

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

---

**Методические указания и задания**  
**к выполнению курсовой работе по программированию**

часть 1

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

Составители  
Назаренко В.И., Демеш Н.С., Юсупова К.Б.

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 курсовой работе по программированию, часть 1 (для студентов специальности 6.050102 дневной и заочной-форм обучения). Сост. В.И.Назаренко, Н.С.Демеш, К.Б.Юсупова. – 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 А1, А2, В1, В2, что является составной частью заданий к курсовой работе по программированию.

Составители:  
доц. Назаренко В.И.,  
асс. Демеш Н.С.,  
асс. Юсупова К.Б.

Ответственный  
за выпуск  
проф. Святный В.А.

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

# 1. УЧЕБНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА А1

## 1.1 Архитектура ЭВМ А1

Структурная схема ЭВМ А1 приведена на рис.1. В состав ЭВМ А1 входят следующие узлы:

- оперативная память MEMORY емкостью 256 16-разрядных ячеек;
- восьмиразрядный счетчик адреса команд SAK, предназначенный для хранения адреса следующей команды;
- восьмиразрядный регистр адреса ячейки памяти RA;
- шестнадцатиразрядный буферный регистр слова RS для временного хранения читаемой или записываемой информации;
- шестнадцатиразрядный регистр команд RK, хранящий выполняемую команду;
- шестнадцатиразрядный индексный регистр XR для хранения значения индекса при обработке массива;
- сумматор адреса SA, предназначенный для формирования исполнительного адреса операнда;
- шестнадцатиразрядный аккумулятор Acc для хранения промежуточных результатов выполнения программы;
- шестнадцатиразрядные операционные регистры OR1 и OR2 для временного хранения операндов;
- арифметико-логическое устройство АЛУ, выполняющее операцию, заданную кодом операции команды;
- шестнадцатиразрядный регистр результата RR, предназначенный для временного хранения результата операции, выполняемой АЛУ;
- дешифратор кода операции DC;
- четырехразрядный регистр признаков RP.

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

Для команды сравнения первые три бита регистра RP формируются аналогично, четвертый бит, определяющий некорректность выполнения операции, не формируется.

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, содержащейся в регистре RK, входят код выполняемой операции КОП (биты 1 – 4), модификатор адреса МА (биты 5 – 8) и адрес ячейки памяти А (биты 9 – 16).

Адреса ячеек изменяются от 0 до 255 (от 00000000 до 11111111 в двоичной системе счисления).

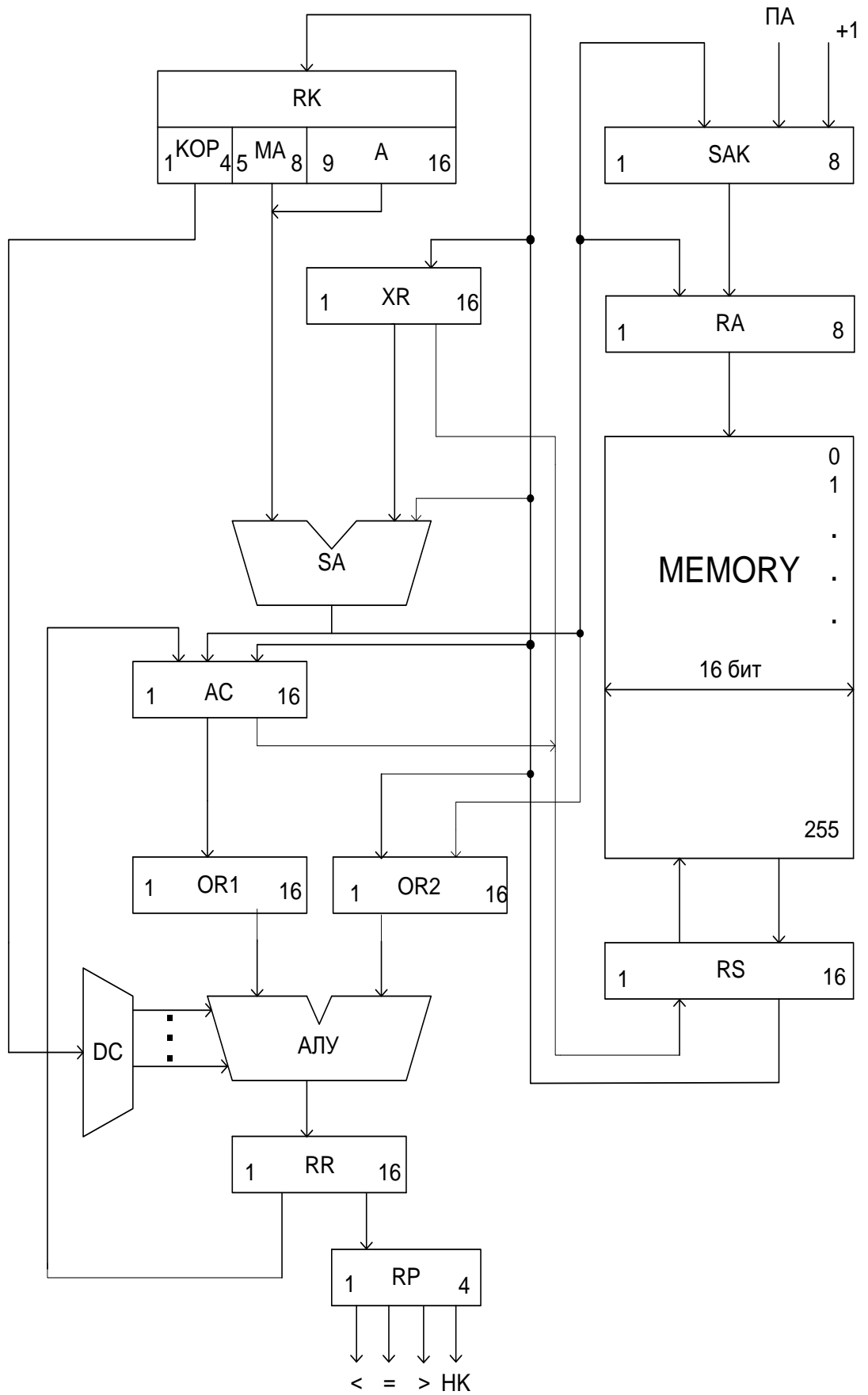


Рисунок 1 – Структурная схема ЭВМ А1

Исполнительный адрес  $A_{исп}$ , по значению которого выбирается из памяти или записывается в память операнд, вычисляется по значению поля А регистра команды в зависимости от значения модификатора адреса МА:

$MA = 0001 \Rightarrow A_{исп} = A$  – прямая адресация;

$MA = 0010 \Rightarrow A_{исп} = A + (XR)$  – индекс-адресация;

$MA = 0100 \Rightarrow A_{исп} = MEMORY(A)$  – косвенная адресация;

$MA = 1000 \Rightarrow (Acc) = A$  или  $(OR2) = A$  – непосредственная адресация.

Система команд ЭВМ А1, приведенная в табл.1, включает в себя:

- команды загрузки и сохранения аккумулятора;
- арифметико-логические команды;
- команды выполнения операций с индексным регистром;
- команды управления порядком выполнения программы.

Таблица 1. Система команд ЭВМ А1

Код опер.	Наименование	Признаки				Выполнение команды
		<	=	>	нк	
0000	Останов	-	-	-	-	
0001	Загрузка аккумулятора	-	-	-	-	$(A_{исп}) \rightarrow Acc$
0010	Сохранение аккумулятора	-	-	-	-	$(Acc) \rightarrow A_{исп}$
0011	Сложение	+	+	+	+	$(Acc) + (A_{исп}) \rightarrow Acc$
0100	Вычитание	+	+	+	+	$(Acc) - (A_{исп}) \rightarrow Acc$
0101	Умножение	+	+	+	+	$(Acc) * (A_{исп}) \rightarrow Acc$
0110	Деление	+	+	+	+	$(Acc) : (A_{исп}) \rightarrow Acc$
0111	Сравнение	+	+	+	-	$(Acc) \geq (A_{исп}) \rightarrow RP$
1000	Конъюнкция	+	+	-	-	$(Acc) \wedge (A_{исп}) \rightarrow Acc$
1001	Дизъюнкция	+	+	-	-	$(Acc) \vee (A_{исп}) \rightarrow Acc$
1010	Сдвиг вправо логический	+	+	-	-	$(Acc) \gg A_{исп} \Rightarrow$
1011	Сдвиг влево логический	+	+	-	-	$(Acc) \ll A_{исп}$
1100	Переход	-	-	-	-	См. описание
1101	Обращение к п/п	-	-	-	-	$SAK \rightarrow 16; A \rightarrow SAK$
1110	Загрузка индекса	-	-	-	-	$(A_{исп}) \rightarrow XR$
1111	Наращивание индекса	-	-	-	-	$(XR) + 1 \rightarrow A_{исп}$

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

Для работы с индексным регистром предусмотрены команды «Загрузка индекса» и «Наращивание индекса». Команда «Загрузка индекса» позволяет формировать содержимое индексного регистра, а команда «Наращивание индекса» - изменять его значение.

В состав команд управления порядком выполнения программы входят: «Останов», «Обращение к подпрограмме» и «Переход».

В команде останова биты 5-16 не используются.

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

## 1.2 Выполнение команд в ЭВМ А1

В ЭВМ А1 используются лишь числа с фиксированной запятой (целые числа со знаком). Отрицательные значения таких чисел хранятся в дополнительном коде.

Выполнение команды начинается с загрузки содержимого ячейки памяти, адрес которой задан в SAK, в регистр команд RK. В этом случае 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. Затем из состава регистра RK выделяются поля KOP, MA 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 дешифруемых команд).

Для арифметико-логической команды следующим этапом ее выполнения является формирование в сумматоре SA исполнительного адреса  $A_{ucn}$  с учетом приведенных выше значений поля MA.

Арифметико-логические операции являются двухместными. При этом первый операнд сохраняется в аккумуляторе и затем передается в операционный регистр OR1, второй операнд записывается в регистр OR2.

Если MA = 0001 (прямая адресация), то исполнительный адрес  $A_{ucn} = A$  передается в регистр RA, по этому адресу производится чтение содержимого ячейки памяти, которое через буферный регистр RS записывается в OR2.

Если MA = 0010 (индекс-адресация), то в сумматоре SA значение поля A складывается с содержимым разрядов 9 – 16 индексного регистра XR, полученное значение передается в RA и, как в предыдущем случае, прочитанная информация через регистр RS записывается в OR2.

Если MA = 0100 (косвенная адресация), то по значению поля A, как и ранее, читается содержимое ячейки памяти, но после прохождения регистра RS это содержимое записывается не в регистр OR2, а снова поступает на

сумматор адреса SA, т.е. воспринимается не как число, а как адрес MEMORY(A). Этот адрес снова поступает в регистр RA и уже на данном этапе прочитанное содержимое ячейки памяти через регистр RS записывается в OR2.

Если  $MA = 1000$  (непосредственная адресация), то значение компонента A непосредственно из сумматора SA передается в операционный регистр OR2.

Арифметико-логическая операция, операнды которой находятся в регистрах OR1 и OR2, выполняется в арифметико-логическом устройстве АЛУ. Результат операции через регистр RR загружается в аккумулятор.

Если в команде «Загрузка аккумулятора» указана непосредственная адресация, то значение поля  $A_{ucn}$  регистра команд из сумматора SA переписывается в аккумулятор. При других видах адресации команды «Загрузка аккумулятора» выполняется передача информации из регистра RS в аккумулятор.

В команде «Сохранение аккумулятора» выполняется передача информации из аккумулятора Acc в регистр RS с последующей записью в заданную ячейку памяти. В этой команде непосредственная адресация не используется.

Команды сдвига также ориентированы на аккумулятор. Освобождаемые при выполнении операции разряды в левой или в правой части аккумулятора заполняются нулями. Количество сдвигов определяется значением адреса  $A_{ucn}$ . Если  $A_{ucn} > 15$ , то сдвиг не производится, но во все разряды аккумулятора записываются нули. Непосредственно сдвиг выполняется в регистре OR1, куда поступает содержимое аккумулятора Acc. В регистр OR2 из сумматора SA передается значение поля  $A_{ucn}$ . Результат операции через регистр RR передается в аккумулятор.

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.1. Признак «нк» (некорректность операции) означает, что в результате выполнения данной операции возникает переполнение регистра результатов RR или же здесь отмечена попытка деления на нуль.

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 перехода).

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

*Примечание.* Если в команде, приведенной в табл.1, все графы призна-

ков отмечены символом “–”, то это означает лишь то, что при этом сохраняются предыдущие значения признаков (но не выполняется сброс их на нуль).

Команда “Сравнение” в основном аналогична команде “Вычитание”, но ее результат не записывается в аккумулятор, а влияет лишь на формирование регистра признаков RP. Фактически в команде сравнения определяется разность первого и второго операндов, но без записи результата. Тогда  $RP[1] = 1$ , если  $(A1_{ucn}) < (A2_{ucn})$ ,  $RP[2] = 1$ , если  $(A1_{ucn}) > (A2_{ucn})$ ,  $RP[3] = 1$ , если  $(A1_{ucn}) = (A2_{ucn})$ . Признак  $RP[4]$  в команде не вырабатывается.

В команде «Загрузка индекса» вначале формируется адрес  $A_{ucn}$ , после чего содержимое ячейки памяти с адресом, заданным в RA, через регистр RS загружается в индексный регистр XR. В команде «Наращивание индекса» к содержимому регистра XR добавляется единица, и результат через буферный регистр RS записывается в ячейку памяти, определяемую содержимым регистра адреса RA. В обоих этих командах непосредственная адресация не применяется.

В команде перехода поле MA считается маской, а не модификатором адреса. 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, т.е. при этом адрес A из сумматора SA пересылается в счетчик SAK и, следовательно, воспринимается как адрес следующей команды. Частные случаи:

- MA = 0000 - пустая команда (нет передачи управления);
- MA = 1000 - условный переход по знаку «-»;
- MA = 0100 - условный переход по нулю;
- MA = 0010 - условный переход по знаку «+»;
- MA = 1010 - условный переход по знаку «-» или по нулю;
- .....
- MA = 1111 - безусловный переход.

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

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

Значение  $MA = 0001$  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.

По команде «Обращение к подпрограмме» значение счетчика SAK (адрес следующей команды) запоминается в ячейке памяти с адресом 16 (в битах 9 – 16), а значение поля A данной команды через сумматор SA переписывается



вается в счетчик адреса команды SAK. В данной команде  $A_{исп} = A$ .

### 1.3 Программирование в кодах ЭВМ А1

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

Машинная команда ЭВМ А1 состоит из трех частей: код операции, модификатор адреса и адрес операнда.

Для кода операции будем использовать следующие обозначения:

Halt – останов;  
Load – загрузка аккумулятора;  
Save – сохранение аккумулятора;  
Add – сложение;  
Sub – вычитание;  
Mul – умножение;  
Div – деление;  
Comp – сравнение;  
And – конъюнкция;  
Or – дизъюнкция;  
Shr – сдвиг вправо логический;  
Shl – сдвиг влево логический;  
Jump – переход;  
Call – обращение к подпрограмме;  
Lx – загрузка индекса;  
Inc – наращивание индекса.

Аналогично введем условные обозначения для модификатора адреса:

P – прямая адресация;  
I – индекс-адресация;  
K – косвенная адресация;  
N – непосредственная адресация.

В адресной части команды может быть:

- непосредственный операнд;
- 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) Load N 25

Здесь в адресной части указан непосредственный операнд - константа

25.

б) Save P R

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

в) Jump F Met1

Безусловный переход по адресу Met1 (на метку Met1).

г) Add P <x>

В аккумулятор добавляется содержимое ячейки, в которой записано значение переменной x.

### 1.3.1 Программирование арифметического выражения

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

При обработке числовых констант, не превышающих значения 255, будем применять непосредственную адресацию, в остальных случаях – прямую.

14	Load	P	<a>	$a \rightarrow Acc$
15	Mul	N	2	$(Acc) * 2 \rightarrow Acc$
16	Sub	N	1	$(Acc) - 1 \rightarrow Acc$
17	Save	P	R	$2a - 1 \rightarrow R$
18	Load	P	<a>	$a \rightarrow Acc$
19	Mul	N	25	$25a \rightarrow Acc$
1A	Add	P	<338>	$25a + 338 \rightarrow Acc$
1B	Div	P	R	$(Acc)/(R) \rightarrow Acc$
1C	Save	P	<y>	$(Acc) \rightarrow \langle y \rangle$
1D	Halt			
1E			a	
1F			338	
20			y	
21			R	

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	0001	0001	1110
15	0101	1000	0000	0001
16	0100	1000	0000	0001
17	0010	0001	0010	0001
18	0001	0001	0001	1110

19	0101	1000	0001	1001
1A	0011	0001	0001	1111
1B	0110	0001	0010	0001
1C	0010	0001	0010	0000
1D	0000	0000	0000	0000
1E	0000	0000	0000	1010
1F	0000	0001	0101	0010

Здесь адреса 20 и 21 не отображаются, поскольку значение переменной  $y$  и содержимое буферной ячейки  $R$  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
010436
054002
044001
020441
010436
054031
030437
060441
020440
000000
000012
000522

```

Значение переменной  $y$  как результат вычислений должно быть записано в ячейку с адресом  $20_{16} = 32_{10}$ . Этот результат равен  $33_{10} = 21_{16} = 41_8$

В п. 1.3.1 проверены машинные команды Load, Save, Add, Sub, Mul, Div, Halt, а также два вида адресации – прямая и непосредственная.

### 1.3.2 Программирование разветвляющихся процессов

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

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.

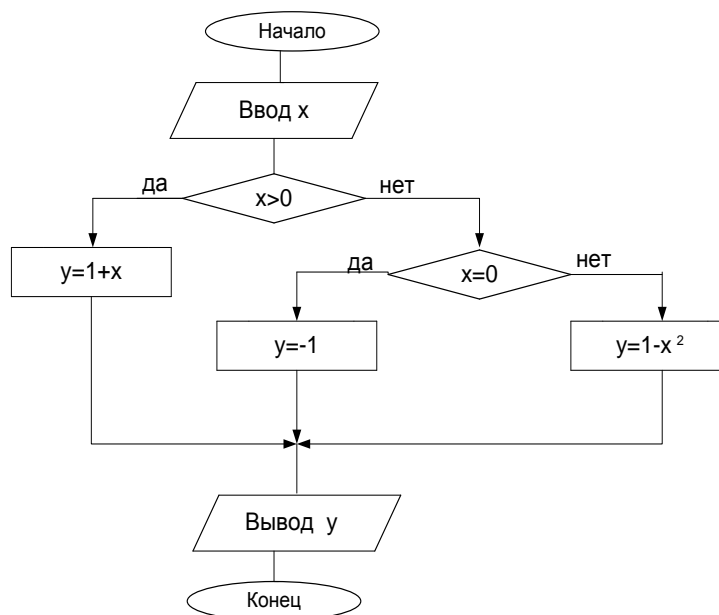


Рисунок 2 – Двухмерная блок-схема задачи с разветвлениями

	Load	P	<x>	$x \rightarrow Acc$
	Jump	A	Met2	Переход на ветви 2 и 3 ( $x \leq 0$ )
	Add	N	1	$x + 1 \rightarrow Acc$
	Jump	F	Met4	Переход на запись ( $Acc$ ) $\rightarrow$ <y>
Met2	Jump	8	Met3	Переход на ветвь 3
	Load	P	-1	$-1 \rightarrow Acc$
	Jump	F	Met4	Переход на запись ( $Acc$ ) $\rightarrow$ <y>
Met3	Mul	P	<x>	$x^2 \rightarrow Acc$
	Save	P	R1	$x^2 \rightarrow R1$
	Load	N	1	$1 \rightarrow Acc$
	Sub	P	R1	$1 - x^2 \rightarrow Acc$

Met4	Save	P	y	(Acc) → < y >
	Halt			
			-1	
			x	
			y	
			R1	

Маски в командах перехода представлены в шестнадцатеричном виде.

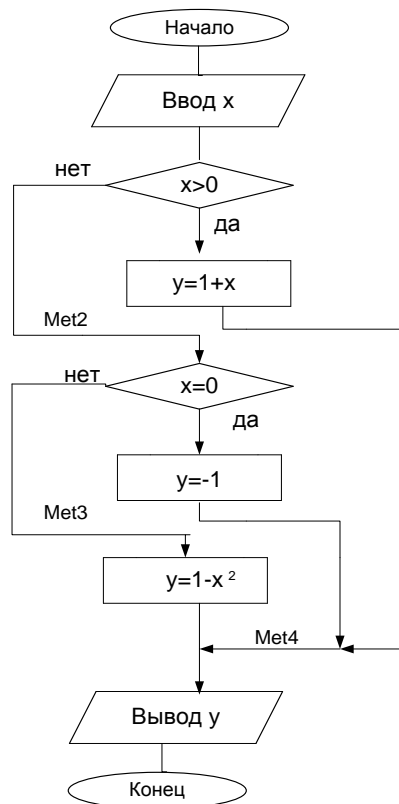


Рисунок 3 – Линейная блок-схема задачи с разветвлениями

Второй этап – запись адресов ячеек памяти.

55	Load	P	<x>	$x \rightarrow Acc$
56	Jump	A	Met2	Переход на ветви 2 и 3 ( $x \leq 0$ )
57	Add	N	1	$x + 1 \rightarrow Acc$
58	Jump	F	Met4	Переход на запись (Acc) → < y >
59	Met2	Jump	8 Met3	Переход на ветвь 3
5A	Load	P	-1	$-1 \rightarrow Acc$
5B	Jump	F	Met4	Переход на запись (Acc) → < y >
5C	Met3	Mul	P <x>	$x^2 \rightarrow Acc$
5D	Save	P	R1	$x^2 \rightarrow R1$
5E	Load	N	1	$1 \rightarrow Acc$
5F	Sub	P	R1	$1 - x^2 \rightarrow Acc$

60	Met4 Save	P	<y>	(Acc) → <y>
61	Halt			
62		-1		
63		x		
64		y		
65		R1		

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

В п.1.3.2 дополнительно проверена команда Jump.

### 1.3.3 Организация циклической программы

Наличие индексного регистра и команд работы с этим регистром («Загрузка индекса» и «Наращивание индекса») позволяет легко реализовать циклическую программу. В качестве примера рассмотрим программную реализацию вычисления 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

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

	14	Load	N	0	$0 \rightarrow Acc$
	15	Save	P	<S>	$0 \rightarrow \langle S \rangle$
	16	Save	P	< $k_c$ >	$0 \rightarrow \langle k_c \rangle$
	17	Lx	N	0	$0 \rightarrow XR$
Met	18	Load	P	<S>	$S \rightarrow Acc$
	19	Add	I	< $x_1^*$ >	$(Acc) + x_1^* \rightarrow Acc$
	1A	Save	P	<S>	$(Acc) \rightarrow \langle S \rangle$
	1B	Inc	P	< $k_c$ >	$(XR) + 1 \rightarrow \langle k_c \rangle$
	1C	Load	P	< $k_c$ >	$\langle k_c \rangle \rightarrow Acc$
	1D	Comp	P	<5>	$k_c = 5?$
	1E	Jump	A	Met	$k_c \neq 5 \Rightarrow Met$
	1F	Load	P	<8>	$8 \rightarrow Acc$
	20	Mul	P	<a>	$8a \rightarrow Acc$
	21	Sub	P	<S>	$8a - S \rightarrow Acc$
	22	Save	P	<y>	$y \rightarrow \langle y \rangle$

23	Halt		<i>останов</i>
24		0	<i>счетчик циклов <math>k_c</math></i>
25		8	<i>константа 8</i>
26		5	<i>количество циклов</i>
27	a =	31	
28	x <sub>1</sub> =	10	
29	x <sub>2</sub> =	-15	
2A	x <sub>3</sub> =	25	
2B	x <sub>4</sub> =	40	
2C	x <sub>5</sub> =	50	
2D	Y		
2E	S		

В начале программы обнуляются счетчик циклов  $k_c$  и ячейка для суммы  $S$ , а также заносится нуль в регистр  $XR$ . 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 счетчик циклов и содержимое индексного регистра, имеющие начальное нулевое значение, в конце каждого цикла должны увеличиваться на единицу.

Поскольку в команде  $\text{Add I } \langle x_1^* \rangle$  указана индекс-адресация, то в первом цикле к адресу  $\langle x_1 \rangle$  добавляется 0, во втором – 1, в третьем – 2 и т.д. Следовательно, при выполнении этой команды к нулевой исходной сумме будут добавляться значения  $x_1, x_2, x_3, \dots$ . Перед проверкой условия завершения цикла счетчик циклов  $k_c$  и содержимое регистра  $XR$  командой наращивания индекса  $\text{Inc}$  увеличиваются на единицу, затем счетчик циклов загружается в аккумулятор и сравнивается с числом 5, определяющем заданное количество циклов. Если текущее значение счетчика меньше, чем 5, то команда перехода передает управление на повторение цикла, в противном случае – следующей команде.

В п. 1.3.3 дополнительно проверяются команды  $\text{LX}$ ,  $\text{Inc}$ ,  $\text{Comp}$  и индекс-адресация.

Запишем теперь рассматриваемую программу в машинном виде и сформируем в 16 с/с текстовый файл программы. При этом пусковой адрес программы, равный  $20_{10} = 14_{16}$ , записывается в первой строке файла в 10 с/с, в остальных строках – машинные команды и числа в 16 с/с.

	20
0001100000000000	1800
0010000100101110	212E
0010000100100100	2124
1110100000000000	E800
0001000100101110	112E
0011001000101000	3228
0010000100101110	212E
1111000100100100	F124
0001000100100100	1124
0111000100100110	7126
1100101000011000	CA18
0001000100100101	1125
0101000100100111	5127
0100000100101110	412E
0010000100101101	212D
0000000000000000	0000
0000000000000000	0000
00000000000001000	0008
00000000000000101	0005
00000000000011111	001F
00000000000001010	000A
11111111111110001	FFF1
00000000000011001	0019
00000000000101000	0028
00000000000110010	0032

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

### 1.3.4 Программирование подпрограмм

ЭВМ А1 не содержит команды возврата из подпрограммы, здесь производимая командой «Обращение к подпрограмме» запись значения SAK в ячейку с адресом 16 предоставляет лишь информацию о том, куда требуется передать управление после окончания работы подпрограммы. В связи с этим рекомендуется использовать следующий прием.

Последней командой подпрограммы нужно записать команду безусловной передачи управления с нулевой адресной частью, а в финишной части подпрограммы вставить в эту команду адрес, содержащийся в ячейке 16. Это может иметь следующий вид:



k+0	<i>начало подпрограммы</i>			
.....				
	Load	P	k+n	$(k+n) \rightarrow Acc$
	And	P	$\langle k_6 \rangle$	$(Acc) \wedge \langle k_6 \rangle \rightarrow Acc$
	Or	P	16	$(Acc) \vee (16) \rightarrow Acc$
	Save	P	k+n	$(Acc) \rightarrow k+n$
k+n	Jump	F	0	<i>возврат из n/n</i>
k+n+1	<i>основная программа</i>			
.....				
k+m	Call	0	k+0	<i>обращение к n/n</i>
k+m+1	.....			
.....				
	Halt			<i>останов</i>
	1111	1111	0000	0000
				<i>константа выделения <math>k_6</math></i>

Предположим, что команда обращения к подпрограмме Call имеет адрес k+m. Тогда при ее выполнении в ячейку 16 будет записан адрес следующей команды 000 0000 k+m+1 (в разряды 1-8 – нули, в разряды 9-16 – адрес k+m+1). После этого управление будет передано по адресу k+0, т.е. на начало подпрограммы.

Последней командой подпрограммы является безусловный переход с нулевой адресной частью. Перед этим команда Load производит загрузку содержимого ячейки k+n, т.е. команду Jump, в аккумулятор, дальше при выполнении команды логического умножения And обнуляется адресная часть содержимого аккумулятора (она может быть ненулевой после предыдущего обращения к подпрограмме) и по команде логического сложения к ней добавляется содержимое ячейки 16, формируя тем самым адресную часть для команды возвращения из подпрограммы. Команда Save записывает сформированную команду по адресу k+n, а при выполнении команды Jump производится возврат в основную программу по адресу k+m+1.

После реализации пп. 1.3.1, 1.3.2 и 1.3.3 остались непроверенными команды And, Or, Shr, Shl и Call, а также косвенный тип адресации.

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

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

	Load	P	a	
	Save	P	l	$l := a$
	Load	P	b	
	Save	P	m	$m := b$
	Load	P	c	
	Save	P	n	$n := c$
	Call	Met1		Обращение к подпрограмме
	Load	P	w	
	Save	P	u	$u := w$
	Load	P	d	
	Save	P	l	$l := d$
	Load	P	e	
	Save	P	m	$m := e$
	Load	P	f	
	Save	P	n	$n := f$
	Call	Met1		Обращение к подпрограмме
	Load	P	w	
	Save	P	v	$v := w$
	Halt			
Met1	Load	P	l	$l \rightarrow Acc$
	Shl	N	3	$(Acc) \leftarrow^3$
	And	P	m	$(Acc) \wedge \langle m \rangle \rightarrow Acc$
	Shr	N	2	$(Acc) \rightarrow^2 \rightarrow Acc$
	Or	P	n	$(Acc) \vee \langle n \rangle \rightarrow Acc$
	Save	P	w	$(Acc) \rightarrow \langle w \rangle$
	Load	P	Met2	Формирование команды
	And	P	k	возврата из
	Or	P	16	подпрограммы
	Save	P	Met2	
Met2	Jump	F	0	

a  
b  
c  
d  
e  
f  
k=FFFF0000  
l  
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.

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

$$a = 0001\ 0010\ 0011\ 0100\ 0101\ 0110_2 = 123456_{16}$$

$$b = 0111\ 1000\ 1001\ 1010\ 1011\ 1100_2 = 789ABC_{16}$$

$$c = 1101\ 1110\ 1111\ 1110\ 1101\ 1100_2 = DEFEDC_{16}$$

$$a \leftarrow^3 = 1001\ 0001\ 1010\ 0010\ 1011\ 0000_2 = 91A2B0_{16}$$

$$(a \leftarrow^3) \wedge b = 0001\ 0000\ 1000\ 0010\ 1011\ 0000_2 = 1082B0_{16}$$

$$(a \leftarrow^3) \wedge b \rightarrow^2 = 0000\ 0100\ 0010\ 0000\ 1010\ 1100_2 = 0420AC_{16}$$

$$u = (a \leftarrow^3) \wedge b \rightarrow^2 \vee c = 1101\ 1110\ 1111\ 1110\ 1111\ 1100_2 = DEF EFC_{16}$$

## 2. УЧЕБНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА А2

### 2.1 Архитектура ЭВМ А2

Учебная ЭВМ А2 является двухадресной машиной, ее структурная схема приведена на рис.4. В состав ЭВМ А2 входят следующие узлы:

- оперативная память MEMORY емкостью 256 24-разрядных ячеек;
- восьмиразрядный счетчик адреса команд SAK, предназначенный для хранения адреса следующей команды;
- восьмиразрядный регистр адреса RA, определяющий номер ячейки памяти, с которой происходит взаимодействие в данный момент времени;
- 24-разрядный буферный регистр слова RS для временного хранения читаемой или записываемой информации;
- 24-разрядный регистр команд RK, хранящий выполняемую команду;
- схема формирования адреса СФА, предназначенная для формирования исполнительного адреса операнда;
- 24-разрядные операционные регистры OR1 и OR2 для временного хранения операндов;
- арифметико-логическое устройство АЛУ, выполняющее операцию, заданную кодом операции команды;
- 24-разрядный регистр результата RR, предназначенный для временного хранения результата операции, выполняемой АЛУ;
- дешифратор кода операции DC;
- четырехразрядный регистр признаков завершения операции RP.

При выполнении арифметической операции первый бит регистра RP устанавливается в «1», если ее результат отрицательный; второй бит устанавливается в «1» при положительном результате этой операции; третий бит - при нулевом результате; четвертый бит устанавливается в «1» в случае некорректности выполнения операции (переполнение регистра результатов RR 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;  $RP[3] = 1$ , если этот результат равен нулю.

В состав команды входят код выполняемой операции КОП (биты 1 – 4), модификатор первого адреса МА1 (биты 5 – 6), модификатор второго адреса МА2 (биты 7 – 8), адрес первого операнда А1 (биты 9 – 16) и адрес второго операнда А2 (биты 17 – 24).

Адреса ячеек изменяются от 0 до 255 (от 00000000 до 11111111 в двоичной системе счисления).

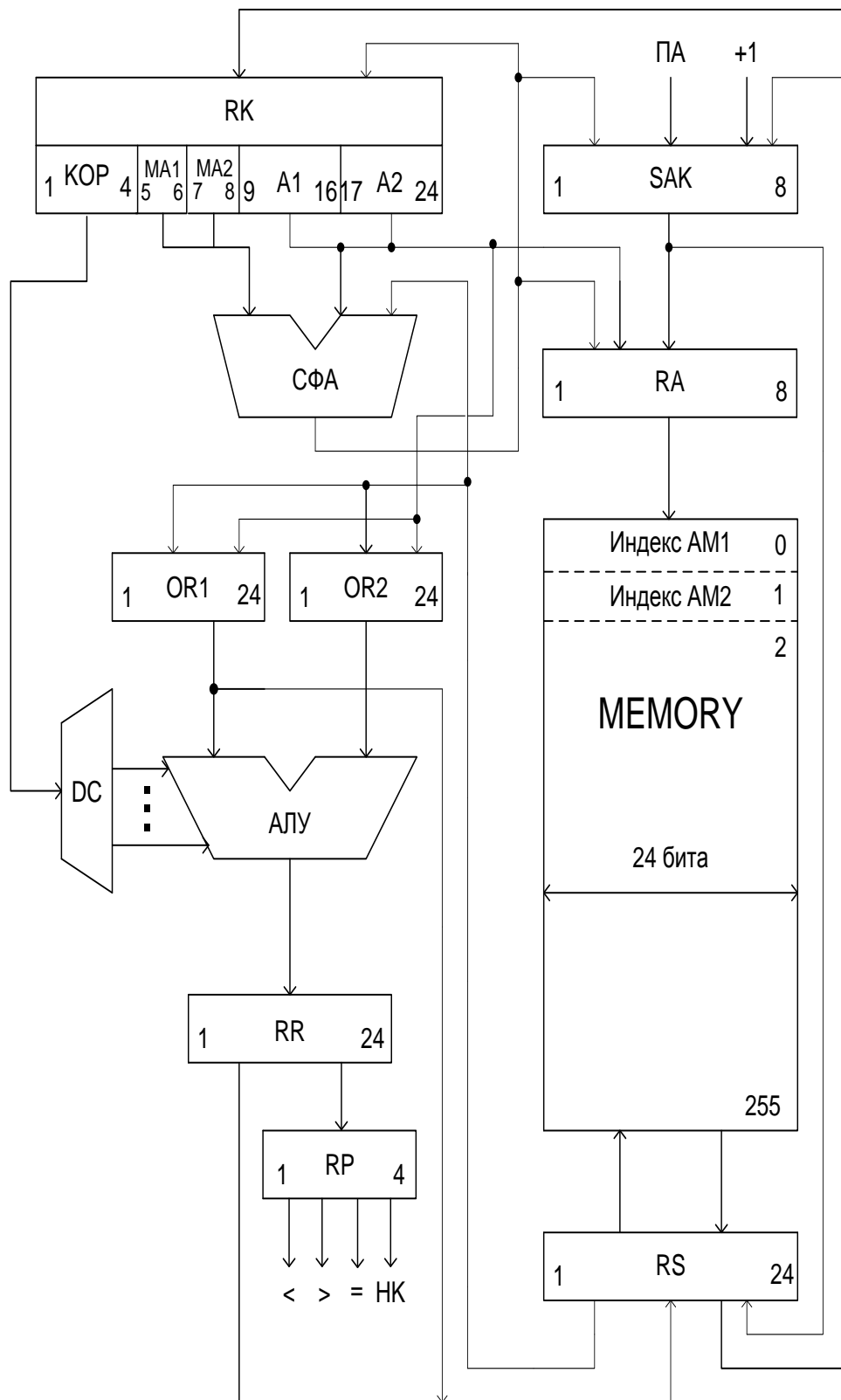


Рисунок 4 – Структурная схема ЭВМ А2

Исполнительные адреса  $A1_{ucn}$  и  $A2_{ucn}$ , по значению которых определяются ячейки памяти для первого и второго операндов, вычисляются по значению полей A1 и A2 регистра команды в зависимости от значения модификаторов MA1 и MA2 следующим образом:

$MA_i = 00 \Rightarrow A_{iucn} = A_i + MEMORY(P_i)$  – индекс-адресация;

$MA_i = 01 \Rightarrow OR_i = A_i$  – непосредственная адресация;

$MA_i = 10 \Rightarrow A_{iucn} = MEMORY(A_i)$  – косвенная адресация;

$MA_i = 11 \Rightarrow A_{iucn} = A_i$  – прямая адресация.

Здесь  $i = 1$  или  $2$ . Следовательно, рассматриваются модификаторы адреса MA1 и MA2, адреса A1 и A2, исполнительные адреса  $A1_{ucn}$  и  $A2_{ucn}$ .  $P_i$  – это адрес индексной ячейки, а именно  $P_1 = 0$ ,  $P_2 = 1$ , т.е. при индекс-адресации в качестве индексного регистра для первого операнда используется ячейка памяти с адресом 0, для второго операнда – с адресом 1.

Система команд ЭВМ А2, приведенная в табл.2, включает в себя:

- команды пересылки данных из одной ячейки памяти в другую;
- арифметико-логические команды;
- команды управления порядком выполнения программы.

В первую группу входят команды пересылки первого операнда во второй и наоборот.

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

В состав команд управления порядком выполнения программы входят: «Останов», «Вызов подпрограммы», «Возврат из подпрограммы» и «Переход».

В арифметико-логических командах операция выполняется над операндами, выбранными из памяти по адресам  $A1_{ucn}$  и  $A2_{ucn}$ , результат записывается в ячейку памяти по адресу  $A1_{ucn}$ .

Команда сравнения является исключением в группе арифметико-логических команд. Ее результат никуда не записывается, лишь формируется регистр признаков RP. Фактически в команде сравнения определяется разность первого и второго операндов, но без записи результата. Тогда  $RP[1] = 1$ , если  $(A1_{ucn}) < (A2_{ucn})$ ,  $RP[2] = 1$ , если  $(A1_{ucn}) > (A2_{ucn})$ ,  $RP[3] = 1$ , если  $(A1_{ucn}) = (A2_{ucn})$ . Признак  $RP[4]$  в команде не вырабатывается.

В команде останова биты 5 – 24 не используются.

Таблица 2. Система команд ЭВМ А2

Код опер.	Наименование	Признаки				Выполнение команды
		<	>	=	нк	
0000	Останов	-	-	-	-	
0001	Пересылка влево	-	-	-	-	$(A2_{ucn}) \rightarrow A1_{ucn}$
0010	Пересылка вправо	-	-	-	-	$(A1_{ucn}) \rightarrow A2_{ucn}$
0011	Сложение	+	+	+	+	$(A1_{ucn}) + (A2_{ucn}) \rightarrow A1_{ucn}$
0100	Вычитание	+	+	+	+	$(A1_{ucn}) - (A2_{ucn}) \rightarrow A1_{ucn}$
0101	Умножение	+	+	+	+	$(A1_{ucn}) * (A2_{ucn}) \rightarrow A1_{ucn}$
0110	Деление	+	+	+	+	$(A1_{ucn}) : (A2_{ucn}) \rightarrow A1_{ucn}$
0111	Сравнение	+	+	+	-	$(A1_{ucn}) \Leftrightarrow (A2_{ucn}) \rightarrow RP$
1000	Конъюнкция	+	-	+	-	$(A1_{ucn}) \wedge (A2_{ucn}) \rightarrow A1_{ucn}$
1001	Дизъюнкция	+	-	+	-	$(A1_{ucn}) \vee (A2_{ucn}) \rightarrow A1_{ucn}$
1010	Сумма по модулю 2	+	-	+	-	$(A1_{ucn}) \oplus (A2_{ucn}) \rightarrow A1_{ucn}$
1011	Сдвиг вправо логич.	+	-	+	-	$(A1_{ucn}) \ A2 \Rightarrow$
1100	Сдвиг влево логич.	+	-	+	-	$(A1_{ucn}) \ A2 \Leftarrow$
1101	Переход	-	-	-	-	См. описание
1110	Вызов подпрограммы	-	-	-	-	$(SAK) \rightarrow A1_{ucn}; A2_{ucn} \rightarrow SAK$
1111	Возврат из п/п	-	-	-	-	$(A1_{ucn}) \rightarrow SAK$

## 2.2 Выполнение команд в ЭВМ А2

В ЭВМ А2 используются 24-разрядные числа с фиксированной запятой (целые числа со знаком). Отрицательные значения таких чисел хранятся в дополнительном коде.

Максимальное значение 24-разрядного двоичного числа составляет  $2^{23} - 1 = 1024 \cdot 1024 \cdot 8 - 1 = 8\,388\,607$ . Следовательно, при моделировании ЭВМ А2 средствами Турбо Паскаля таким числам не может соответствовать тип `integer` (здесь максимальное значение равно  $2^{15} - 1 = 32767$ ), а лишь тип `longint` (максимальное значение равно  $2^{31} - 1 = 2\,147\,483\,647$ ), но при выполнении арифметических операций необходимо контролировать выход за пределы допустимого диапазона  $-(2^{23} - 1) \dots (2^{23} - 1)$ .

Выполнение команды начинается с загрузки содержимого ячейки памяти, адрес которой задан в SAK, в регистр команд RK. В этом случае 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. Затем из RK выделяются поля KOP, MA1, MA2, 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 дешифруемых команд).

Далее во всех командах, кроме останова и перехода, производится формирование исполнительных адресов на основании текущих значений полей MA1, MA2, A1 и A2. При индекс-адресации и косвенной адресации формирование адреса  $A_{исп}$  выполняет схема формирования адреса СФА.

Круглые скобки в графе «Выполнение команды» означают содержимое регистра или ячейки памяти. Например, запись  $A2_{исп}$  – это адрес ячейки, а  $(A2_{исп})$  – содержимое ячейки с адресом  $A2_{исп}$ .

Если  $MA_i = 00$  (индекс-адресация), то устройство СФА производит чтение из регистра команд RK адреса  $A_i$ , а из памяти MEMORY – содержимого ячейки  $P_i$ , выполняет сложение их значений, а полученный результат снова записывает в регистр RK на место адреса  $A_i$ .

Если  $MA_i = 01$  (непосредственная адресация), то значение адреса  $A_i$  в качестве операнда непосредственно передается на операционный регистр  $OR_i$ .

Если  $MA_i = 10$  (косвенная адресация), то адрес  $A_i$  через СФА поступает на регистр RA, по этому адресу читается содержимое соответствующей ячейки памяти, которое, как и при индекс-адресации, через СФА записывается в регистр RK на место адреса  $A_i$ .

Если  $MA_i = 11$  (прямая адресация), то адрес  $A_i$  передается на регистр RA, по этому адресу читается содержимое ячейки памяти и через регистр RS пересылается на операционный регистр  $OR_i$ .

При прямой и непосредственной адресации содержимое регистра RK не изменяется.

Команды пересылки не производят непосредственно пересылку данных из одной ячейки памяти в другую. При пересылке влево из RK читается сформированный адрес  $A2_{исп}$ . Этот адрес передается на регистр RA, а прочитанное содержимое ячейки памяти через регистр RS передается на регистр OR1. Если  $MA2 = 01$  (непосредственная адресация), то значение  $A2_{исп}$  непосредственно передается в регистр OR1. После этого из RK читается адрес  $A1_{исп}$  и содержимое регистра OR1 через RS записывается в память по адресу,



переданному в регистр RA. В команде пересылки влево не может быть задано  $MA1 = 01$ , т.е. для первого операнда запрещено указание непосредственной адресации.

Команда пересылки вправо работает аналогично. Здесь непосредственная адресация запрещена для второго операнда.

В арифметико-логических командах для второго операнда допустимы все виды адресации, для первого операнда считается недопустимой непосредственная адресация. Для первого операнда сформированный адрес A1 передается в регистр RA, выполняется чтение ячейки памяти, а полученная информация через регистр RS записывается в OR1. Для второго операнда прочитанное содержимое ячейки памяти пересылается в регистр OR2. Если  $MA2 = 01$ , то в OR2 пересылается адрес A2, который в данном случае считается операндом. Результат операции, выполненной в АЛУ по отношению к содержимому регистров OR1 и OR2, направляется в регистр результатов RR, после чего записывается в память по адресу A1, повторно прочитанному из регистра команд RK.

*Примечание.* Выше было указано, что исполнительные адреса при косвенной и индекс-адресации после их формирования записываются на прежнее место в регистр RK. Следовательно, при выполнении арифметико-логической команды чтение адресов для передачи соответствующей информации в регистры OR1 и OR2 производится из регистра RK.

В командах сдвига для второго операнда всегда должна указываться непосредственная адресация, для первого операнда такая адресация неприменима. При выполнении команды первый операнд, как и ранее, читается из памяти и пересылается в регистр OR1, второй операнд A2 пересылается непосредственно из регистра команд RK в операционный регистр OR2. Освобождаемые при выполнении операции разряды в левой или в правой части OR1 заполняются нулями. Количество сдвигов определяется значением адреса A2. Если  $A2 > 23$ , то сдвиг не производится, но во все разряды регистра OR1 записываются нули. Результат операции через регистры RR и RS записывается в память по адресу A1.

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 при выполнении команды перехода. Требуемые изменения регистра RP отражены в табл.2. Признак «нк» (некорректность операции) означает, что в результате выполнения данной операции возникает переполнение регистра результатов RR или же здесь отмечена попытка деления на нуль.

Символ “-” в графе признаков табл.2 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 положительный результат. Тогда в состояние «1» должен быть установлен лишь второй бит регистра RP, остальные биты должны быть сброшены на нуль.

*Примечание.* Если в команде, приведенной в табл.2, все графы признаков отмечены символом “–”, то это означает лишь то, что при этом сохраняются предыдущие значения признаков (но не выполняется сброс их на нуль).

Команда “Сравнение” в основном аналогична команде “Вычитание”, но ее результат не записывается в память, а влияет лишь на формирование регистра признаков RP.

В команде «Переход» биты 5 – 8 рассматриваются как маска MS команды, а не модификаторы адреса MA1 и MA2. При выполнении этой команды содержимое поля MS логически умножается на содержимое регистра признаков RP. Если  $(MS) \wedge (RP) \neq 0$ , то производится переход по адресу A2, т.е. значение адреса A2 посылается в счетчик SAK и, следовательно, воспринимается как адрес следующей команды. Если  $(MS) \wedge (RP) = 0$ , то значение счетчика SAK не изменяется и выполняется команда, адрес которой в данный момент содержится в этом счетчике, т.е. команда, записанная в программе после команды перехода.

Поле A1 в команде перехода не используется. В этой команде применяется лишь прямая адресация, поскольку компонент MS – это маска, а не модификатор адреса.

Частные случаи:

- MS = 0000 - пустая команда (нет передачи управления);
- MS = 1000 - условный переход по знаку «-»;
- MS = 0100 - условный переход по знаку «+»;
- MS = 0010 - условный переход по нулю;
- MS = 1010 - условный переход по знаку «-» или по нулю;
- .....
- MS = 1111 - безусловный переход.

Некоторой особенностью обладает рассматриваемая команда при значении маски MS = 1111. В этом случае логическое умножение маски на содержимое регистра признаков RP не производится и, следовательно, безусловный переход по адресу A2 выполняется при любом состоянии этого регистра.

Значение MS = 0001 определяет переход по признаку некорректности. В этом случае необходимо по адресу A2 перейти на вывод сообщения об аварийном прерывании, указав при этом код операции и адрес команды, вы-

завшей прерывание, после чего произвести останов ЭВМ.

В команде «Вызов подпрограммы» значение поля  $A_{1_{исп}}$  передается в регистр RA, а текущее значение счетчика SAK – в регистр RS. После этого содержимое RS записывается в память по адресу, который определен значением RA. Этим самым осуществляется запоминание адреса возврата в вызывающую программу (на команду, записанную после команды обращения к подпрограмме). Вслед за тем адрес  $A_{2_{исп}}$ , соответствующий первой команде подпрограммы, записывается в счетчик SAK; это означает, что произведен переход к выполнению заданной подпрограммы.

В команде «Возврат из подпрограммы», которая записывается последней в подпрограмме, поле  $A_{1_{исп}}$  передается в регистр RA, по этому адресу выбирается в RS соответствующая ячейка памяти. Содержимое RS переписывается в счетчик SAK, что и определяет возврат в вызывающую программу.

### 2.3 Программирование в кодах ЭВМ А2

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

Машинная команда ЭВМ А2 состоит из следующих частей:

- код операции;
- модификатор первого адреса MA1;
- модификатор второго адреса MA2;
- адрес A1;
- адрес A2.

Для кода операции будем использовать следующие обозначения:

- Halt – останов;
- MovL – пересылка влево;
- MovR – пересылка вправо;
- Add – сложение;
- Sub – вычитание;
- Mul – умножение;
- Div – деление;
- Comp – сравнение;
- And – конъюнкция;
- Or – дизъюнкция;
- Xor – сумма по модулю 2 (исключающее ИЛИ);
- Shr – сдвиг вправо логический;
- Shl – сдвиг влево логический;
- Jump – переход;

Comp – сравнение;  
 Call – обращение к подпрограмме;  
 Ret – возврат из подпрограммы.

Для модификаторов адреса, как и для кода операции, будем использовать условные обозначения:

P – прямая адресация;  
 I – индекс-адресация;  
 K – косвенная адресация;  
 N – непосредственная адресация.

В адресной части команды может быть:

- непосредственный операнд;
- 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) MovL PN R 25

Здесь в адресной части A2 указан непосредственный операнд - константа 25, в адресной части A1 – адрес буферной ячейки R.

б) Jump F 0 Met1

Безусловный переход по адресу Met1 (на метку Met1).

в) Add PP R <x>

К содержимому рабочей ячейки R добавляется содержимое ячейки, в которой записано значение переменной x, полученная сумма пересылается в ту же ячейку R.

### 2.3.1 Программирование арифметического выражения

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

При обработке числовых констант, не превышающих значения 255, будем применять непосредственную адресацию, в остальных случаях – прямую.

14	MovL	PP	R1	<a>	$a \rightarrow R1$
15	Mul	PN	R1	2	$2a \rightarrow R1$
16	Sub	PN	R1	1	$2a - 1 \rightarrow R1$
17	MovL	PP	R2	<a>	$a \rightarrow R2$
18	Mul	PN	R2	25	$25a \rightarrow R2$
19	Add	PP	R2	<338>	$25a + 338 \rightarrow R2$
1A	Div	PP	R2	R1	$(R2)/(R1) \rightarrow R2$
1B	MovR	PP	R2	<y>	$R2 \rightarrow \langle y \rangle$

1C	Halt
1D	a = 10
1E	338
1F	y
20	R1
21	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  $14_{16} = 20_{10}$ . Запись (R1) означает содержимое ячейки с номером R1.

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

14	0001	1111	0010	0000	0001	1101
15	0101	1101	0010	0000	0000	0010
16	0100	1101	0010	0000	0000	0001
17	0001	1111	0010	0001	0001	1101
18	0101	1101	0010	0001	0001	1001
19	0011	1111	0010	0001	0001	1110
1A	0110	1111	0010	0001	0010	0000
1B	0010	1111	0010	0001	0001	1111
1C	0000	0000	0000	0000	0000	0000
1D	0000	0000	0000	0000	0000	1010
1E	0000	0000	0000	0001	0101	0010

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
07620035
27220002
23220001
07620435
27220431
17620436
33620440
13620437
00000000

00000012  
00000522

Значение переменной  $y$  как результат вычислений должен быть записан в ячейку с адресом  $1F_{16} = 31_{10}$ . Этот результат равен  $33_{10} = 21_{16} = 41_8$ .

В п. 2.3.1 проверены машинные команды MovL, MovR, Add, Sub, Mul, Div, Halt, а также два вида адресации – прямая и непосредственная.

### 2.3.2 Программирование разветвляющихся процессов

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

Блок-схемное решение рассматриваемой задачи приведено на рис. 2 и 3 в описании ЭВМ А1.

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.

	MovL	PN	R2	0	$0 \rightarrow R2$
	Add	PP	R2	<x>	$x+0 \rightarrow R2$
	Jump	A	0	Met2	Переход на ветви 2 и 3 ( $x \leq 0$ )
	Add	PN	R2	1	$x+1 \rightarrow R2$
	Jump	F	0	Met4	Переход на запись ( $R2$ ) $\rightarrow y$
Met2	Jump	8	0	Met3	Переход на ветвь 3
	MovL	PP	R2	<-1>	$-1 \rightarrow R2$
	Jump	F	0	Met4	Переход на запись ( $R2$ ) $\rightarrow y$
Met3	Mul	PP	R2	<x>	$x^2 \rightarrow R2$
	MovR	PP	R2	R3	( $R2$ ) $\rightarrow R3$
	MovL	PN	R2	1	$1 \rightarrow R2$
	Sub	PP	R2	R3	$1-x^2 \rightarrow R2$
Met4	MovR	PP	R2	<y>	( $R2$ ) $\rightarrow <y>$
	Halt				
				-1	
				x	

Y  
R2  
R3

В ЭВМ А2 команды пересылки не вырабатывают признаков их выполнения. Поэтому на начальном участке значение  $x$  складывается с нулем (команда сложения вырабатывает необходимые признаки).

Второй этап – запись адресов ячеек памяти.

51		MovL	PN	R2	0	$0 \rightarrow R2$
52		Add	PP	R2	<x>	$x+0 \rightarrow R2$
53		Jump	A	0	Met2	Переход на ветви 2 и 3 ( $x \leq 0$ )
54		Add	PN	R2	1	$x+1 \rightarrow R2$
55		Jump	F	0	Met4	Переход на запись ( $R2$ ) $\rightarrow y$
56	Met2	Jump	8	0	Met3	Переход на ветвь 3
57		MovL	PP	R2	<-1>	$-1 \rightarrow R2$
58		Jump	F	0	Met4	Переход на запись ( $R2$ ) $\rightarrow y$
59	Met3	Mul	PP	R2	<x>	$x^2 \rightarrow R2$
5A		MovR	PP	R2	R3	( $R2$ ) $\rightarrow R3$
5B		MovL	PN	R2	1	$1 \rightarrow R2$
5C		Sub	PP	R2	R3	$1-x^2 \rightarrow R2$
5D	Met4	MovR	PP	R2	<y>	( $R2$ ) $\rightarrow$ <y>
5E		Halt				
5F					-1	
60					x	
61					y	
62					R2	
63					R3	

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

В п.2.3.2 дополнительно проверена команда Jump.

### 2.3.3 Организация циклической программы

Наличие в ЭВМ А2 индекс-адресации позволяет сравнительно легко организовать разработку циклической программы.

Пусть нам требуется вычислить

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

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

Представим заданное выражение в виде

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

	14	MovR	NP	0	1	$0 \rightarrow \text{ячейка } 1$
	15	MovR	NP	0	$\langle S \rangle$	$0 \rightarrow \langle S \rangle$
	16	MovR	NP	0	$\langle k_c \rangle$	$0 \rightarrow \langle k_c \rangle$
Met	17	Add	PI	$\langle S \rangle$	$\langle x_1^* \rangle$	$S := S + x_i$
	18	Add	PN	1	1	$(\text{ячейка } 1) + 1 \rightarrow \text{ячейка } 1$
	19	Add	PN	$\langle k_c \rangle$	1	$k_c + 1 \rightarrow \langle k_c \rangle$
	1A	Comp	PN	$\langle k_c \rangle$	5	$k_c \geq 5 ?$
	1B	Jump	8	0	Met	<i>управление циклом</i>
	1C	MovL	PN	$\langle y \rangle$	8	$8 \rightarrow \langle y \rangle$
	1D	Mul	PP	$\langle y \rangle$	$\langle a \rangle$	$8a \rightarrow \langle y \rangle$
	1E	Sub	PP	$\langle y \rangle$	$\langle S \rangle$	$8 \cdot a - S \rightarrow \langle y \rangle$
	1F	Halt	<i>останов</i>			
	20		a	= 31		
	21		$x_1$	= 10		
	22		$x_2$	= -15		
	23		$x_3$	= 25		
	24		$x_4$	= 40		
	25		$x_5$	= 50		
	26		y			
	27		$k_c$	(счетчик)		

В начале программы обнуляется счетчик циклов  $k_c$  и ячейка для суммы  $S$ , а также заносится нуль в ячейку с адресом 1.

В команде с адресом 17 производится накопление суммы элементов  $x_i$ . При этом исполнительный адрес второго операнда должен в каждом цикле увеличиваться на единицу. Такое изменение адреса  $A2_{исп}$  обеспечивается индексной ячейкой 1, в которую до цикла заносится нулевое значение, а в цикле производится увеличение на единицу. Для управления циклом используется команда перехода с маской 1000. Эта команда передает управление на начало цикла (команда с меткой Met), если значение счетчика циклов меньше 5. Как только значение счетчика станет равным 5, управление передается следующей команде.

При отработке циклического фрагмента программы может произво-



даться переадресация (изменение адреса) первого операнда, второго операнда или обоих операндов одновременно. В качестве мнемонического приема рекомендуется записать символ «\*» над изменяемым операндом. В приведенном выше примере изменяется второй операнд, а в качестве индексной ячейки используется ячейка с адресом 1, содержимое которой можно считать константой переадресации. Поскольку индексная ячейка в данном случае должна определять переадресацию лишь второго операнда, то изменение ее содержимого выполняется в команде, имеющей адрес 18, с использованием непосредственной адресации для второго адреса этой команды (MA2 = 01).

Поскольку в команде `Add PI <S> <x1*>` для второго адреса указана индекс-адресация, то в первом цикле к адресу  $x_1$  добавляется 0, во втором – 1, в третьем – 2 и т.д. Следовательно, при выполнении этой команды к нулевой исходной сумме будут добавляться значения  $x_1, x_2, x_3, \dots$

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
001001110000000000000001	270001
0010011100000000000100111	270027
001001110000000000100001	270021
001111000010100000100010	3C2822
0011110100000000100000001	3D0101
001111010010011100000001	3D2701
011111010010011100000101	7D2705
110110000000000000010111	D80017
000111010010011100001000	1D2708
010111110010011100100000	5F2720
010011110010011100101000	4F2728
000000000000000000000000	000000
0000000000000000000011111	00001F
000000000000000000001010	00000A
111111111111111111110001	FFFFFF1
0000000000000000000011001	000019
0000000000000000000101000	000028
000000000000000000110010	000032

*Примечание.* В состав текстового файла не включено содержимое ячеек  $\langle u \rangle$  и  $\langle k_c \rangle$ , поскольку это содержимое формируется в программе, а не вводится извне.

### 2.3.4 Программирование подпрограмм

После реализации пп. 2.3.1, 2.3.2 и 2.3.3 остались непроверенными команды And, Or, Xor, Shr, Shl, Call и Ret, а также косвенный тип адресации.

Пусть нам требуется вычислить

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

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

Вычисления будем производить в подпрограмме, реализующий оператор

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

	MovL	PP	<l>	<a>	$l := a$
	MovL	PP	<m>	<b>	$m := b$
	MovL	PP	<n>	<c>	$n := c$
	MovL	PP	<p>	<d>	$p := d$
	Call	PP	1	Met1	Обращение к п/п
	MovL	PP	<u>	<w>	$u := w$
	MovL	PP	<l>	<e>	$l := e$
	MovL	PP	<m>	<f>	$m := f$
	MovL	PP	<n>	<g>	$n := g$
	MovL	PP	<p>	<h>	$p := h$
	Call	PP	1	Met1	Обращение к п/п
	MovL	PP	<v>	<w>	$v := w$
	Halt				
Met1	MovL	PP	R1	<l>	$l \rightarrow R1$
	Shl	PN	R1	3	$(R1) := l \leftarrow^3$
	And	PP	R1	<m>	$R1 := (R1) \wedge m$
	Shr	PN	R1	2	$(R1) := (R1) \rightarrow^2$
	Or	PP	R1	<n>	$(R1) := (R1) \vee n$
	Xor	PP	R1	<p>	$(R1) := (R1) \oplus p$
	MovL	PP	R1	<w>	$w := R1$
	Ret	PP	1	0	возврат из подпрограммы

a  
b  
c  
d  
e  
f  
g  
h  
l

m  
n  
p  
u  
v  
w

Команда Call («Обращение к подпрограмме») передает управление команде с меткой Met1, т.е. первой команде подпрограммы. Одновременно в ячейку 1 записывается адрес следующей команды, т.е. адрес команды которая должна быть активизирована после выхода из подпрограммы. Содержимое ячейки 1 используется в команде Ret для выхода из подпрограммы в вызывающую программу.

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

$$\begin{aligned}
 a &= 0001\ 0010\ 0011\ 0100\ 0101\ 0110_2 = 123456_{16} \\
 b &= 0111\ 1000\ 1001\ 1010\ 1011\ 1100_2 = 789ABC_{16} \\
 c &= 1101\ 1110\ 1111\ 1110\ 1101\ 1100_2 = DEFEDC_{16} \\
 d &= 0011\ 0101\ 0111\ 1001\ 1011\ 1101_2 = 3579BD_{16} \\
 a \leftarrow^3 &= 1001\ 0001\ 1010\ 0010\ 1011\ 0000_2 = 91A2B0_{16} \\
 (a \leftarrow^3) \wedge b &= 0001\ 0000\ 1000\ 0010\ 1011\ 0000_2 = 1082B0_{16} \\
 (a \leftarrow^3) \wedge b \rightarrow^2 &= 0000\ 0100\ 0010\ 0000\ 1010\ 1100_2 = 0420AC_{16} \\
 (a \leftarrow^3) \wedge b \rightarrow^2 \vee c &= 1101\ 1110\ 1111\ 1110\ 1111\ 1100_2 = DEF EFC_{16} \\
 u = (a \leftarrow^3) \wedge b \rightarrow^2 \vee c \oplus d &= 1110\ 1011\ 1000\ 0111\ 0100\ 0001_2 = \\
 &= EB8741_{16}
 \end{aligned}$$

### 3. УЧЕБНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА В1

#### 3.1 Архитектура ЭВМ В1

Учебная ЭВМ В1 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 В1 приведена на рис.5. В состав ЭВМ В1 входят следующие основные узлы:

- оперативная память MEMORY емкостью 256 16-разрядных ячеек;
- восьмиразрядный счетчик адреса команд SAK, предназначенный для хранения адреса следующей команды;
- 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;
- шестнадцатиразрядный регистр команд RK, хранящий выполняемую команду;
- шестнадцатиразрядный регистр-аккумулятор Acc;
- арифметико-логическое устройство, выполняющее операцию, заданную кодом операции команды;
- 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» в случае некорректности выполнения операции (переполнение аккумулятора 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.

В состав команды, содержащейся в регистре RK, входят код выполняемой операции KOP (биты 1 – 4), адрес ячейки модификации MP (биты 5 – 8) и адрес ячейки памяти A (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. Ячейки 1 – 15 отведены для модификаторов адреса.

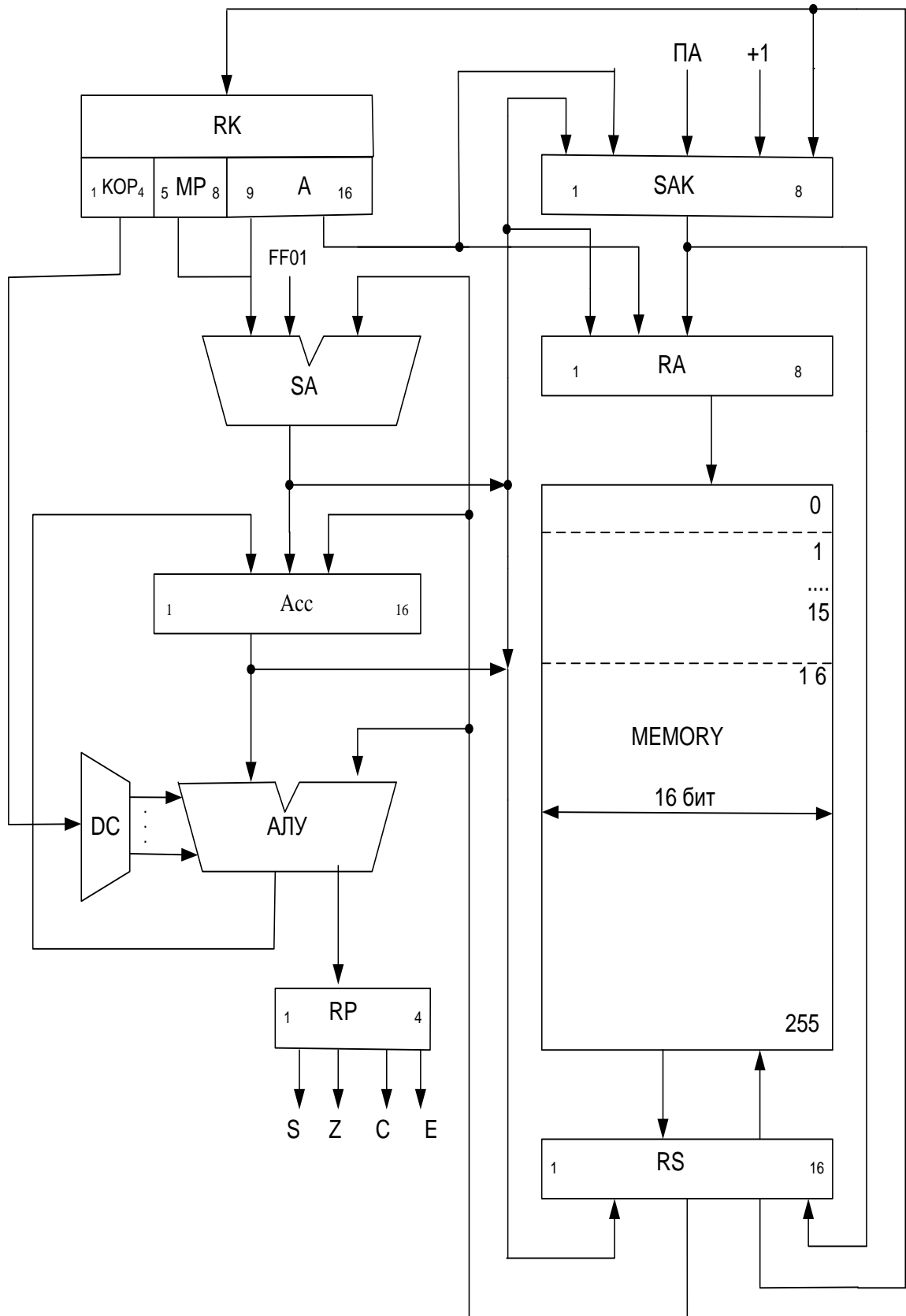


Рисунок 5 – Структурная схема ЭВМ В1

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

$$A_{исп} = A.$$

Ячейки модификации с адресами от 1 до 15 содержат в разрядах 9 – 16 модификатор адреса МА, а в разрядах 1 – 8 - счетчик повторений SP. Исполнительный адрес операнда формируется путем сложения в сумматоре SA содержимого поля А и значения МА ячейки, адрес которой записан в поле MP регистра команды (индекс-адресация):

$$A_{исп} = A + MEMORY(MP[9-16]).$$

Система команд ЭВМ В1, приведенная в табл.3, включает в себя:

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

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

Таблица 3. Система команд ЭВМ В1

Код опер.	Наименование	Признаки				Выполнение команды
		S	Z	C	E	
0000	Останов	-	-	-	-	
0001	Сложение	+	+	+	-	$(Acc) + (A_{исп}) \rightarrow Acc$
0010	Вычитание	+	+	+	-	$(Acc) - (A_{исп}) \rightarrow Acc$
0011	Умножение	+	+	+	-	$(Acc) * (A_{исп}) \rightarrow Acc$
0100	Деление	+	+	+	-	$(Acc) : (A_{исп}) \rightarrow Acc$
0101	Конъюнкция	+	+	-	-	$(Acc) \wedge (A_{исп}) \rightarrow Acc$
0110	Дизъюнкция	+	+	-	-	$(Acc) \vee (A_{исп}) \rightarrow Acc$
0111	Сдвиг вправо логический	+	+	-	-	$(Acc) \ A_{исп} \Rightarrow$
1000	Загрузка аккумулятора	-	-	-	-	$(A_{исп}) \rightarrow Acc$
1001	Сохранение аккумулятора	-	-	-	-	$(Acc) \rightarrow A_{исп}$
1010	Вызов подпрограммы	-	-	-	-	$(SAK) \rightarrow 16; A_{исп} \rightarrow SAK$
1011	Возврат из подпрограммы	-	-	-	-	$(16) \rightarrow SAK$
1100	Безусловный переход	-	-	-	-	$A_{исп} \rightarrow SAK$
1101	Усл.переход по $(Acc) < 0$	-	-	-	-	$A_{исп} \rightarrow SAK, \text{ если } S = 1$
1110	Усл.переход по $(Acc) = 0$	-	-	-	-	$A_{исп} \rightarrow SAK, \text{ если } Z = 1$
1111	Конец цикла	-	-	-	+	См.описание

В состав команд управления порядком выполнения программы входят: вызов подпрограммы, возврат из подпрограммы, безусловный переход, условный переход по  $(Acc) < 0$ , условный переход по  $(Acc) = 0$ , конец цикла и останов.

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  $A_{исп}$ .

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 вырабатывается признак  $E = 1$  и выполняется следующая команда. Если значение  $SP$  неотрицательное, то признак  $E$  сохраняет нулевое значение и переход осуществляется по адресу, заданному в поле  $A$ . В частности, это означает, что в команде «Конец цикла» применяется лишь прямая адресация.

### 3.2 Выполнение команд в ЭВМ В1

В ЭВМ В1 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  $SAC$ , в регистр команд  $RK$ . При этом значение адреса выполняемой команды из  $SAC$  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  $KOP$ ,  $MP$  и  $A$ . Для подготовки выборки следующей команды значение счетчика  $SAC$  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» и «Конец цикла», выполняется формирование исполнительного адреса  $A_{исп}$ . Если  $MP = 0000$ , то адрес  $A_{исп}$ , равный значению  $A$ , передается в счетчик  $SAC$  (для команд управления порядком выполнения программы) или в регистр  $RA$  (для арифметико-логических команд и команд работы с аккумулятором). Если  $MP$

$\neq 0000$ , то это значение поступает в разряды 5 – 8 регистра адреса RA и по нему выбирается ячейка модификации (разряды 5 – 8 определяют адреса 1 – 15 ячеек памяти). Содержимое разрядов 9 – 16 ячейки модификации через регистр RS подается на второй вход сумматора SA; на первый вход SA поступает значение адреса A; результат суммирования передается аналогично предыдущему в счетчик SAK или в регистр RA.

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 Асс. В команде «Сохранение аккумулятора» содержимое Асс 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 освобождающиеся слева разряды аккумулятора заполняются нулями, а разряды, выходящие при сдвиге за пределы Асс, 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 Асс заполняются нулями.

В арифметико-логических командах содержимое ячейки памяти с адресом, который задан в регистре RA, через регистр слова RS поступает на второй вход АЛУ, а на первый вход АЛУ передается содержимое аккумулятора Асс. Арифметико-логическое устройство выполняет операцию, заданную дешифратором DC, результат операции запоминается в аккумуляторе Асс. По результату операции в регистре RP формируются признаки 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 системе команд ЭВМ В1 нет специальной команды, реагирующей на возникновение признака  $C = 1$ . Поэтому при эмуляции машинной программы должны проверяться те команды, при выполнении которых может возникнуть некорректность решения, и при обнаружении 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.

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 перехода).

Сущность изменения регистра RP рассмотрим на конкретном примере. Предположим, что в программе выполнена команда сложения и при этом по-



лучен 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.

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 В1 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 безусловной передачи управления (БП) значение  $A_{исп}$  заносится в счетчик SAK. В командах условной передачи управления (УП) по  $(Acc) < 0$  и  $(Acc) = 0$  вначале анализируется значение соответствующего признака и, если этот признак равен 1, то происходит формирование  $A_{исп}$  и занесение его в счетчик SAK.

В команде «Конец цикла» формирование  $A_{исп}$  не производится. Здесь вначале изменяется содержимое ячейки модификации, при этом поле A данной команды передается в регистр RA, выбирается соответствующая ячейка модификации и ее содержимое через регистр RS поступает на второй вход сумматора SA. На первый вход сумматора подается код 1111111100000001 ( $FF01_{16}$ ). Сумматор SA выполняет суммирование с блокировкой переноса из разрядов 1 и 9 (при нумерации разрядов слева направо). Это означает, что из содержимого счетчика повторений SP вычитается единица, а к значению модификатора адреса MA добавляется единица. Новое значение ячейки мо-

дификации через регистр RS записывается в память. Если в результате суммирования получится неотрицательное значение SP, то формируется признак  $E = 0$ , а значение адреса A записывается в счетчик SAK, что приводит к повторному выполнению данного цикла. Если значение  $SP < 0$ , то формируется признак  $E = 1$ , значение A не заносится в SAK и, следовательно, выполняется следующая по порядку команда, т.е. производится выход из цикла.

*Пример.*

$$\begin{array}{r}
 \text{SP} \qquad \text{MA} \\
 0000 \ 0101 \ 0000 \ 0011 \\
 + 1111 \ 1111 \ 0000 \ 0001 \\
 \hline
 0000 \ 0100 \ 0000 \ 0100
 \end{array}$$

При втором выполнении команды «конец цикла» получим

$$\begin{array}{r}
 0000 \ 0100 \ 0000 \ 0100 \\
 + 1111 \ 1111 \ 0000 \ 0001 \\
 \hline
 0000 \ 0011 \ 0000 \ 0101
 \end{array}$$

При шестом выполнении той же команды имеем

$$\begin{array}{r}
 0000 \ 0000 \ 0000 \ 1000 \\
 + 1111 \ 1111 \ 0000 \ 0001 \\
 \hline
 1111 \ 1111 \ 0000 \ 1001
 \end{array}$$

При этом  $SP < 0$  (первый бит, определяющий знак числа, равен 1), что приводит к выходу из цикла.

*Примечание.* Из примера видно, что начальное значение параметра SP должно быть задано на единицу меньше требуемого количества выполнения цикла в программе.

В системе команд ЭВМ В1 признак результата  $(Acc) > 0$  не вырабатывается и соответственно отсутствует команда УП по  $(Acc) > 0$ . Тем не менее это не создает трудностей в реализации разветвляющейся программы, если одна из ее ветвей должна выполняться при  $(Acc) > 0$ . В этом случае после арифметической команды (например, команды сложения) записывают подряд команды условного перехода по  $(Acc) < 0$  и условного перехода по  $(Acc) = 0$ . Тогда при  $(Acc) > 0$  будет выполняться команда, непосредственно записанная после этих команд условного перехода.

### 3.3 Программирование в кодах ЭВМ В1

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

Машинная команда ЭВМ В1 состоит из трех частей: код операции, адрес ячейки модификации адреса и адрес операнда.

Для кода операции будем использовать следующие обозначения:

Halt – останов;  
Load – загрузка аккумулятора;  
Save – сохранение аккумулятора;  
Add – сложение;  
Sub – вычитание;  
Mul – умножение;  
Div – деление;  
And – конъюнкция;  
Or – дизъюнкция;  
Shr – сдвиг вправо логический;  
Jump – безусловный переход;  
Call – обращение к п/п;  
Ret – возврат из подпрограммы;  
Ifn – условный переход по (Acc) < 0;  
Ifz – условный переход по (Acc) = 0;  
Cicl – конец цикла.

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 01 <x>

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

### 3.3.1. Программирование арифметического выражения

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

14	Load	0	<a>	$a \rightarrow Acc$
15	Mul	0	<2>	$2a \rightarrow Acc$
16	Sub	0	<1>	$2a - 1 \rightarrow Acc$
17	Save	0	R	$2a - 1 \rightarrow R$
18	Load	0	<a>	$a \rightarrow Acc$
19	Mul	0	<25>	$25a \rightarrow Acc$
1A	Add	0	<338>	$25a + 338 \rightarrow Acc$
1B	Div	0	R	$(Acc)/(R) \rightarrow Acc$
1C	Save	0	<y>	$(Acc) \rightarrow <y>$
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. 3.3.1 проверены машинные команды Load, Save, Add, Sub, Mul, Div, Halt, а также прямой вид адресации.

### 3.3.2. Программирование разветвляющихся процессов

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

Блок-схемное решение поставленной задачи приведено на рис.2 и 3 в описании ЭВМ А1.

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>	$x \rightarrow Acc$
	Add	0 0	$x + 0 \rightarrow Acc$
	Ifn	0 Met1	УП по $(Acc) < 0$
	Ifz	0 Met2	УП по $(Acc) = 0$
	Add	0 <1>	$x + 1 \rightarrow Acc$
	Jump	0 Met3	БП на Met3
Met1	Load	0 <-1>	$-1 \rightarrow Acc$
	Jump	0 Met3	БП на Met3
Met2	Mul	0 <x>	$x * x \rightarrow Acc$
	Save	0 R	$x^2 \rightarrow R$
	Load	0 <1>	$1 \rightarrow Acc$
	Sub	0 R	$1 - x^2 \rightarrow Acc$
Met3	Save	0 <y>	$y \rightarrow <y>$
	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 выработан признак S или Z. В команде сложения нуль читается из нулевой ячейки памяти, в которой, как это указано в описании ЭВМ В1, всегда содержится нулевое значение. После выработки признака в программе записаны две команды условного перехода – по  $(Acc) < 0$  и  $(Acc) = 0$ . Тогда фрагмент программы после этих команд условного перехода будет соответствовать ветви алгоритма с условием  $x > 0$ . После реализации этой ветви в программе записан безусловный переход по адресу Met3, где выполняется запись результата в ячейку < y >.

Второй этап – запись адресов ячеек памяти.

55		Load	0	65	$x \rightarrow Acc$
56		Add	0	0	$x + 0 \rightarrow Acc$
57		Ifn	0	5B	УП по $(Acc) < 0$
58		Ifz	0	5D	УП по $(Acc) = 0$
59		Add	0	63	$x + 1 \rightarrow Acc$
5A		Jump	0	61	БП на Met3
5B	Met1	Load	0	64	$-1 \rightarrow Acc$
5C		Jump	0	61	БП на Met3
5D	Met2	Mul	0	65	$x * x \rightarrow Acc$
5E		Save	0	66	$x^2 \rightarrow R$
5F		Load	0	63	$1 \rightarrow Acc$
60		Sub	0	66	$1 - x^2 \rightarrow Acc$
61	Met3	Save	0	67	$y \rightarrow \langle y \rangle$
62		Halt			
63			1		
64			-1		
65			x		
66			R		
67			y		

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

В п.3.3.2 дополнительно проверены команды Jump, Ifn и Ifz.

### 3.3.3. 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 программа может иметь следующий вид:

14		Load	0	0	$0 \rightarrow Acc$
15		Save	0	$\langle S \rangle$	$0 \rightarrow \langle S \rangle$
16		Load	0	$\langle km \rangle$	константа $km \rightarrow Acc$

	17	Save	0	1	<i>константа km</i> → ячейка 1
Met	18	Load	0	< S >	$S \rightarrow Acc$
	19	Add	1	< $x_1^*$ >	$S := S + x_i$
	1A	Save	0	< S >	$S \rightarrow \langle S \rangle$
	1B	Cic1	1	Met	<i>управление циклом</i>
	1C	Load	0	< 8 >	$8 \rightarrow Acc$
	1D	Mul	0	< a >	$8 \cdot a \rightarrow Acc$
	1E	Sub	0	< S >	$8 \cdot a - S \rightarrow Acc$
	1F	Save	0	< y >	$y \rightarrow \langle y \rangle$
	20	Halt			<i>останов</i>
	21	0	4	0	<i>константа km</i>
	22		8		
	23		a = 31		
	24		$x_1 = 10$		
	25		$x_2 = -15$		
	26		$x_3 = 25$		
	27		$x_4 = 40$		
	28		$x_5 = 50$		
	29		y		
	2A		S		

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

По адресу 21 записана константа  $km$ , определяющая начальное содержимое ячейки модификации. Это содержимое также двумя командами передается в ячейку 0001, 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, определяющую исходное состояние ячейки модификации, записывается значение SP, на единицу меньшее количества повторений цикла.

В команде сложения 19 записан адрес элемента  $x_1$ , но при индекс-адресации исполнительный адрес этой команды в каждом цикле должен увеличиваться на единицу. Чтобы не забыть выполнить такое изменение адреса операнда, рекомендуется изменяемый операнд обозначать символом «\*». Тогда это будет означать, что в данной команде применяется индекс-адресация, а соответствующая ячейка модификации, имеющая в своей адресной части начальное нулевое значение, должна в каждом цикле увеличивать эту часть на единицу, при этом одновременно счетчик циклов будет уменьшаться на 1.



В п. 3.3.3 дополнительно проверена команда «конец цикла», а также индекс-адресация.

Запишем теперь рассматриваемую программу в машинном виде и сформируем в 16 с/с текстовый файл программы. При этом пусковой адрес программы, равный  $20_{10} = 14_{16}$ , запишется в первой строке файла в 10 с/с, в остальных строках – машинные команды и числа в 16 с/с.

	20
1000000000000000	8000
1001000000101010	902A
1000000000100001	8021
1001000000000001	9001
1000000000101010	802A
0001000100100100	1124
1001000000101010	902A
1111000100011000	F118
1000000000100010	8022
0011000000100011	3023
0010000000101010	2029
1001000000101001	9029
0000000000000000	0000
0000010000000000	0400
0000000000000100	0008
0000000000001111	001F
00000000000001010	000A
1111111111110001	FFF1
00000000000011001	0019
00000000000101000	0028
00000000000110010	0032

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

### 3.3.4. Программирование подпрограмм

После реализации пп. 3.3.1, 3.3.2 и 3.3.3 остались непроверенными команды And, Or, Shr, Call и Ret.

Пусть нам требуется вычислить

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

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

Вычисления будем производить в подпрограмме, реализующий оператор

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

	Load	0	a	
	Save	0	l	$l := a$
	Load	0	b	
	Save	0	m	$m := b$
	Load	0	c	
	Save	0	n	$n := c$
	Call	Met1		Обращение к п/п
	Load	0	w	
	Save	0	u	$u := w$
	Load	0	d	
	Save	0	l	$l := d$
	Load	0	e	
	Save	0	m	$m := e$
	Load	0	f	
	Save	0	n	$n := f$
	Call	Met1		Обращение к п/п
	Load	0	w	
	Save	0	v	$v := w$
	Halt			
Met1	Load	0	l	$l \rightarrow Acc$
	Shr	0	3	$(Acc) \rightarrow^3$
	And	0	m	$(Acc) \wedge m \rightarrow Acc$
	Shr	0	2	$(Acc) \rightarrow^2$
	Or	0	n	$(Acc) \vee n \rightarrow Acc$
	Save	0	w	$(Acc) \rightarrow \langle w \rangle$
	Ret	0	0	возврат из п/п
			a	
			b	
			c	
			d	
			e	
			f	
			l	
			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  $a$ ,  $b$ ,  $c$ .

$$a = 1101\ 0010\ 1011\ 0100\ 1111\ 1000_2 = D2B4F8_{16}$$

$$b = 0111\ 1000\ 1001\ 1010\ 1011\ 1100_2 = 789ABC_{16}$$

$$c = 1101\ 1110\ 1111\ 1110\ 1101\ 1100_2 = DEFEDC_{16}$$

$$a \rightarrow^3 = 0001\ 1010\ 0101\ 0110\ 1001\ 1111_2 = 1A569F_{16}$$

$$(a \rightarrow^3) \wedge b = 0001\ 1000\ 0001\ 0010\ 1001\ 1100_2 = 18129C_{16}$$

$$(a \rightarrow^3) \wedge b \rightarrow^2 = 0000\ 0110\ 0000\ 0100\ 1010\ 0111_2 = 0604A7_{16}$$

$$u = (a \rightarrow^3) \wedge b \rightarrow^2 \vee c = 1101\ 1110\ 1111\ 1110\ 1111\ 1111_2 = DEF EFF_{16}$$

## 4. УЧЕБНАЯ ВЫЧИСЛИТЕЛЬНАЯ МАШИНА В2

### 4.1 Архитектура ЭВМ В2

Учебная ЭВМ В2 представляет собой трехадресную ЭВМ с возможностью модификации адресов.

Структурная схема ЭВМ В2 приведена на рис.6. В состав ЭВМ В2 входят следующие основные узлы:

- оперативная память MEMORY емкостью 256 32-разрядных ячеек;
- восьмиразрядный счетчик адреса команд SAK, предназначенный для хранения адреса следующей команды;
- 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;
- 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;
- тридцатидвухразрядный регистр RR, 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 регистра RR или попытка деления на нуль). Состоянием четвертого бита регистра RP управляет команда «Конец цикла». Этим битам на схеме ЭВМ соответствуют 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 КОП (биты 1 – 4), адрес ячейки модификации МР (биты 5 – 8), базовые адреса А1 (биты 9 – 16), А2 (биты 17 – 24) и А3 (биты 25 – 32) первого операнда, второго операнда и результата.

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.

Адрес ячейки модификации может принимать значения от 0 до 15, т.е. он определяет первые 16 ячеек памяти MEMORY.

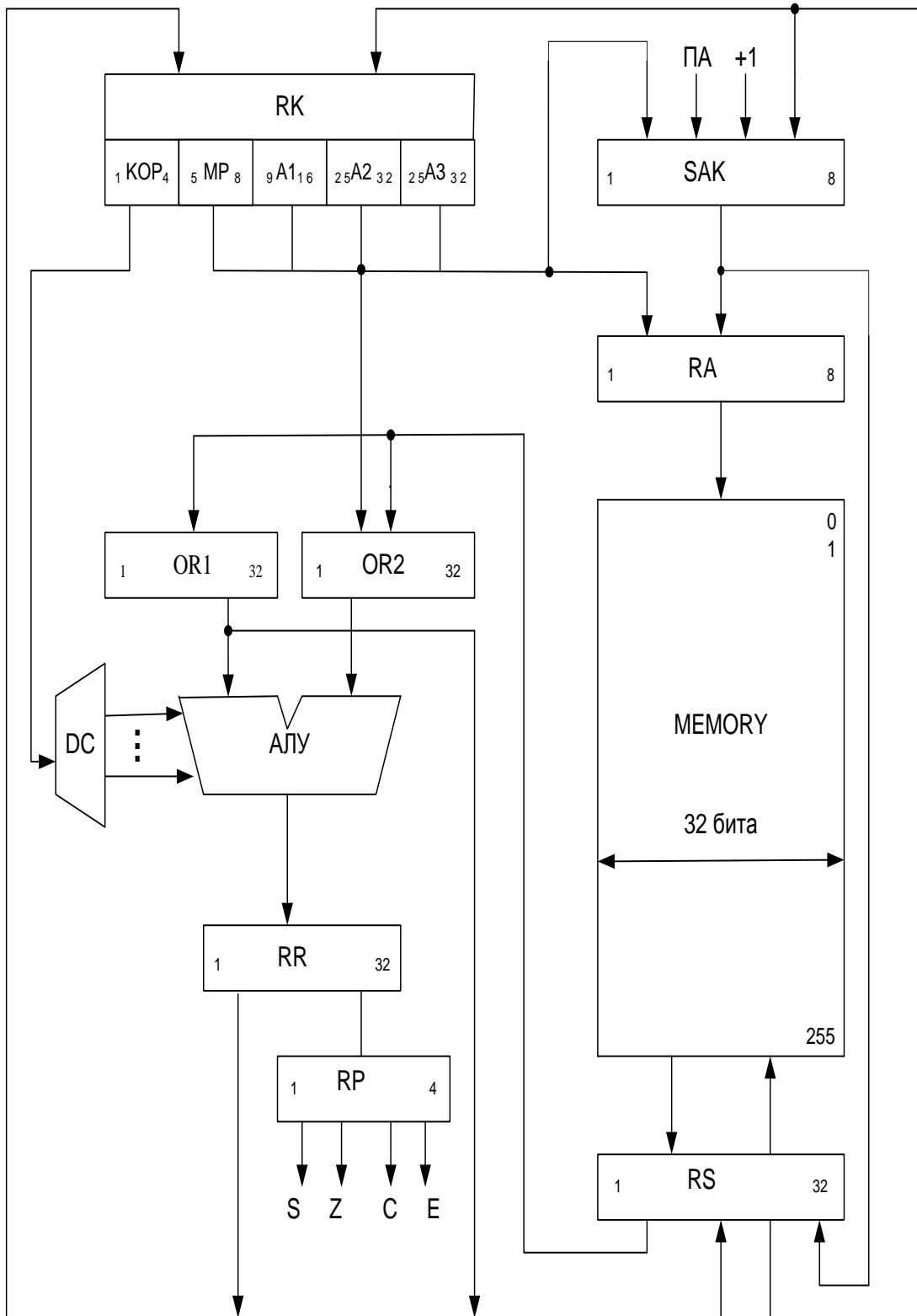


Рисунок 6 – Структурная схема ЭВМ В2

Структурно содержимое ячейки модификации разделяется на следующие поля: счетчик повторений SP (биты 1 – 8), модификатор MA1 адреса первого операнда (биты 9 – 16), модификатор MA2 адреса второго операнда (биты 17 – 24) и модификатор MA3 адреса результата (биты 25 – 32). Исполнительные адреса, по которым из памяти выбираются значения операндов и записывается результат, формируются путем сложения содержимого битов 9 – 32 ячейки модификации с полями A1, A2, A3 регистра команды:

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

Система команд ЭВМ В2, приведенная в табл.4, включает в себя:

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

Таблица 4. Система команд ЭВМ В2

Код опер.	Наименование	Признаки				Выполнение команды
		S	Z	C	E	
0000	Останов	–	–	–	–	
0001	Сложение	+	+	+	–	$(A1_{исп}) + (A2_{исп}) \rightarrow A3_{исп}$
0010	Вычитание	+	+	+	–	$(A1_{исп}) - (A2_{исп}) \rightarrow A3_{исп}$
0011	Умножение	+	+	+	–	$(A1_{исп}) * (A2_{исп}) \rightarrow A3_{исп}$
0100	Деление	+	+	+	–	$(A1_{исп}) : (A2_{исп}) \rightarrow A3_{исп}$
0101	Отрицание	+	+	–	–	$\neg(A1_{исп}) \rightarrow A3_{исп}$
0110	Конъюнкция	+	+	–	–	$(A1_{исп}) \wedge (A2_{исп}) \rightarrow A3_{исп}$
0111	Дизъюнкция	+	+	–	–	$(A1_{исп}) \vee (A2_{исп}) \rightarrow A3_{исп}$
1000	Сумма по модулю 2	+	+	–	–	$(A1_{исп}) \oplus (A2_{исп}) \rightarrow A3_{исп}$
1001	Сдвиг влево логический	+	+	–	–	$(A1_{исп}) \ll A2_{исп}; 0 \rightarrow A3_{исп}$
1010	Сдвиг вправо логический	+	+	–	–	$(A1_{исп}) \gg A2_{исп}; 0 \rightarrow A3_{исп}$
1011	Вызов подпрограммы	–	–	–	–	$(SAK) \rightarrow A3_{исп}; A1_{исп} \rightarrow SAK$
1100	Возврат из подпрограммы	–	–	–	–	$(A3_{исп}) \rightarrow SAK$
1101	Безусловный переход	–	–	–	–	$A1_{исп} \rightarrow SAK$
1110	Условный переход	–	–	–	–	См.описание
1111	Конец цикла	–	–	–	+	См.описание

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

Команды сдвига: сдвиг влево логический и сдвиг вправо логический.

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.

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  $A1_{исп}$ .

## 4.2 Выполнение команд в ЭВМ В2

В ЭВМ В2 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, MP, 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» и «Конец цикла», выполняется формирование исполнительных адресов  $A1_{исп}$ ,  $A2_{исп}$  и  $A3_{исп}$ . При этом значение поля MP (биты 5 – 8 регистра RK) передается в регистр RA, ячейка памяти с адресом, заданном в RA, поступает в операционный регистр OR1, поля A1, A2, A3 (биты 9 – 32 RK) передаются в биты 9 – 32 операционного регистра OR2. Арифметико-логическое устройство АЛУ производит сложение битов 9 – 32 регистров OR1 и OR2, при этом блокируются переносы из разрядов 9, 17 и 25. Результат сложения через регистр RR возвращается в биты 9 – 32 регистра команд RK. Следовательно, в регистре команд RK будут находиться исполнительные адреса операндов.

Если  $MP = 0000$ , то поскольку при этом заранее известно, что в ячейке 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  $A1_{ucn}$  (биты 9 – 16 RK) передается в RA, содержимое ячейки MEMORY(RA) через регистр слова RS поступает в операционный регистр OR1. Вслед за этим адрес  $A2_{ucn}$  (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 КОП, над содержимым регистров OR1 и OR2. Результат выполнения операции записывается в регистр RR. После этого результат операции из регистра RR пересылается в буферный регистр RS, и записывается в память по адресу  $A3_{ucn}$ , который предварительно был передан из регистра команд RK в регистр адреса RA.

В команде сдвига содержимое ячейки с адресом  $A1_{ucn}$ , как и в арифметико-логических операциях, записывается в операционный регистр OR1. Однако адрес  $A2_{ucn}$  используется здесь не для выборки операнда из памяти, а для определения количества сдвигов содержимого регистра OR1. Адрес  $A2_{ucn}$  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  $A2_{ucn}$  разрядов влево или вправо. Освобождающиеся при сдвиге разряды регистра OR1 заполняются нулями, а разряды, выходящие при сдвиге за пределы OR1, исчезают. Если  $A2_{ucn} > 31$ , то сдвиг разрядов регистра OR1 не выполняется; в этом случае по команде сдвига все разряды в OR1 заполняются нулями. После этого содержимое регистра OR1 записывается по адресу  $A1_{ucn}$ .

В обеих командах сдвига выполняется дополнительная работа – запись нуля в ячейку с адресом  $A3_{ucn}$ . Следовательно, финальная часть работы команды сдвига разделяется на два этапа:

- запись содержимого регистра OR1 в ячейку памяти  $A1_{ucn}$ ;
- занесение нуля в ячейку памяти  $A3_{ucn}$ .
- 

Указанное свойство команды сдвига можно использовать для обнуления ячеек памяти. В принципе обнуление содержимого ячейки можно выполнить тремя способами:

- 1) 0001 0000 0000 0000 A ( 0001 0 0 0 A )
- 2) 1001 0000 A 0000 A ( 1001 0 A 0 A )
- 3) 1010 0000 A 0000 A ( 1010 0 A 0 A )

В первом случае в регистры OR1 и OR2 посылаются содержимое нуле-



вой ячейки памяти, т.е. значение 0. После этого производится сложение нуля с нулем и засылка результата в ячейку А. Во втором и в третьем случаях содержимое ячейки А сдвигается на нуль разрядов, после этого в ячейку А записывается ее исходное состояние, а затем заносится нуль.

*Примечание.* Записать команду

1001 0000 0000 0000 А (1001 0 0 0 А)

нельзя, так как после сдвига содержимого регистра OR1 полученный результат записывается по адресу  $A1_{исп}$ , т.е. в данном случае в нулевую ячейку, что запрещено в соответствии с описанием ЭВМ В2.

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 результатов RR или попытка деления на нуль). Требуемые изменения регистра RP отражены в табл.4.

Для логических операций и для команд сдвига, как было ранее указано, 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.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 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 состоянии «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). 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 В2 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 В2 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 адрес  $A1_{исп}$ , 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  $A2_{исп}$  и  $A3_{исп}$  в данной команде не используются.

*Примечание.* Вполне очевидно, что адреса  $A3_{исп}$  в команде «Вызов подпрограммы» и  $A1_{исп}$  в команде «Возврат из подпрограммы» должны быть одинаковыми.

В команде «Конец цикла» формирование исполнительных адресов не производится. Здесь вначале значение поля MP (биты 5 – 8 RK) поступает в регистр RA и по адресу, записанному в RA, выбирается ячейка памяти, содержимое которой через регистр слова RS загружается в операционный регистр OR1. Следовательно, в регистре OR1 содержится в данный момент изменяемая в дальнейшем ячейка модификации 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)

КМ1, КМ2, КМ3 модификаторов МА1, МА2, МА3. 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 МА1, МА2, МА3 увеличиваются на значения соответствующих констант КМ1, КМ2, КМ3. Результат суммирования, выполненный в АЛУ, записывается в регистр RR. 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 регистра RR записывается в память по адресу, заданному в поле MP 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.

Рассмотрим работу команды «конец цикла» на конкретном примере.

Пусть команда «конец цикла» имеет следующий вид:

1111 1000 0110 0100 0111 1000 1011 1001

Здесь  $MP = 1000_2 = 8_{10}$

$A1 = 0110 0100_2 = 100_{10}$

$A2 = 0111 1000_2 = 120_{10}$

$A3 = 1011 1001_2 = 185_{10}$

Предположим, что в ячейках 8 и 100 записано следующее:

(8) = 0000 0101 0000 0010 0000 0000 0000 0001

(100) = 0000 0000 0000 0100 0000 0001 0000 0110

или в десятичной записи

(8) = 5 2 0 1

(100) = 0 1 2 6

Тогда регистры OR1 и OR2 будут иметь такое содержимое:

(OR1) = 0000 0101 0000 0010 0000 0000 0000 0001

(OR2) = 1111 1111 0000 0001 0000 0010 0000 0110

При сложении (OR1) и (OR2) получим:

(RR) = 0000 0100 0000 0011 0000 0010 0000 0111,

т.е. значение SP уменьшилось на 1, адрес A1 увеличился на 1, адрес A2 – на 2, адрес A3 – на 6.

### 4.3 Программирование в кодах ЭВМ В2

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

Машинная команда ЭВМ В2 состоит из пяти частей: код операции, адрес ячейки модификации и адреса трех операндов.

Для кода операции будем использовать следующие обозначения:

Halt – останов;  
Add – сложение;  
Sub – вычитание;  
Mul – умножение;  
Div – деление;  
Not – отрицание;  
And – конъюнкция;  
Or – дизъюнкция;  
Xor – сумма по модулю 2 (исключающее ИЛИ);  
Shr – сдвиг вправо логический;  
Shl – сдвиг влево логический;  
Call – обращение к подпрограмме;  
Ret – возврат из подпрограммы;  
Jump – безусловный переход;  
If – условный переход;  
Cicl – конец цикла.

Адрес ячейки модификации будем записывать в 16 с/с (диапазон от 0 до F).

В адресной части команды может быть:

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

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

Непосредственный операнд может быть лишь в команде сдвига на месте адреса A2, в этом случае он рассматривается как константа сдвига.

*Примеры.*

а) Shl 0 R 4 140

Выполняется сдвиг влево на 4 разряда содержимого ячейки с адресом R, одновременно записывается 0 в ячейку с адресом 140.

б) Add 0 <x> <25> <y>

К значению переменной  $x$  добавляется константа 25, полученный результат присваивается переменной  $y$ , т.е. записывается по адресу, который отведен для переменной  $y$ .

#### 4.3.1 Программирование арифметического выражения

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

14	Mul	0	<a>	<2>	R1	$2a \rightarrow R1$
15	Sub	0	R1	<1>	R1	$2a - 1 \rightarrow R1$

16	Mul	0	<a>	<25>	R2	25 a → R2
17	Add	0	R2	<338>	R2	25 a + 338 → R2
18	Div	0	R2	R1	y	(R2)/(R1) → <y>
19	Halt					
1A		1				
1B		2				
1C		25				
1D		338				
1E		a				
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  $14_{16} = 20_{10}$ .

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

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

Здесь адреса 1F, 20 и 21 не отображаются, поскольку значение переменной y и содержимое буферных ячеек R1 и R2 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  
06007415437  
04007615037  
06007416040

```

02010016440
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. 4.3.1 проверены машинные команды Mul, Sub, Add, Div и Halt.

### 4.3.2 Программирование разветвляющихся процессов

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

Блок-схемное решение поставленной задачи приведено на рис. 2 и 3 в описании ЭВМ А1.

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 Met1..Met4.

	Add	0	0	<x>	R	$x+0 \rightarrow R$
	If	0	Met2	Met3	Met1	Переход на вторую, третью и первую ветви
Met1	Add	0	<x>	<1>	<y>	$x+1 \rightarrow \langle y \rangle$
	Jump	0	Met4	0	0	
Met2	Add	0	0	<-1>	<y>	$-1 \rightarrow \langle y \rangle$
	Jump	0	Met4	0	0	
Met3	Mul	0	<x>	<x>	R	$x^2 \rightarrow R$
	Sub	0	<1>	R	<y>	$1-x^2 \rightarrow \langle y \rangle$
Met4	Halt					
		1				
		-1				
		x				
		R				
		y				

Второй этап – запись адресов ячеек памяти.

	55	Add	0	0	<x>	R	$x+0 \rightarrow Ac$
	56	If	0	Met2	Met3	Met1	Переход на вторую, третью и первую ветви
Met1	57	Add	0	<x>	<1>	<y>	$x+1 \rightarrow \langle y \rangle$
	58	Jump	0	Met4	0	0	
Met2	59	Add	0	0	<-1>	<y>	$-1 \rightarrow \langle y \rangle$
	5A	Jump	0	Met4	0	0	
Met3	5B	Mul	0	<x>	<x>	R	$x^2 \rightarrow R$
	5C	Sub	0	<1>	R	<y>	$1-x^2 \rightarrow \langle y \rangle$
Met4	5D	Halt					
	5E		1				
	5F		-1				
	60		x				
	61		R				
	62		y				

Этапы 3 и 4 аналогичны этапам 2 и 3 предыдущей программы.  
В п. 4.3.2 дополнительно проверены команды Jump и If.

### 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:

	14	Add	0	0	0	<S>	$0 \rightarrow \langle S \rangle$
	15	Add	0	< $k_m$ >	0	1	$k_m \rightarrow \text{ячейка } 1$
Met1	16	Add	1	<S>	< $x_1^*$ >	<S>	$S + x_i \rightarrow \langle S \rangle$
	17	Cic1	1	< $k_c$ >	Met1	Met2	<i>управление циклом</i>
Met2	18	Mul	0	<8>	<a>	<y>	$8a \rightarrow \langle y \rangle$

19	Sub	0	<y>	<S>	<y>	$8 a - S \rightarrow \langle y \rangle$
1A	Halt					<i>останов</i>
1B	0	4	0	0	0	<i>модификатор <math>k_m</math></i>
1C	0	0	0	1	0	<i>к - та переадресации <math>k_c</math></i>
1D		8				
1E		a = 31				
1F		$x_1 = 10$				
20		$x_2 = -15$				
21		$x_3 = 25$				
22		$x_4 = 40$				
23		$x_5 = 50$				
24		y				
25		S				

Здесь Met1 - обозначение метки адреса перехода, < y > - обозначение адреса ячейки, отведенной для переменной y (в данном случае 24).

По адресу 1B записано исходное (начальное) значение  $k_m$  ячейки модификации. В частности, ее 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 меньше заданного количества повторений цикла (в данном случае 4 вместо 5).

Первая команда программы обнуляет ячейку, отведенную для размещения переменной S. Для этого в команде сложения нуль складывается с нулем, а полученный результат присваивается переменной S.

Вторая команда, складывая содержимое ячейки 1B с нулем, фактически пересылает исходное значение ячейки модификации в ячейку с адресом 0001, выбранную в этой программе в качестве реальной ячейки модификации.

Третья команда добавляет к переменной S значение элемента  $x_i$ . Так как адресная часть ячейки модификации 0001 при первом выполнении данной команды нулевая, то в первом цикле к переменной S 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 1C.

Дальше срабатывает команда «Конец цикла», использующая ячейку





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

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

Вычисления будем производить в подпрограмме, реализующий опера-  
тор

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

	Add	0	0	<a>	<l>	l:=a
	Add	0	0	<b>	<m>	m:=b
	Add	0	0	<c>	<n>	n:=c
	Add	0	0	<d>	<p>	p:=d
	Call	0	Met1	0	R1	обращение к п/п
	Add	0	0	<w>	<u>	u:=w
	Add	0	0	<e>	<l>	l:=e
	Add	0	0	<f>	<m>	m:=f
	Add	0	0	<g>	<n>	n:=g
	Add	0	0	<h>	<p>	p:=h
	Call	0	Met1	0	R1	обращение к п/п
	Add	0	0	<w>	<v>	v:=w
	Halt					
Met1	Not	0	<l>	0	R2	$\neg l \rightarrow R2$
	Shl	0	R2	3	R2	$(R2) \leftarrow^3$
	And	0	0	<m>	R4	$m \rightarrow (R4)$
	Shr	0	R4	2	R3	$(R4) \rightarrow^2$
	Or	0	R3	<n>	R3	$(R3) \vee n \rightarrow (R3)$
	And	0	R2	R3	R2	$(R3) \wedge (R2) \rightarrow (R2)$
	Xor	0	R1	<p>	<w>	$(R2) \oplus p \rightarrow \langle w \rangle$
	Ret	0	R1	0	0	возврат из п/п
			a			
			b			
			c			
			d			
			e			
			f			
			g			
			h			
			u			
			v			
			w			
			R1			
			R2			
			R3			
			R4			

Остальные этапы программной реализации выполняются аналогично предыдущему.

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

$$a = 0101\ 1001\ 0111\ 0000\ 1111\ 1010\ 1110\ 1001_2 = 5970FAE9_{16}$$

$$b = 1100\ 0011\ 0100\ 0101\ 0110\ 0111\ 1000\ 1001_2 = C3456789_{16}$$

$$c = 0110\ 1010\ 1011\ 1100\ 1101\ 1110\ 1111\ 0000_2 = 6ABCDEF0_{16}$$

$$d = 0001\ 0011\ 0101\ 0111\ 1001\ 1011\ 1101\ 1111_2 = 13579BDF_{16}$$

$$\neg a = 1010\ 0110\ 1000\ 1111\ 0000\ 0101\ 0001\ 0110_2 = A68F0516_{16}$$

$$(\neg a) \leftarrow^3 = 0011\ 0100\ 0111\ 1000\ 0010\ 1000\ 1011\ 0000_2 = 347828B0_{16}$$

$$b \rightarrow^2 = 0011\ 0000\ 1101\ 0001\ 0101\ 1001\ 1110\ 0010_2 = 30D159E2_{16}$$

$$(b \rightarrow^2) \vee c = 0111\ 1010\ 1111\ 1101\ 1111\ 1111\ 1111\ 0010_2 = 7AFDFFF2_{16}$$

$$\begin{aligned} ((\neg a) \leftarrow^3) \wedge ((b \rightarrow^2) \vee c) &= 0011\ 0000\ 0111\ 1000\ 0010\ 1000\ 1011\ 0000_2 = \\ &= 307828B0_{16} \end{aligned}$$

$$u = 0010\ 0011\ 0010\ 1111\ 1011\ 0011\ 0110\ 1111_2 = 232FB36F_{16}$$

## СОДЕРЖАНИЕ

Стр.

1. Учебная вычислительная машина А1 .....	3
1.1. Архитектура ЭВМ А1 .....	3
1.2. Выполнение команд в ЭВМ А1 .....	6
1.3. Программирование в кодах ЭВМ А1 .....	9
1.3.1. Программирование арифметического выражения .....	10
1.3.2. Программирование разветвляющихся процессов .....	11
1.3.3. Организация циклической программы.	14
1.3.4. Программирование подпрограмм .....	16
2. Учебная вычислительная машина А2 .....	20
2.1. Архитектура ЭВМ А2 .....	20
2.2. Выполнение команд в ЭВМ А2 .....	23
2.3. Программирование в кодах ЭВМ А2 .....	27
2.3.1. Программирование арифметического выражения .....	28
2.3.2. Программирование разветвляющихся процессов .....	30
2.3.3. Организация циклической программы.	31
2.3.4. Программирование подпрограмм .....	34
3. Учебная вычислительная машина В1 .....	36
3.1. Архитектура ЭВМ В1 .....	36
3.2. Выполнение команд в ЭВМ В1 .....	39
3.3. Программирование в кодах ЭВМ В1 .....	42
3.3.1. Программирование арифметического выражения .....	44
3.3.2. Программирование разветвляющихся процессов .....	45
3.3.3. Организация циклической программы.	47
3.3.4. Программирование подпрограмм .....	49
4. Учебная вычислительная машина В2 .....	52
4.1. Архитектура ЭВМ В2 .....	52
4.2. Выполнение команд в ЭВМ В2 .....	55
4.3. Программирование в кодах ЭВМ В2 .....	59
4.3.1. Программирование арифметического выражения .....	60
4.3.2. Программирование разветвляющихся процессов .....	62
4.3.3. Организация циклической программы.	63
4.3.4. Программирование подпрограмм .....	65