения

$$y = 8a - \sum_{i=1}^{5} x_i$$

При этом предполагается, что элементы массива $X$ расположены в смежных ячейках памяти.

Представим заданное выражение в следующем виде:

$$y = 8a - R; \quad R = \sum_{i=1}^{5} x_i$$

Тогда фрагмент программы может иметь следующий вид:

```
k+0  0010 00 00   k+8     00000001  восстановление мод − ра
k+1  0010 00 00 00000000   < R >    0 →< R >
k+2  1000 00 01 00000000   < R >    R → AC
k+3  1000 10 01    < x1* >  < R >   (AC) + xi → AC, < R >
k+4  0001 00 01    k+9       k+2    конец цикла
```

```
k+5   1010 01 00    < 8 >    < a >    8·a → AC
k+6   1001 10 00    < R >    < y >    8a − R →< y >
k+7   0000 00 00 00000000 00000000  останов
k+8   0000 01 00 00000000 00000000  исходное значение
                                    модификато ра
k+9   0000 00 00 00000001 00000000  константа переадреса ции
k+10  0000 00 00 00000000 00001000  8
k+11          a = 31
k+12          x1 = 10
k+13          x2 = -15
k+14          x3 = 25
k+15          x4 = 40
k+16          x5 = 50
k+17          y
k+18          R
```

Здесь $k$, $k$+1, … - условное обозначение адреса команды, < $y$ > - обозначение адреса ячейки, отведенной для переменной $y$ (в данном случае $k$+13).

Поскольку содержимое ячейки модификации изменяется в процессе работы цикла, то вне пределов исполняемой части программы, после команды останова, записано исходное значение модификатора. Первая команда программы засылает это значение в ячейку с адресом 01, которая в данном случае выбрана для модификатора. Вторая команда программы очищает ячейку < R >, предназначенную для накопления суммы элементов $x_i$, посылая туда содержимое нулевой ячейки, т.е. нуль. Непосредственно в теле цикла выполняются две команды:

- пересылка значения переменной $R$ в аккумулятор путем сложения этого значения с нулем и
- добавление к содержимому аккумулятора очередного элемента $x_i$ с последующей записью в ячейку $k+18$ (ячейку $< R >$).

В команде сложения, имеющей адрес $k+3$, записан адрес модификатора 01. Это приводит к формированию исполнительных адресов путем добавления к адресам операндов А1 и А2 адресной части модификатора. В первом цикле адресная часть ячейки модификации равна нулю, но в каждом очередном цикле первый адрес этой ячейки под воздействием команды «Конец цикла» увеличивается на единицу, поскольку в константе переадресации, находящейся по адресу $k+9$, записана единица на месте адреса А1.

Первые 8 разрядов модификатора – это счетчик повторений цикла SP, занимающий разряды 1 – 8. Исходное значение счетчика на единицу меньше заданного количества повторений цикла, что обусловлено особенностями выполнения команды «Конец цикла».

Из мнемонических соображений изменяемый адрес в команде k+3 обо-

значен символом «*» с тем, чтобы четко организовать модификацию и восстановление изменяемой команды.

Предположим, что пусковой адрес программы задан равным $k = 20_{10} = 14_{16}$. Тогда текстовый файл, подготовленный для загрузки машинных команд и обрабатываемых данных в память MEMORY, будет иметь следующий вид (пусковой адрес представлен в 10 с/с, остальная информация – в 2 с/с и параллельно в 16 с/с):

```
          20                      20
00100000000111000000001      201C01
00100000000000000100110      200026
10000001000000000100110      810026
10001001001000000100110      892026
00010001000111010010010      111D12
10100100000111100011111      A41E1F
10011000001001100100101      942625
00000000000000000000000      000000
00000100000000000000000      040000
00000000000000100000000      000100
00000000000000000001000      000008
00000000000000000011111      00001F
00000000000000000001010      00000A
11111111111111111110001      FFFFF1
00000000000000000011001      000019
00000000000000000101000      000028
00000000000000000110010      000032
```

*Примечание.* В состав текстового файла не включено содержимое ячеек <y> и <R>, поскольку это содержимое формируется в программе, а не вводится извне.

### 2.3.5. Программирование подпрограмм

В систему команд ЭВМ С2 входят вызов подпрограммы и возврат из подпрограммы, поэтому реализация подпрограммы выполняется сравнительно просто.

После реализации предыдущих пунктов остались непроверенными команды And, Or, Xor, Smod, Call и Ret.

Пусть нам требуется вычислить

$$u = (|a| - |b|) \wedge c \vee d \oplus e$$

$$v = (|f| - |g|) \wedge h \vee k \oplus l$$

Вычисления будем производить в подпрограмме, реализующий оператор

$$w = (|m| - |n|) \wedge p \vee q \oplus r$$

```
        Mov   0   0  <a>  <m>
        Mov   0   0  <b>  <n>
        Mov   0   0  <c>  <p>
        Mov   0   0  <d>  <q>
        Mov   0   0  <l>  <r>
        Call  0   0  Met1  R   Обращение к подпрограмме
        Mov   0   0  <w>  <u>
        Mov   0   0  <f>  <m>
        Mov   0   0  <g>  <n>
        Mov   0   0  <h>  <p>
        Mov   0   0  <k>  <q>
        Mov   0   0  <l>  <r>
        Call  0   0  Met1  R   Обращение к подпрограмме
        Mov   0   0  <w>  <v>
        Halt
Met1    Smod  0   0  <m>  <n>  |m| - |n| → Acc
        And   3   0  <p>   0   (Acc) ∧ p → Acc
        Or    3   0  <q>   0   (Acc) ∨ <q> → Acc
        Xor   2   0  <r>  <w>  (Acc) ⊕ <r> → <w>
        Ret   0   0   R    0   Возврат из подпрограммы
            a
            b
            c
            d
            e
            f
            g
            h
            k
            l
            m
            n
            p
            q
            r
            u
            v
            w
            R
```

Дальнейшие этапы выполняются аналогично предыдущему.

Для контроля правильности работы рассматриваемой программы выполним расчет значения параметра *u* для конкретных величин переменных *a, b, c, d, e.*

$a = 328585_{10} = 50339_{16} = 0000\ 0101\ 0000\ 0011\ 0011\ 1001_2$

$b = -117466_{10} = E3526_{16} = 1111\ 1110\ 0011\ 0101\ 0010\ 0110_2$

$|a| - |b| = 211119_{10} = 338AF_{16} = 0000\ 0011\ 0011\ 1000\ 1010\ 1111_2$

$c = 0000\ 0001\ 0010\ 0011\ 0100\ 0110_2 = 012346_{16}$

$d = 0010\ 0111\ 1000\ 1001\ 1010\ 1100_2 = 2789AC_{16}$

$e = 0001\ 1101\ 1110\ 1111\ 1110\ 1100_2 = 1DEFEC_{16}$

$\overbrace{|a| - |b|} \wedge c = 0000\ 0001\ 0010\ 0000\ 0000\ 0110_2 = 012006_{16}$

$\overbrace{|a| - |b|} \wedge c \vee d = 0010\ 0111\ 1010\ 1001\ 1010\ 1110_2 = 27A9AE_{16}$

$u = \overbrace{|a| - |b|} \wedge c \vee d \oplus e = 0011\ 1010\ 0100\ 0110\ 0100\ 0010_2 = 3A4642_{16}$

# 3. УЧЕБНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА D1

## 3.1. Архитектура ЭВМ D1

Учебная ЭВМ D1 представляет собой трехадресную ЭВМ с возможностью модификации адресов.

Структурная схема ЭВМ D1 приведена на рис.5. В состав ЭВМ D1 входят следующие основные узлы:

- оперативная память MEMORY емкостью 256 32-разрядных ячеек;
- восьмиразрядный регистр адреса RA, определяющий номер ячейки памяти, с которой происходит взаимодействие в данный момент времени;
- тридцатидвухразрядный регистр слова RS для временного хранения читаемой или записываемой информации;
- восьмиразрядный счетчик адреса команд SAK, предназначенный для хранения адреса следующей команды;
- тридцатидвухразрядный регистр команд RK, хранящий выполняемую команду;
- тридцатидвухразрядные операционные регистры OR1 и OR2 для хранения исходных значений операндов;
- арифметико-логическое устройство АЛУ, выполняющее операцию, заданную кодом операции команды;
- сверхоперативное запоминающее устройство СОЗУ для хранения ячеек модификации;
- сумматор адреса SA для формирования исполнительных адресов операндов и результата;
- тридцатидвухразрядный регистр – аккумулятор AC, сохраняющий результат выполнения операции в АЛУ;
- дешифратор кода операции DC;
- четырехразрядный регистр признаков завершения операции RP.

При выполнении арифметической операции первый бит RP устанавливается в «1», если ее результат отрицательный; второй бит устанавливается в «1» при нулевом результате этой операции; третий бит принимает значение «1» в случае некорректности выполнения операции (переполнение аккумулятора AC или попытка деления на нуль). Состоянием четвертого бита регистра RP управляет команда «Конец цикла». Эти биты на схеме ЭВМ обозначены как признаки S, Z, C и E.

Для логических операций и для команд сдвига производится лишь анализ результата на равенство нулю: RP[1] = 1, если результат операции не равен нулю; RP[2] = 1, если этот результат равен нулю.

Регистр команды разделяется на следующие поля: код выполняемой операции КОР (биты 1 – 4), адрес ячейки модификации M (биты 5 – 8), базовые адреса A1 (биты 9 – 16), A2 (биты 17 – 24) и A3 (биты 25 – 32) первого операнда, второго операнда и результата.

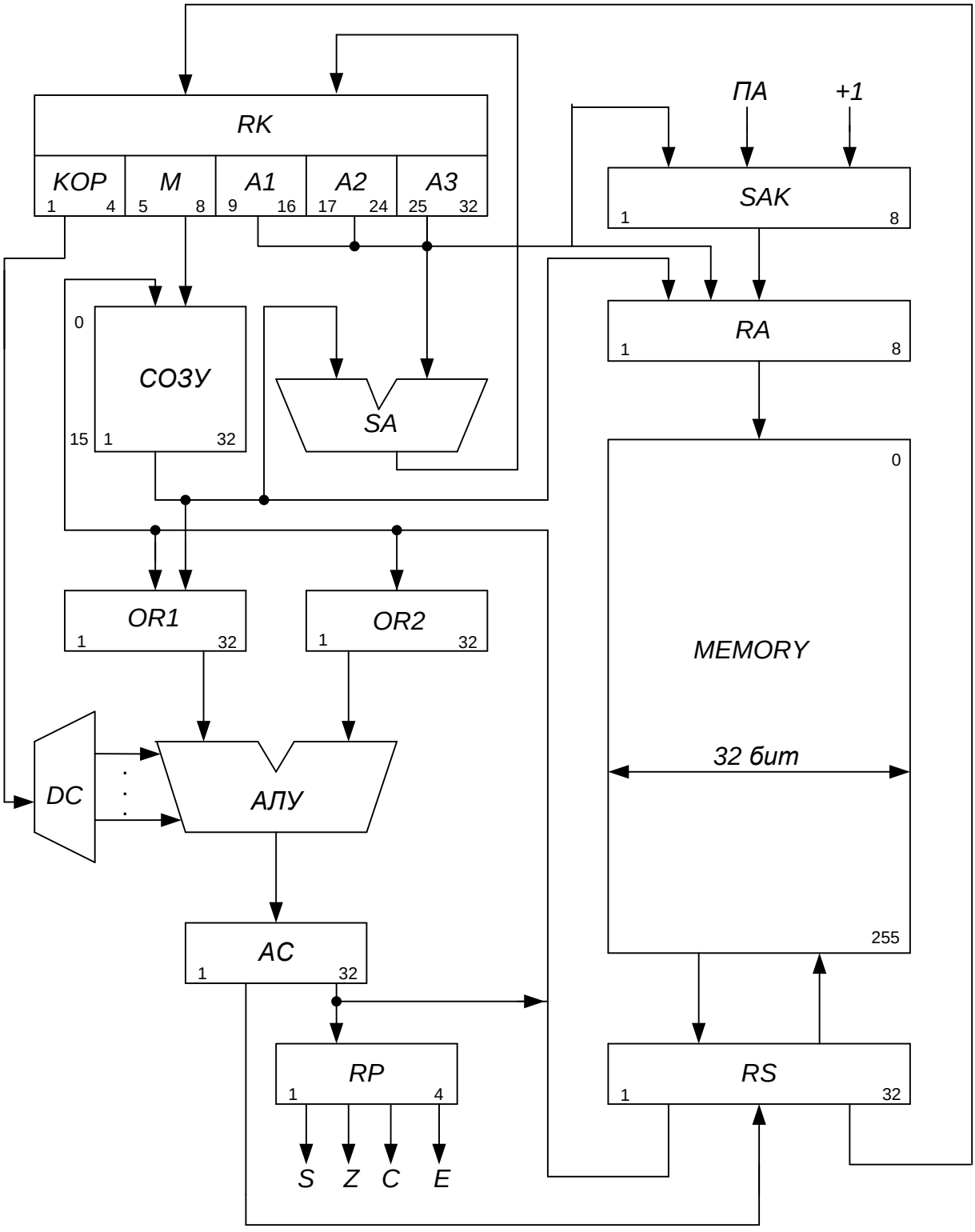


*Рисунок 5 - Структурная схема ЭВМ D1*

Адреса ячеек памяти изменяются от 0 до 255 (от 00000000 до 11111111 в двоичной системе счисления). В ячейке с нулевым адресом постоянно находится нуль, который можно только считывать.

Модификатор адреса М адресует одну из ячеек СОЗУ и может принимать значение от 0 до 15 (от 0000 до 1111 в двоичной системе счисления). В нулевой ячейке СОЗУ всегда хранится нуль.

Структурно содержимое ячейки модификации разделяется на следующие поля: счетчик повторений SP (биты 1 – 8), модификатор адреса первого операнда МА1 (биты 9 – 16), модификатор адреса второго операнда МА2 (биты 17 – 24) и модификатор адреса результата МА3 (биты 25 – 32). Исполнительные адреса, по которым из памяти выбираются значения операндов и записывается результат, формируются путем сложения в сумматоре SA содержимого битов 9 – 32 ячейки модификации с полями А1, А2, А3 регистра команды:

$$A1_{исп} = A1 + (MA1); \quad A2_{исп} = A2 + (MA2); \quad A3_{исп} = A3 + (MA3)$$

Таблица 3. Система команд ЭВМ D1

| Код опер. | Наименование | Признаки | | | | Выполнение команды |
|---|---|---|---|---|---|---|
| | | S | Z | C | E | |
| 0000 | Останов | – | – | – | – | |
| 0001 | Сложение | + | + | + | – | $(A1_{исп}) \otimes (A2_{исп}) \to A3_{исп}$ |
| 0010 | Вычитание | + | + | + | – | |
| 0011 | Умножение | + | + | + | – | |
| 0100 | Деление | + | + | + | – | |
| 0101 | Конъюнкция | + | + | – | – | |
| 0110 | Дизъюнкция | + | + | – | – | |
| 0111 | Сдвиг влево арифм. | + | + | – | – | $(A1_{исп}) \Leftarrow A2_{исп} \to A3_{исп}$ |
| 1000 | Сдвиг вправо арифм. | + | + | – | – | $(A1_{исп}) \Rightarrow A2_{исп} \to A3_{исп}$ |
| 1001 | Возврат из подпр-мы | – | – | – | – | $(A3_{исп}) \to SAK$ |
| 1010 | Вызов подпрограммы | – | – | – | – | $(SAK) \to A3_{исп}$ ; $A1_{исп} \to SAK$ |
| 1011 | Условный переход | – | – | – | – | См. описание |
| 1100 | Безусловный переход | – | – | – | – | $A1_{исп} \to SAK$ |
| 1101 | Конец цикла | – | – | – | + | См. описание |
| 1110 | Сохранение СОЗУ | – | – | – | – | $(M) \to A1$ |
| 1111 | Загрузка СОЗУ | – | – | – | – | $(A1) \to M$ |

Система команд ЭВМ D1, приведенная в табл.3, включает в себя:

- арифметико-логические команды;
- команды сдвига;
- команды обмена информацией с СОЗУ;
- команды управления порядком выполнения программы.

Круглые скобки в графе «Выполнение команды» означают содержимое регистра или ячейки памяти. Например, запись $A1_{исп}$ – это адрес ячейки, а

$(A1_{исп})$ – содержимое ячейки с адресом $A1_{исп}$.

К арифметико-логическим командам относятся: сложение, вычитание, умножение, деление, конъюнкция и дизъюнкция.

Команды сдвига: сдвиг влево арифметический и сдвиг вправо арифметический.

К третьей группе команд относятся команды загрузки и сохранения СОЗУ.

В состав команд управления порядком выполнения программы входят: вызов подпрограммы, возврат из подпрограммы, безусловный переход, условный переход, конец цикла и останов.

В команде «Останов» разряды 5 – 32 не используются.

## 3.2 Выполнение команд в ЭВМ D1

В ЭВМ D1 используются 32-разрядные числа с фиксированной запятой (целые числа со знаком). Отрицательные значения чисел записываются в дополнительном коде.

Выполнение команды начинается с загрузки содержимого ячейки памяти, адрес которой задан в SAK, в регистр команд RK. При этом значение адреса выполняемой команды из SAK переписывается в регистр адреса RA, по этому адресу читается содержимое соответствующей ячейки памяти MEMORY и через буферный регистр слова RS пересылается в регистр RK. В последующем из RK выделяются поля KOP, M, A1, A2 и A3. Для подготовки выборки следующей команды значение счетчика SAK увеличивается на единицу. Дешифратор DC анализирует значение KOP и определяет команду, которая должна выполняться в данный момент (дешифратор имеет 16 выходов по количеству дешифрируемых команд).

Далее во всех командах, кроме команд «Останов», «Конец цикла», «Загрузка СОЗУ» и «Сохранение СОЗУ» выполняется формирование исполнительных адресов $A1_{исп}$, $A2_{исп}$ и $A3_{исп}$. При этом значение поля M (биты 5 – 8 регистра RK) передается в СОЗУ, прочитанное содержимое ячейки модификации передается на первый вход сумматора SA. На второй вход сумматора поступает адресная часть регистра RK (биты 9 – 32). При сложении модификатора с адресной частью регистра RK блокируются переносы из разрядов 17 и 25. Результат сложения через регистр возвращается в биты 9 – 32 регистра команд RK. Следовательно, регистр RK вместо исходных адресов теперь содержит исполнительные адреса операндов и результата.

Если M = 0000, то поскольку при этом заранее известно, что в ячейке СОЗУ с нулевым адресом всегда находится нуль, никакой модификации адресов не производится, содержимое полей A1, A2 и A3 остается прежним. В этом случае можно считать, что

$$A1_{исп} = A1;\quad A2_{исп} = A2;\quad A3_{исп} = A3$$

После формирования исполнительных адресов в арифметико-логических операциях осуществляется выборка из памяти операндов. При

этом значение $A1_{исп}$ (биты 9 – 16 RK) передается в RA, содержимое ячейки MEMORY(RA) через регистр слова RS поступает в операционный регистр OR1. Вслед за этим адрес $A2_{исп}$ (биты 17 – 24 RK) передается в RA и содержимое ячейки памяти MEMORY(RA) через RS записывается в операционный регистр OR2.

Арифметико-логическое устройство АЛУ выполняет операцию, заданную кодом КОР, над содержимым регистров OR1 и OR2. Результат выполнения операции записывается в аккумулятор АС, а затем через регистр RS в память MEMORY по адресу $A3_{исп}$.

В системе команд ЭВМ D1 вместо логического сдвига используется арифметический сдвиг. В этом случае при сдвиге влево сдвигаются все разряды, кроме знакового. Освобождающиеся справа разряды заполняются нулями. При арифметическом сдвиге вправо освобождающиеся слева разряды заполняются значением знакового разряда (0 или 1). Разряды, выходящие за пределы разрядной сетки операнда, в обоих случаях исчезают.

В команде сдвига содержимое ячейки с адресом $A1_{исп}$, как и в арифметико-логических операциях, записывается в операционный регистр OR1. Однако адрес $A2_{исп}$ используется здесь не для выборки операнда из памяти, а для определения количества сдвигов содержимого регистра OR1. Адрес $A2_{исп}$ непосредственно переписывается из битов 17 – 25 регистра команд RK в биты 25 – 32 операционного регистра OR2, после чего в зависимости от заданной команды осуществляется сдвиг содержимого регистра OR1 на $A2_{исп}$ разрядов влево или вправо. Если $A2_{исп} > 31$, то сдвиг разрядов регистра OR1 не выполняется; в этом случае по команде сдвига влево все разряды, кроме знакового, заполняются нулями, а при сдвиге вправо – значением знакового разряда. Результат операции записывается в аккумулятор, а затем в ячейку памяти $A3_{исп}$.

Результат арифметической операции используется для формирования в регистре RP признака S (S = 1, если результат меньше нуля), Z (Z = 1, если результат равен нулю) или C (C = 1, если отмечена некорректность выполнения операции – переполнение аккумулятора или попытка деления на нуль). Требуемые изменения регистра RP отражены в табл.3.

В регистре RP нет бита, устанавливающегося в 1, если результат арифметической операции положительный. Здесь подразумевается, что если результат такой операции неотрицательный и ненулевой, то он положительный. Следовательно, в этом случае первые три бита регистра RP должны быть установлены в нулевое положение. Состояние последнего бита (признак E) изменяет лишь команда «Конец цикла».

Для логических операций и для команд сдвига формируются лишь два признака: S = 1, если результат операции не равен нулю, и Z = 1, если этот результат равен нулю.

Символ “–” в графе признаков табл.3 определяет ситуацию, которая не

может возникнуть в данной операции (например, некорректность при выполнении операции сдвига), или то, что данная операция не изменяет регистр RP (например, команда безусловного перехода).

Сущность изменения регистра RP рассмотрим на конкретном примере. Предположим, что в программе выполнена команда сложения и при этом получен отрицательный результат. Тогда в состояние «1» должен быть установлен лишь первый бит регистра RP, остальные биты должны быть сброшены на нуль.

*Примечание*. Если в команде, приведенной в табл.3, все четыре графы признаков отмечены символом "–", то это означает лишь то, что при этом сохраняются предыдущие значения признаков (но не выполняется сброс их на нуль). Исключением является команда "Конец цикла", которая может изменять лишь признак E.

В команде безусловного перехода (БП) значение $A1_{исп}$ заносится в счетчик SAK. Исполнительные адреса $A2_{исп}$ и $A3_{исп}$ в этой команде не используются и не формируются.

В команде условного перехода (УП) вначале анализируется значение битов регистра RP. Если признак S = 1 (результат операции меньше нуля), то значение $A1_{исп}$ передается в счетчик SAK; если признак Z = 1 (результат операции равен нулю), то значение $A2_{исп}$ передается в SAK; если S = 0 и Z = 0 (результат операции больше нуля), то в SAK передается адрес $A3_{исп}$.

В системе команд ЭВМ D1 нет специальной команды, реагирующей на возникновение признака C = 1. Поэтому при эмуляции машинной программы должны проверяться те команды, при выполнении которых может возникать некорректность решения, и в случае обнаружения значения C = 1 на экран нужно выдавать информацию об аварийном завершении программы, указав при этом код операции и адрес команды, вызвавшей прерывание, после чего произвести останов ЭВМ.

При обращении к подпрограмме должны быть выполнены два действия: передача управления на начальный участок подпрограммы и возврат в вызывающую программу после отработки подпрограммы. Эта работа в ЭВМ D1 выполняется двумя командами: «Вызов подпрограммы» и «Возврат из подпрограммы».

В команде «Вызов подпрограммы» вначале в регистр RA заносится адрес $A3_{исп}$ и содержимое счетчика SAK через регистр RS запоминается в битах 25 – 32 ячейки памяти с адресом, заданным в RA. После этого значение адреса $A1_{исп}$ переписывается в счетчик SAK, чем достигается переход к выполнению первой команды подпрограммы. Адрес $A2_{исп}$ в данной команде не используется.

В команде «Возврат из подпрограммы» в регистр RA записывается ад-

рес $A3_{исп}$, содержимое ячейки памяти с этим адресом выбирается в регистр RS, а затем значения битов 25 – 32 регистра RS пересылаются в счетчик SAK; тем самым восстанавливается адрес команды, сохраненной ранее при обращении к подпрограмме, т.е. команды, записанной в вызывающей программе после команды вызова подпрограммы. Адреса $A1_{исп}$ и $A2_{исп}$ в данной команде не используются.

*Примечание*. Вполне очевидно, что адреса $A3_{исп}$ в командах «Вызов подпрограммы» и «Возврат из подпрограммы» должны быть одинаковыми.

Команда «Загрузка СОЗУ» выполняет пересылку содержимого ячейки памяти MEMORY с адресом A1 в ячейку СОЗУ с адресом M. Обычно эта команда используется для восстановления начального значения ячейки модификации.

Команда «Сохранение СОЗУ» выполняет противоположную работу – пересылку из СОЗУ в MEMORY. Команда применяется, если программа содержит большое количество циклов и для их обеспечения не хватает ячеек модификации, содержащихся в СОЗУ.

В обоих командах обмена информацией между СОЗУ и памятью MEMORY исполнительные адреса не используются и не формируются.

В команде «Конец цикла» формирование исполнительных адресов также не производится. Здесь вначале значение поля M (биты 5 – 8 RK) поступает в СОЗУ, прочитанное значение ячейки модификации загружается в операционный регистр OR1. Следовательно, в регистре OR1 содержится в данный момент изменяемый в дальнейшем модификатор с полями SP, MA1, MA2, MA3. После этого значение адреса A1 команды «Конец цикла» передается в регистр RA, по этому адресу выбирается ячейка памяти, биты 9 – 16, 17 – 24, 25 – 32 которой интерпретируются как константы переадресации (изменения) KM1, KM2, KM3 модификаторов MA1, MA2, MA3. Содержимое указанной ячейки через регистр RS пересылается в биты 9 – 32 операционного регистра OR2. Одновременно в биты 1 – 8 регистра OR2 записываются единицы. При суммировании в АЛУ содержимого регистров OR1 и OR2 переносы между полями блокируются, в результате суммирования происходит вычитание единицы из значения счетчика повторений SP, а модификаторы MA1, MA2, MA3 увеличиваются на значения соответствующих констант KM1, KM2, KM3. Результат суммирования, выполненный в АЛУ, записывается в аккумулятор AC. Если при суммировании получено SP<0, то вырабатывается признак E = 1, в противном случае E = 0. После этого производится анализ признака E. Если E = 0, то содержимое аккумулятора записывается в СОЗУ по адресу, заданному в поле M регистра RK, а в счетчик SAK заносится адрес A2, чем обеспечивается повторное выполнение данного цикла. Если E = 1, то в SAK передается адрес A3, в результате чего происходит выход из цикла.

### 3.3 Программирование в кодах ЭВМ D1

На начальном этапе разработки машинной программы целесообразно каждую команду заменить ее условным обозначением.

Машинная команда ЭВМ D1 состоит из следующих частей:
- код операции;
- адрес ячейки модификации M;
- базовые адреса A1, A2 и A3.

Для кода операции будем использовать следующие обозначения:

Halt – останов;
Add – сложение;
Sub – вычитание;
Mul – умножение;
Div – деление;
And – конъюнкция;
Or – дизъюнкция.
Shl – сдвиг влево арифметический;
Shr – сдвиг вправо арифметический;
Ret – возврат из подпрограммы;
Call – обращение к подпрограмме;
If – условный переход;
Jump – безусловный переход;
Cikl – конец цикла;
Save – сохранение СОЗУ;
Load – загрузка СОЗУ.

В адресной части команды может быть:
- условное обозначение адреса;
- переменная или константа, записанные в ячейки памяти.

В последнем случае имя переменной или значение константы будем заключать в угловые скобки.

*Примеры.*

а) `Add 0 <x> <y> <z>`

Сложение значений переменных *x* и *y* с записью результата в ячейку, предназначенную для переменной *z*.

б) `Shl 0 <x> 3 R1`

Арифметический сдвиг значения переменной *x* на 3 разряда влево с записью результата по адресу *R*1.

в) `Load 5 <x> 0 0`

Пересылка в ячейку 5 СОЗУ значения переменной *x*.

### 3.3.1 Программирование арифметического выражения

$$y = \frac{338 + 25a}{2a - 1}; \quad a = 10$$

В системе команд ЭВМ D1 непосредственная адресация не применяется. Поэтому все константы, входящие в состав расчетной формулы, должны быть записаны в ячейки памяти.

*Примечание.* Поскольку ячейка памяти ЭВМ D1 имеет размер 32 бита, то максимальное значение переменной может быть $2^{31} - 1 = 2\,147\,483\,647$.

```
14   Mul   0   <2>    <a>   R1     2a → R1
15   Sub   0    R1    <1>   R1     2a − 1 → R1
16   Mul   0   <25>   <a>   R2     25a → R2
17   Add   0   <338>   R2   R2     338 + 25a → R2
18   Div   0    R2     R1   <y>    (25a + 338)/(2a − 1) → < y >
19   Halt
1A        1
1B        2
1C        25
1D        338
1E        a = 10
1F        R1
20        R2
21        y
```

Здесь в первой колонке – адрес ячейки памяти, содержащей машинную команду, значение переменной или константу. Адрес записан в шестнадцатеричной системе счисления. В данном случае принято, что машинная программа имеет пусковой адрес $14_{16} = 20_{10}$. Запись (R1) означает содержимое ячейки с номером R1.

Следующий этап – запись машинной команды в двоичном коде (адреса ячеек памяти по-прежнему остаются шестнадцатеричными).

```
14   00110000 00011011 00011110 00011111
15   00100000 00011111 00011010 00011111
16   00110000 00011100 00011110 00100000
17   00010000 00011101 00100000 00100000
18   01000000 00100000 00011111 00100001
19   00000000 00000000 00000000 00000000
1A   00000000 00000000 00000000 00000001
1B   00000000 00000000 00000000 00000010
1C   00000000 00000000 00000000 00011001
1D   00000000 00000000 00000001 01010010
1E   00000000 00000000 00000000 00001010
```

Последний этап – подготовка файла в заданной входной системе счисления (например, восьмеричной).

В первой строке текстового файла записывается пусковой адрес (в десятичной системе). В остальных строках – содержимое ячеек памяти, предварительно разделенное на триады справа налево. Адреса ячеек памяти не отображаются.

```
20
06006617037
04007615037
06007017040
02007220040
10010017441
00000000000
00000000001
00000000002
00000000031
00000000522
00000000012
```

Значение переменной *y* как результат вычислений должен быть записан в ячейку с адресом $21_{16} = 33_{10}$ . Этот результат равен $33_{10} = 21_{16} = 41_8$.

В п. 3.3.1 проверены машинные команды Add, Sub, Mul, Div и Halt.

### 3.3.2 Программирование разветвляющихся процессов

$$y = \begin{cases} 1 + x & при\ x > 0 \\ -1 & при\ x = 0 \\ 1 - x^2 & при\ x < 0 \end{cases}$$

Блок-схемное решение рассматриваемой задачи приведено на рис. 2 и 3 в описании ЭВМ С1.

На первом этапе в машинной программе с разветвлениями не рекомендуется сразу же записывать шестнадцатеричные адреса ячеек памяти, поскольку в процессе разработки велика вероятность добавления и удаления команд. Адреса перехода отмечаются метками Met1..Met3.

В ЭВМ С2 команды пересылки не вырабатывают признаков их выполнения. Поэтому на начальном участке значение *x* складывается с нулем (команда сложения вырабатывает необходимые признаки).

```
      Add  0  <x>    0     R1    x → R1
      If   0  Met1  Met2  Met3  Усл.передача упр-я
Met1  Mul  0  <x>   <x>    R2    x^2 → R2
      Sub  0  <1>    R2    <y>   1+x^2 →< y >
      Jump 0  Met4   0      0    Передача управления
Met2  Add  0  <-1>   0     <y>   −1→< y >
      Jump 0  Met4   0      0    Передача управления
Met3  Add  0  <1>   <x>    <y>   1+ x →< y >
Met4  Halt
        1
       -1
        x
        y
        R1
        R2
```

Второй этап – запись адресов ячеек памяти.

```
51       Add  0  <x>    0     R1
52       If   0  Met1  Met2  Met3
53  Met1 Mul  0  <x>   <x>    R2
54       Sub  0  <1>    R2    <y>
55       Jump 0  Met4   0      0
56  Met2 Add  0  <-1>   0     <y>
57       Jump 0  Met4   0      0
58  Met3 Add  0  <1>   <x>    <y>
59  Met4 Halt
5A        1
5B       -1
5C        x
5D        y
5E        R1
5F        R2
```

Этапы 3 и 4 аналогичны этапам 2 и 3 предыдущей программы.

В п.2.3.2 дополнительно проверена команда Jump и If.

### 3.3.3 Организация циклической программы

В качестве примера реализации циклической программы рассмотрим вычисление выражения

$$y = 8a - \sum_{i=1}^{5} x_i$$

При этом предполагается, что элементы массива *X* расположены в смежных ячейках памяти.

Представим заданное выражение в следующем виде:

$$y = 8a - R; \quad R = \sum_{i=1}^{5} x_i$$

Тогда фрагмент программы может иметь следующий вид:

```
k+0   1111 0001    k+5     00000000 00000000
k+1   0001 0000 00000000 00000000   < R >
k+2   0001 0001    < R >    < x1* >    < R >
k+3   1101 0001    k+8        k+2        k+4
k+4   0011 0000    < 8 >    < a >      < y >
k+5   0010 0000    < y >    < R >      < y >
k+6   0000 0000 00000000 00000000 00000000
k+7   0000 0100 00000000 00000000 00000000
k+8   0000 0000 00000000 00000001 00000000
k+9   0000 0000 00000000 00000000 00001000
k+10                a = 31
k+11                x1 = 10
k+12                x2 = -15
k+13                x3 = 25
k+14                x4 = 40
k+15                x5 = 50
k+16                y
k+17                R
```

Здесь *k, k*+1, … - условное обозначение адреса команды, < *y* > - обозначение адреса ячейки, отведенной для переменной *y* (в данном случае k+16).

По адресу k+7 записано исходное (начальное) значение ячейки модификации. В частности, здесь первые 8 разрядов – это счетчик повторений цикла SP. Поскольку команда «Конец цикла» передает управление следующей команде (производит выход из цикла) лишь при получении отрицательного значения SP, то в исходном представлении ячейки модификации записано значение SP, на единицу меньшее заданного количества повторений цикла (в данном случае 4 вместо 5).

Первая команда программы пересылает начальное значение ячейки модификации в СОЗУ в ячейку с адресом 0001, т.е. производит восстановле-

ние ячейки модификации.

Вторая команда обнуляет ячейку, отведенную для размещения переменной *R*. С этой целью здесь использована команда сложения, в качестве обоих операндов которой используется содержимое нулевой ячейки.

Третья команда добавляет к переменной *R* значение элемента $x_i$. Так как адресная часть ячейки модификации 0001 при первом выполнении данной команды нулевая, то в первом цикле к переменной *R* добавляется элемент $x_1$. В этой команде при каждом выполнении цикла должен увеличиваться на единицу лишь адрес второго операнда. Это учтено в заранее приготовленной константе переадресации, записанной по адресу k+8.

Дальше срабатывает команда «Конец цикла», использующая ячейку модификации 0001. При каждом выполнении этой команды из счетчика SP, содержащегося в ячейке модификации, вычитается единица, а к адресу второго операнда добавляется единица. Повторение цикла обеспечивается переходом на адрес k+2, выход из цикла – переходом на адрес k+4. В команде k+4 выполняется умножение константы 8 на переменную *a* с временным размещением произведения в ячейке <*y*>, после чего из полученного значения вычитается сумма *R*.

Из мнемонических соображений изменяемый адрес в команде k+2 обозначен символом “*”, чтобы четко организовать модификацию изменяемой команды.

Предположим, что пусковой адрес программы задан равным $k = 20_{10} = 14_{16}$. Тогда текстовый файл, подготовленный для загрузки машинных команд и обрабатываемых данных в память MEMORY, будет иметь следующий вид (пусковой адрес представлен в 10 с/с, остальная информация – в 2 с/с и параллельно в 16 с/с):

```
              20                                20
11110001000110010000000000000000    F1190000
00010000000000000000000000100101    10000025
00010001001001010001111100100101    11251F25
11010001000111000001011000011000    D11C1618
00110000000111010001111000100100    301D1E24
00100000001001000010010100100100    20242524
00000000000000000000000000000000    00000000
00000100000000000000000000000000    04000000
00000000000000000000000100000000    00000100
00000000000000000000000000001000    00000008
00000000000000000000000000011111    0000001F
00000000000000000000000000001010    0000000A
11111111111111111111111111110001    FFFFFFF1
00000000000000000000000000011001    00000019
00000000000000000000000000101000    00000028
```

```
00000000000000000000000000110010    00000032
```

*Примечание.* В состав текстового файла не включено содержимое ячеек <y> и <R>, поскольку это содержимое формируется в программе, а не вводится извне.

### 3.3.5. Программирование подпрограмм

В систему команд ЭВМ D1 входят вызов подпрограммы и возврат из подпрограммы, поэтому реализация подпрограммы выполняется сравнительно просто.

После реализации предыдущих пунктов остались непроверенными команды And, Or, Shr, Shl, Call и Ret.

Пусть нам требуется вычислить

$$u = (a \leftarrow^{3} \wedge b) \rightarrow^{2} \vee c$$

$$v = (d \leftarrow^{3} \wedge e) \rightarrow^{2} \vee f$$

Вычисления будем производить в подпрограмме, реализующий оператор

$$w = (l \leftarrow^{3} \wedge m) \rightarrow^{2} \vee n$$

```
        And   0  <a>  0  <l>
        And   0  <b>  0  <m>
        And   0  <c>  0  <n>
        Call  0  Met1 0   R2  Обращение к подпр-ме
        And   0  <w>  0  <u>
        And   0  <d>  0  <l>
        And   0  <e>  0  <m>
        And   0  <f>  0  <n>
        Call  0  Met1 0   R2  Обращение к подпр-ме
        And   0  <w>  0  <v>
        Halt
Met1    Shl   0  <l>  3   R1   (R2) ←3
        And   0   R1 <m>  R1   (R2)∧<m>→R2
        Shr   0   R1  2   R1   (R2) →2→ Acc
        Or    0   R1  <n> <w>  (Acc)∨<n>→R2
        Ret   0   0   0   R2   Возврат из подпр-мы
            a
            b
            c
            d
            e
```

```
f
l
m
n
u
v
w
R1
R2
```

Дальнейшие этапы выполняются аналогично предыдущему.

Для контроля правильности работы рассматриваемой программы выполним расчет значения параметра *u* для конкретных величин переменных *a*, *b*, *c*, *d*.

$a$ = 1110 1010 1100 0001 0010 0011 0100 $0110_2$ = $EAC12346_{16}$

$b$ = 0011 1001 0010 0111 1000 1001 1010 $1100_2$ = $392789AC_{16}$

$c$ = 1001 1111 0001 1101 0000 1111 1110 $1100_2$ = $9F1D0FEC_{16}$

$a \leftarrow^3$ = 1101 0110 0000 1001 0001 1010 0011 $0000_2$ = $D6091A30_{16}$

$(a \leftarrow^3) \wedge b$ = 0001 0000 0000 0001 0000 1000 0010 $0000_2$ = $10010820_{16}$

$(a \leftarrow^3) \wedge b \rightarrow^2$ = 0000 0100 0000 0000 0100 0010 0000 $1000_2$ = $04004208_{16}$

$u = (a \leftarrow^3) \wedge b \rightarrow^2 \vee c$ = 1001 1111 0001 1101 0100 1111 1110 $1100_2$ = $9F1D4FEC_{16}$

# 4. УЧЕБНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА D2

## 4.1 Архитектура ЭВМ D2

Учебная ЭВМ D2 является одноадресной машиной, поэтому при выполнении операций первый операнд в основном находится в регистре-аккумуляторе АС, а второй – в памяти. Результат операции сохраняется в аккумуляторе.

Структурная схема ЭВМ D2 приведена на рис.6. В состав ЭВМ D2 входят следующие основные узлы:

- оперативная память MEMORY емкостью 256 16-разрядных ячеек;
- восьмиразрядный регистр адреса RA, определяющий номер ячейки памяти, с которой происходит взаимодействие в данный момент времени;
- шестнадцатиразрядный регистр слова RS для временного хранения читаемой или записываемой информации;
- восьмиразрядный счетчик адреса команд SAK, предназначенный для хранения адреса следующей команды;
- шестнадцатиразрядный регистр команд RK, хранящий выполняемую команду;
- шестнадцатиразрядный регистр-аккумулятор АС;
- арифметико-логическое устройство АЛУ, выполняющее операцию, заданную кодом операции команды;
- сверхоперативное запоминающее устройство СОЗУ для хранения ячеек модификации;
- сумматор адреса SA, выполняющий модификацию адресной части команды и изменение модификаторов;
- дешифратор кода операции DC;
- четырехразрядный регистр признаков завершения операции RP.

При выполнении арифметической операции первый бит RP устанавливается в «1», если ее результат отрицательный; второй бит устанавливается в «1» при нулевом результате этой операции; третий бит принимает значение «1» при положительном результате, четвертый – в случае некорректности выполнения операции (переполнение аккумулятора или попытка деления на нуль).

Для логических операций и для команд сдвига производится лишь анализ результата на равенство нулю: RP[1] = 1, если результат операции не равен нулю; RP[2] = 1, если этот результат равен нулю.

В состав команды входят код выполняемой операции КОР (биты 1 – 4), адрес ячейки модификации М (биты 5 – 8) и адрес ячейки памяти А (биты 9 – 16).

Адреса ячеек памяти изменяются от 0 до 255 (от 00000000 до 11111111 в двоичной системе счисления). В ячейке с нулевым адресом постоянно находится нуль, который можно только считывать.

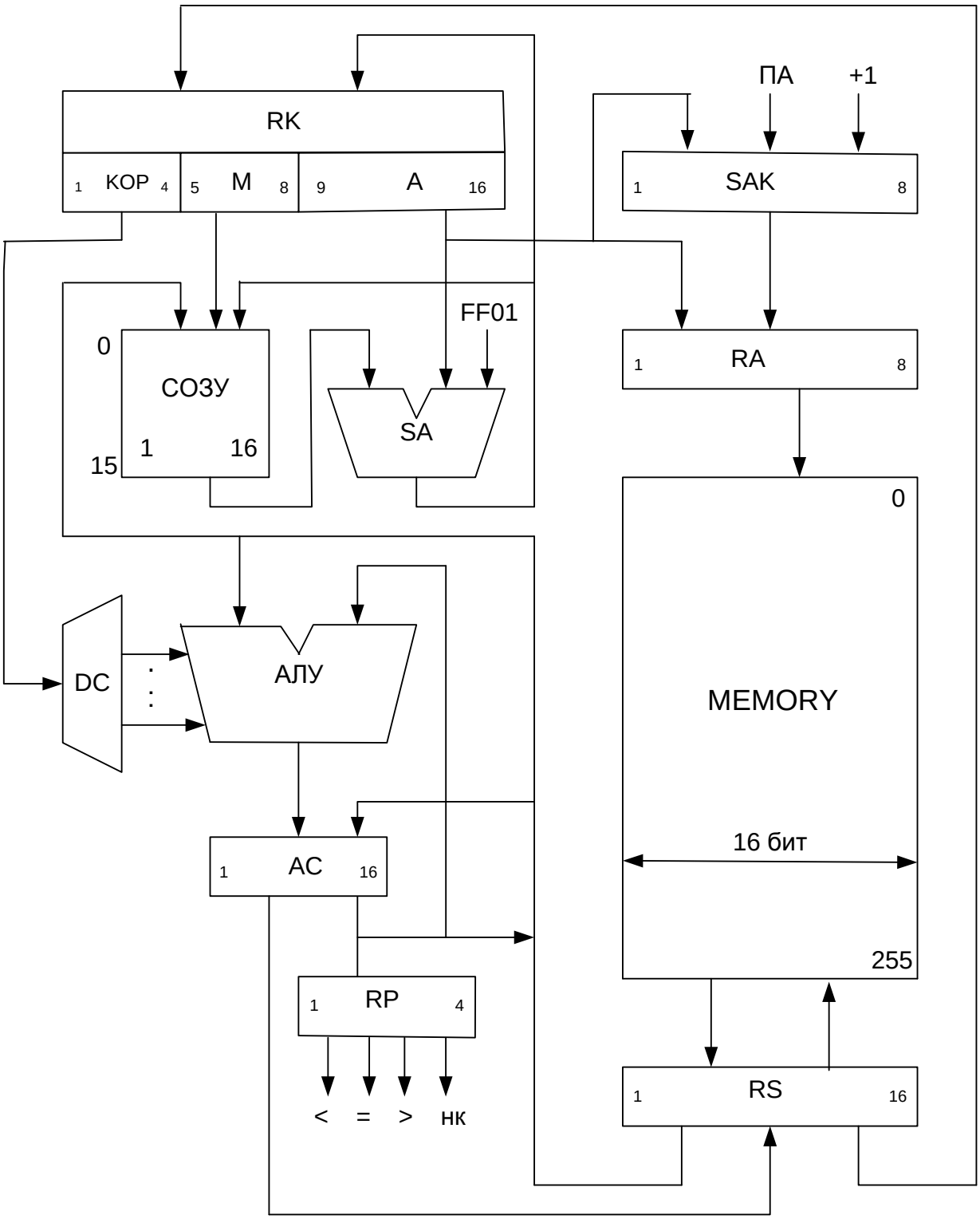


Рисунок 6 - Структурная схема ЭВМ D2

Адрес ячейки модификации M относится к СОЗУ и может принимать значение от 0 до 15 (от 0000 до 1111 в двоичной системе счисления). В нулевой ячейке СОЗУ всегда хранится нуль.

Ячейки модификации с адресами от 1 до 15 содержат в разрядах 9 – 16 модификатор адреса МА, а в разрядах 1 – 8 - счетчик повторений SP. Ис-

полнительный адрес операнда формируется путем сложения в сумматоре SA содержимого поля A и содержимого ячейки СОЗУ, адрес которой записан в поле M регистра команды (индекс-адресация):

$$A_{исп} = A + СОЗУ(M)[9-16].$$

Значение M = 0000 используется для задания адреса операнда без модификации (прямая адресация). В этом случае исполнительный адрес

$$A_{исп} = A.$$

Система команд ЭВМ D2, приведенная в табл.4, включает в себя:

- команды загрузки и сохранения аккумулятора и модификатора;
- арифметико-логические команды;
- команды сдвига;
- команды управления порядком выполнения программы.

Таблица 4. Система команд ЭВМ D2

| Код опер. | Наименование | Признаки | | | | Выполнение команды |
|---|---|---|---|---|---|---|
| | | < | = | > | нк | |
| 0000 | Останов | – | – | – | – | |
| 0001 | Сложение | + | + | + | + | $(AC) \otimes (A_{исп}) \to AC$ |
| 0010 | Вычитание | + | + | + | + | |
| 0011 | Умножение | + | + | + | + | |
| 0100 | Деление | + | + | + | + | |
| 0101 | Сложение по модулю 2 | + | + | – | – | |
| 0110 | Сдвиг вправо логический | + | + | – | – | $(AC) \quad A_{исп} \Rightarrow$ |
| 0111 | Сдвиг влево логический | + | + | – | – | $(AC) \quad A_{исп} \Leftarrow$ |
| 1000 | Загрузка аккумулятора | – | – | – | – | $(A_{исп}) \to AC$ |
| 1001 | Сохранение аккумулятора | – | – | – | – | $(AC) \to A_{исп}$ |
| 1010 | Загрузка модификатора | – | – | – | – | $(A) \to СОЗУ(M)$ |
| 1011 | Сохранение модификатора | – | – | – | – | $СОЗУ(M) \to A$ |
| 1100 | Конец цикла | – | – | – | – | См. описание |
| 1101 | Вызов подпрограммы | – | – | – | – | $(SAK) \to 16;\ A_{исп} \to SAK$ |
| 1110 | Возврат из подпрогр-ы | – | – | – | – | $(16) \to SAK$ |
| 1111 | Переход | – | – | – | – | См.описание |

Круглые скобки в графе «Выполнение команды» означают содержимое регистра или ячейки памяти. Например, запись $A_{исп}$ – это адрес ячейки, а $(A_{исп})$ – содержимое ячейки с адресом $A_{исп}$.

К арифметико-логическим командам относятся: сложение, вычитание,

умножение, деление, сложение по модулю 2.

Для сдвига используются две команды: сдвиг влево логический и сдвиг вправо логический.

Команды загрузки и сохранения: загрузка аккумулятора, сохранение аккумулятора, загрузка модификатора, сохранение модификатора.

В состав команд управления порядком выполнения программы входят: вызов подпрограммы, возврат из подпрограммы, переход, конец цикла и останов.

В команде «Останов» разряды 5 – 16 не используются.

В команде «Вызов подпрограммы» адрес возврата запоминается в ячейке памяти с адресом 16.

Команда «Конец цикла» используется для организации циклических алгоритмов с переадресацией, при выполнении ее происходит изменение содержимого ячейки модификации: из содержимого счетчика повторений SP вычитается единица, а к значению модификатора адреса MA добавляется единица. Если при этом значение SP стало меньше нуля, то выполняется следующая команда. Если значение SP неотрицательное, то переход осуществляется по адресу, заданному в поле A. В частности, это означает, что в команде «Конец цикла» применяется лишь прямая адресация.

## 8.2 Выполнение команд в ЭВМ D2

В ЭВМ D2 используются 16-разрядные числа с фиксированной запятой (целые числа со знаком). Отрицательные значения чисел записываются в дополнительном коде.

Выполнение команды начинается с загрузки содержимого ячейки памяти, адрес которой задан в SAK, в регистр команд RK. При этом значение адреса выполняемой команды из SAK переписывается в регистр адреса RA, по этому адресу читается содержимое соответствующей ячейки памяти MEMORY и через буферный регистр слова RS пересылается в регистр RK. В последующем из RK выделяются поля KOP, M и A. Для подготовки выборки следующей команды значение счетчика SAK увеличивается на единицу. Дешифратор DC анализирует значение KOP и определяет команду, которая должна выполняться в данный момент (дешифратор имеет 16 выходов по количеству дешифрируемых команд).

Далее во всех командах, кроме команд «Останов», «Конец цикла», «Переход», «Загрузка модификатора» и «Сохранение модификатора», выполняется формирование исполнительного адреса $A_{исп}$. При этом значение поля M (биты 5 – 8 регистра RK) передается в СОЗУ, прочитанное содержимое ячейки модификации с адресом M посылается на первый вход сумматора SA. На второй вход сумматора поступает адресная часть A регистра RK (биты 9 – 16). При сложении модификатора с адресной частью регистра RK блокируется перенос из разряда 9. Результат сложения возвращается в биты 9 – 16 регистра команд RK. Следовательно, поле A регистра RK вместо исходно-

го адреса теперь содержит исполнительный адрес операнда.

Если M = 0000, то поскольку при этом заранее известно, что в ячейке СОЗУ с нулевым адресом всегда находится нуль, никакой модификации адреса не производится, содержимое поля A остается прежним. В этом случае можно считать, что

$$A_{исп} = A$$

После формирования $A_{исп}$ в команде «Загрузка аккумулятора» содержимое ячейки памяти, адрес которой записан в регистре RA, выбирается в RS и затем загружается в аккумулятор AC. В команде «Сохранение аккумулятора» содержимое AC через RS записывается в ячейку памяти, адрес которой задан в RA. В выполнении этих двух команд арифметико-логическое устройство не участвует.

В команде логического сдвига вправо (влево) освобождающиеся слева (справа) разряды аккумулятора заполняются нулями, а разряды, выходящие при сдвиге за пределы AC, исчезают. Количество сдвигов определяется значением адреса $A_{исп}$. Если $A_{исп} > 15$, то сдвиг разрядов аккумулятора не выполняется; в этом случае по команде сдвига все разряды в AC заполняются нулями.

В арифметико-логических командах содержимое ячейки памяти с адресом, который задан в регистре RA, через регистр слова RS поступает на первый вход АЛУ, а на второй вход АЛУ передается содержимое аккумулятора AC. Арифметико-логическое устройство выполняет операцию, заданную дешифратором DC, результат операции запоминается в аккумуляторе AC.

Большинство команд, выполняемых в арифметико-логическом устройстве АЛУ, кроме непосредственного формирования своего результата, изменяют также все или часть битов регистра признаков RP, что используется в дальнейшем в АЛУ при выполнении команды перехода. Требуемые изменения регистра RP отражены в табл.4. Признак «нк» (некорректность операции) означает, что в результате выполнения данной операции возникает переполнение аккумулятора или же здесь отмечена попытка деления на нуль.

Символ “–” в графе признаков табл.4 определяет ситуацию, которая не может возникнуть в данной операции (например, некорректность при выполнении операции сдвига), или то, что данная операция не изменяет регистр RP (например, команда перехода).

Сущность изменения регистра RP рассмотрим на конкретном примере. Предположим, что в программе выполнена команда сложения и при этом получен отрицательный результат. Тогда в состояние «1» должен быть установлен лишь первый бит регистра RP, остальные биты должны быть сброшены на нуль.

*Примечание*. Если в команде, приведенной в табл.4, все четыре графы признаков отмечены символом “–”, то это означает лишь то, что при этом сохраняются предыдущие значения признаков (но не выполняется сброс их на нуль).

При обращении к подпрограмме должны быть выполнены два действия: передача управления на начальный участок подпрограммы и возврат в вызывающую программу после отработки подпрограммы. Эта работа в ЭВМ D2 выполняется двумя командами: «Вызов подпрограммы» и «Возврат из подпрограммы».

В команде «Вызов подпрограммы» вначале в регистр RA заносится адрес, равный 16, затем содержимое счетчика SAK через регистр RS запоминается в ячейке памяти с адресом 16. После этого происходит формирование адреса $A_{исп}$ и запись его в счетчик SAK, чем достигается переход к выполнению первой команды подпрограммы.

В команде «Возврат из подпрограммы» в регистр RA записывается значение «16», содержимое ячейки памяти с адресом 16 выбирается в регистр RS, а затем значения битов 9 – 16 регистра RS пересылаются в счетчик SAK; тем самым восстанавливается адрес команды, сохраненной ранее в ячейке 16, т.е. команды, записанной в вызывающей программе после команды обращения к подпрограмме.

В команде перехода поле M считается маской, а не адресом ячейки переадресации. При выполнении этой команды производится логическое умножение маски на содержимое регистра признаков RP. Если результат логического умножения не нулевой, то происходит передача управления по адресу A.

Частные случаи:
- M = 0000 - пустая команда (нет передачи управления);
- M = 1000 - условный переход по знаку «-»;
- M = 0010 - условный переход по нулю;
- M = 1010 - условный переход по знаку «-» или по нулю;
…….
- M = 1111 - безусловный переход.

В команде перехода применяется лишь прямая адресация, поскольку поле M – это маска, а не модификатор адреса.

Некоторой особенностью обладает рассматриваемая команда при значении маски M = 1111. В этом случае логическое умножение маски на содержимое регистра признаков RP не производится и, следовательно, безусловный переход по адресу A выполняется при любом состоянии указанного регистра.

Значение M = 0001 определяет переход по признаку некорректности. В этом случае необходимо по адресу A перейти на вывод сообщения об аварийном прерывании, указав при этом код операции и адрес команды, вызвавшей прерывание, после чего произвести останов ЭВМ.

В команде «Конец цикла» формирование $A_{исп}$ также не производится. Здесь вначале изменяется содержимое ячейки модификации, при этом поле M данной команды передается в СОЗУ, выбирается соответствующая ячейка

модификации и ее содержимое поступает на первый вход сумматора SA. На второй вход сумматора подается код 1111111100000001 ($FF01_{16}$). Сумматор SA выполняет суммирование с блокировкой переноса из разрядов 1 и 9. Это означает, что из содержимого счетчика повторений SP вычитается единица, а к значению модификатора адреса MA добавляется единица. Новое значение ячейки модификации записывается в СОЗУ. Если в результате суммирования получится неотрицательное значение SP, то значение адреса A записывается в счетчик SAK, что приводит к повторному выполнению данного цикла. Если значение SP < 0, то выполняется следующая по порядку команда, т.е. производится выход из цикла.

*Пример.*

```
        SP          MA
    0000 0101  0000 0011
  + 1111 1111  0000 0001
  ---------------------
    0000 0100  0000 0100
```

## 4.3 Программирование в кодах ЭВМ D2

На начальном этапе разработки машинной программы целесообразно каждую команду заменить ее условным обозначением.

Машинная команда ЭВМ D2 состоит из трех частей: код операции, адрес ячейки модификации адреса и адрес операнда.

Для кода операции будем использовать следующие обозначения:

Halt – останов;
Add – сложение;
Sub – вычитание;
Mul – умножение;
Div – деление;
Xor – сложение по модулю 2;
Shr – сдвиг вправо логический;
Shl – сдвиг влево логический;
Load – загрузка акумулятора;
Save – сохранение аккумулятора;
Lmod – загрузка модификатора;
Smod – сохранение модификатора;
Cicl – конец цикла;
Call – обращение к подпрограмме;
Ret – возврат из подпрограммы;
Jump – переход.

Для адреса ячейки модификации будем использовать непосредственно ее значение в диапазоне от 0 до F.

В адресной части команды может быть:

- условное обозначение адреса;
- переменная или константа, записанные в ячейки памяти.

В последнем случае имя переменной или значение команды будем заключать в угловые скобки.

*Примеры.*

а) Save 0 R

Содержимое аккумулятора записывается в память по адресу R (в буферную ячейку R).

б) Jump 0 Met1

Безусловный переход по адресу Met1 (на метку Met1).

в) Add 1 <x>

В аккумулятор добавляется содержимое ячейки, в которой записано значение переменной *x* (с учетом значения модификатора адреса в ячейке 01).

### 4.3.1. Программирование арифметического выражения

$$y = \frac{338 + 25a}{2a - 1}; \quad a = 10$$

```
14   Load    0    <a>      a → Acc
15   Mul     0    <2>      2a → Acc
16   Sub     0    <1>      2a − 1 → Acc
17   Save    0    R        2a − 1 → R
18   Load    0    <a>      a → Acc
19   Mul     0    <25>     25 a → Acc
1A   Add     0    <338>    25a + 338 → Acc
1B   Div     0    R        (Acc)/(R) → Acc
1C   Save    0    <y>      (Acc) → < y >
1D   Halt
1E        a = 10
1F        338
20        1
21        2
22        25
23        R
24        y
```

Здесь в первой колонке – адрес ячейки памяти, содержащей машинную

команду, значение переменной или константу. Адрес записан в шестнадцатеричной системе счисления. В данном случае принято, что машинная программа имеет пусковой адрес $14_{16} = 20_{10}$ .

Следующий этап – запись машинной команды в двоичном коде (адреса ячеек памяти по-прежнему остаются шестнадцатеричными).

```
14   1000 0000 0001 1110
15   0011 0000 0010 0001
16   0010 0000 0010 0000
17   1001 0000 0010 0011
18   1000 0000 0001 1110
19   0011 0000 0010 0010
1A   0001 0000 0001 1111
1B   0100 0000 0010 0011
1C   1001 0000 0010 0100
1D   0000 0000 0000 0000
1E   0000 0000 0000 1010
1F   0000 0001 0101 0010
20   0000 0000 0000 0001
21   0000 0000 0000 0010
22   0000 0000 0001 1001
```

Здесь адреса 23 и 24 не отображаются, поскольку значение переменной *y* и содержимое буферной ячейки *R* формируются в процессе работы программы.

Последний этап – подготовка файла в заданной входной системе счисления (например, восьмеричной).

В первой строке текстового файла записывается пусковой адрес (в десятичной системе). В остальных строках – содержимое ячеек памяти, предварительно разделенное на триады справа налево. Адреса ячеек памяти не отображаются.

```
  20
100036
030041
020040
110043
100036
030042
010037
040043
110044
000000
000012
```

```
000522
000001
000002
000031
```

Значение переменной *y* как результат вычислений должно быть записано в ячейку с адресом $24_{16} = 36_{10}$ . Этот результат равен $33_{10} = 21_{16} = 41_8$.

В п. 4.3.1 проверены машинные команды Load, Save, Add, Sub, Mul, Div, Halt.

### 4.3.2. Программирование разветвляющихся процессов

$$y = \begin{cases} 1 + x & при\ x > 0 \\ -1 & при\ x = 0 \\ 1 - x^2 & при\ x < 0 \end{cases}$$

Блок-схемное решение поставленной задачи приведено на рис.2 и 3 в описании ЭВМ С1.

На первом этапе в машинной программе с разветвлениями не рекомендуется сразу же записывать шестнадцатеричные адреса ячеек памяти, поскольку в процессе разработки велика вероятность добавления и удаления команд. Адреса перехода отмечаются метками Met1..Met3.

```
        Load    0  <x>    x → Acc
        Add     0   0     x + 0 → Acc
        Jump    8  Met1   УП по (Acc)<0
        Jump    4  Met2   УП по (Acc)=0
        Add     0  <1>    x + 1 → Acc
        Jump    F  Met3   БП на Met3
Met1    Mul     0  <x>    x^2 → Acc
        Save    0   R     x^2 → R
        Load    0  <1>    1 → Acc
        Sub     0   R     1 - x^2 → Acc
        Jump    F  Met3   БП на Met3
Met2    Load    0  <-1>   -1 → Acc
Met3    Save     0 <y>    y → <y>
        Halt
            1
            -1
            x
            R
```

```
                Y
```

Здесь Met1, Met2, Met3 – условное обозначение адреса команды, < *y* > - обозначение адреса ячейки, отведенной для переменной *y*, (*Acc*) – содержимое аккумулятора, *R* – адрес буферной ячейки.

Поскольку команда загрузки аккумулятора не вырабатывает признака завершения операции, то после загрузки значения переменной *x* производится сложение этого значения с нулем, при этом будет установлен в единичное состояние первый второй или третий бит регистра RP.

Второй этап – запись адресов ячеек памяти.

```
55          Load    0  <x>
56          Add     0   0
57          Jump    8  Met1
58          Jump    4  Met2
59          Add     0  <1>
5A          Jump    F  Met3
5B  Met1    Mul     0  <x>
5C          Save    0   R
5D          Load    0  <1>
5E          Sub     0   R
5F          Jump    F  Met3
60  Met2    Load    0  <-1>
61  Met3    Save      0 <y>
62          Halt
63             1
64            -1
65             x
```

Этапы 3 и 4 аналогичны этапам 2 и 3 предыдущей программы.

В п.4.3.2 дополнительно проверена команда Jump.

### 4.3.3. Организация циклической программы

Наличие индекс-адресации позволяет легко реализовать циклическую программу. В качестве примера рассмотрим программирование выражения

$$y = 8a - \sum_{i=1}^{5} x_i$$

При этом предполагается, что элементы массива *X* расположены в смежных ячейках памяти.

Представим заданное выражение в виде

$$y = 8a - S, \quad S = \sum_{i=1}^{5} x_i$$

Тогда фрагмент программы может иметь следующий вид:

| | | | |
|---|---|---|---|
| k+0 | 1000 | 0000 00000000 | $0 \to AC$ |
| k+1 | 1001 | 0000 < R > | $0 \to < R >$ |
| k+2 | 1010 | 0001 <константа> | *константа* $\to$ *СОЗУ* (*яч*.1) |
| k+3 | 1000 | 0000 < R > | $R \to AC$ |
| k+4 | 0001 | 0001 < $x_1^*$ > | $R + x_i \to AC$ |
| k+5 | 1001 | 0000 < R > | $R \to < R >$ |
| k+6 | 1100 | 0001 k+4 | *управление циклом* |
| k+7 | 1000 | 0000 < 8 > | $8 \to AC$ |
| k+8 | 0011 | 0000 < a > | $8 \cdot a \to AC$ |
| k+9 | 0010 | 0000 < R > | $8a + R \to AC$ |
| k+10 | 1001 | 0000 < y > | $y \to < y >$ |
| k+11 | 0000 | 0000 00000000 | *останов* |
| k+12 | 0000 | 0100 00000000 | *константа* |
| k+13 | 0000 | 0000 00001000 | 8 |
| k+14 | | a = 31 | |
| k+15 | | $x_1$ = 10 | |
| k+16 | | $x_2$ = -15 | |
| k+17 | | $x_3$ = 25 | |
| k+18 | | $x_4$ = 40 | |
| k+19 | | $x_5$ = 50 | |
| k+20 | | y | |
| k+21 | | R | |

Здесь *k, k*+1, … - условное обозначение адреса команды, < *y* > - обозначение адреса ячейки, отведенной для переменной *y*, (АС) – содержимое аккумулятора АС.

Первые две команды очищают ячейку <*R*>, предназначенную для накопления суммы элементов $x_i$ (вначале нуль заносится в аккумулятор, а затем содержимое аккумулятора записывается в ячейку <*R*>).

По адресу *k*+12 записана константа, определяющая начальное содержимое ячейки модификации. Это содержимое передается в ячейку 0001 СОЗУ, выбранную в программе в качестве ячейки модификации.

Поскольку команда «Конец цикла» передает управление следующей команде (производит выход из цикла) лишь при получении отрицательного значения счетчика циклов SP в ячейке модификации (в данном случае эта ячейка имеет адрес 0001), то в константу, содержащую исходное значение ячейки модификации, в разряды 1 – 8 записывается значение SP, на единицу

меньшее количества повторений цикла (4 вместо 5).

В команде сложения записан адрес элемента $x_1$, который при каждом повторении цикла должен увеличиваться на единицу. С тем, чтобы при разработке машинной программы не забыть выполнить такое изменение адреса операнда, рекомендуется переадресуемый операнд в его условной записи обозначить символом «*». Тогда это будет означать, что в данной команде применяется индекс-адресация, а константа переадресации, имеющая начальное нулевое значение, в конце каждого цикла должна увеличиваться на единицу.

Предположим, что пусковой адрес программы задан равным $k = 20_{10} = 14_{16}$. Тогда текстовый файл, подготовленный для загрузки машинных команд и обрабатываемых данных в память MEMORY, будет иметь следующий вид (пусковой адрес представлен в 10 с/с, остальная информация – в 2 с/с и параллельно в 16 с/с):

```
      20              20
1000000000000000     8000
1001000000101001     9029
1010000100100000     A120
1000000000101001     8029
0001000100100011     1123
1001000000101001     9029
1100000100011000     C118
1000000000100001     8021
0011000000100010     3022
0010000000101001     2029
1001000000101000     9028
0000000000000000     0000
0000010000000000     0400
0000000000001000     0008
0000000000011111     001F
0000000000001010     000A
1111111111110001     FFF1
0000000000011001     0019
0000000000101000     0028
0000000000110010     0032
```

*Примечание.* В состав текстового файла не включено содержимое ячеек <y> и <R>, поскольку это содержимое формируется в программе, а не вводится извне.

### 4.3.4. Программирование подпрограмм

После реализации предыдущих пунктов остались непроверенными ко-

манды Xor, Shr, Shl, Call и Ret.

Пусть нам требуется вычислить

$$u = (a \to^3 \oplus b) \to^2$$

$$v = (c \to^3 \oplus d) \to^2$$

Вычисления будем производить в подпрограмме, реализующий оператор

$$w = (m \to^3 \oplus n) \to^2$$

```
        Load   0  <a>
        Save   0  <m>          m:=a
        Load   0  <b>
        Save   0  <n>          n:=b
        Call   Met1        Обращение к п/п
        Load   0  <w>
        Save   0  <u>          u:=w
        Load   0  <c>
        Save   0  <m>          m:=c
        Load   0  <d>
        Save   0  <n>          n:=c
        Call   Met1        Обращение к п/п
        Load   0  <w>
        Save   0  <v>          v:=w
        Halt
Met1    Load   0  <m>          m → Acc
        Shr    0  3        (Acc) →^3
        Xor    0  <n>          (Acc) ⊕ m → Acc
        Shl    0  2        (Acc) →^2
        Ret    0  0        возврат из п/п
            a
            b
            c
            d
            m
            n
            u
            v
            w
```

Команда Call («Обращение к подпрограмме») передает управление команде с меткой Met1, т.е. первой команде подпрограммы. Одновременно в

ячейку 16 записывается адрес следующей команды, т.е. адрес команды, которая должна быть активизирована после выхода из подпрограммы. В команде возврата из подпрограммы Ret адресная часть не используется, поскольку эта команда читает содержимое ячейки с заранее заданным адресом 16.

Для контроля правильности работы рассматриваемой программы выполним расчет значения параметра *u* для конкретных величин переменных *a*, *b*, *c*.

$a = 1101\ 0010\ 1011\ 1000_2 = D2B8_{16}$

$b = 1001\ 1010\ 1011\ 1100_2 = 9ABC_{16}$

$a \rightarrow^3 = 0001\ 1010\ 0101\ 0111_2 = 1A57_{16}$

$(a \rightarrow^3) \oplus b = 1000\ 0000\ 1110\ 1111_2 = 80EF_{16}$

$u = (a \rightarrow^3) \oplus b \leftarrow^2 = 0000\ 0011\ 1011\ 1100_2 = 03BC_{16}$

# С О Д Е Р Ж А Н И Е

Стр.