

# Программирование в кодах ЦВМ

**Основные шаги при программировании в кодах:**

- 1) разработка подробной схемы алгоритма;**
- 2) распределение памяти (определение адресов ячеек для размещения констант и переменных, определение пускового адреса программы);**
- 3) составление программы на специальном бланке (таблица с колонками «адрес», «код команды», «примечание»).**

# Программирование в кодах ЦВМ

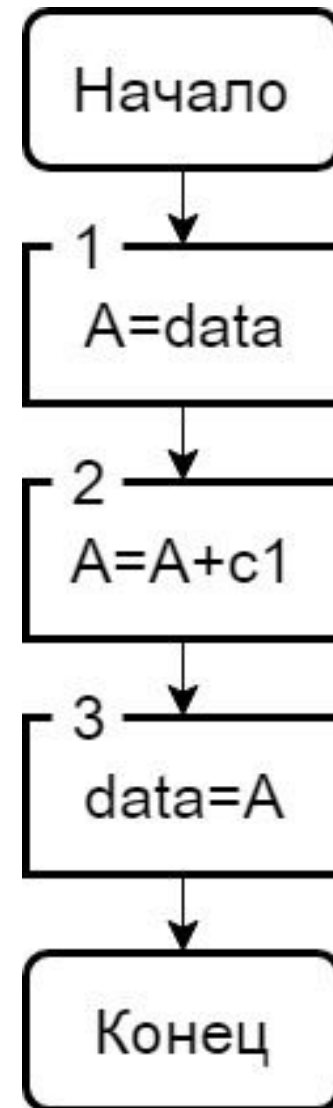
Разработать программу для увеличения переменной data на 1.

1) разработка подробной схемы алгоритма;

A – аккумулятор

data – переменная

C1 – константа равная единице



# Программирование в кодах ЦВМ

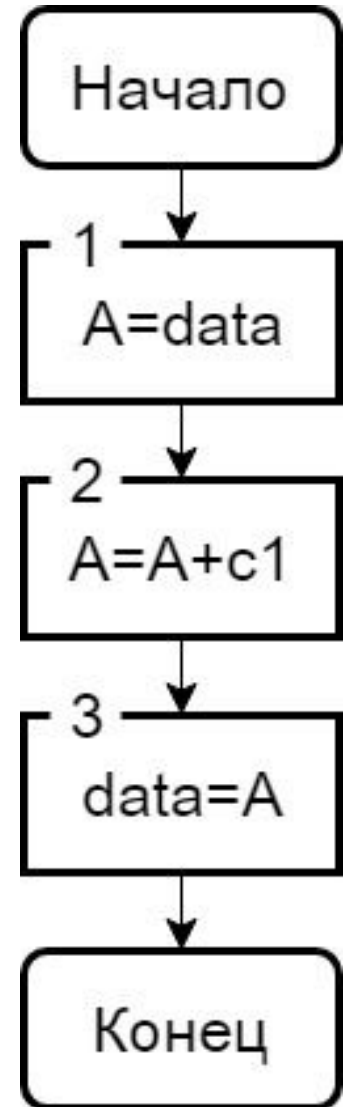
Разработать программу для увеличения переменной data на 1.

2) распределение памяти (определение адресов ячеек для размещения констант и переменных, определение пускового адреса программы);

data –  $01E_{16}$

C1 –  $021_{16}$

пусковой адрес (адрес начала программы)  
-  $014_{16}$

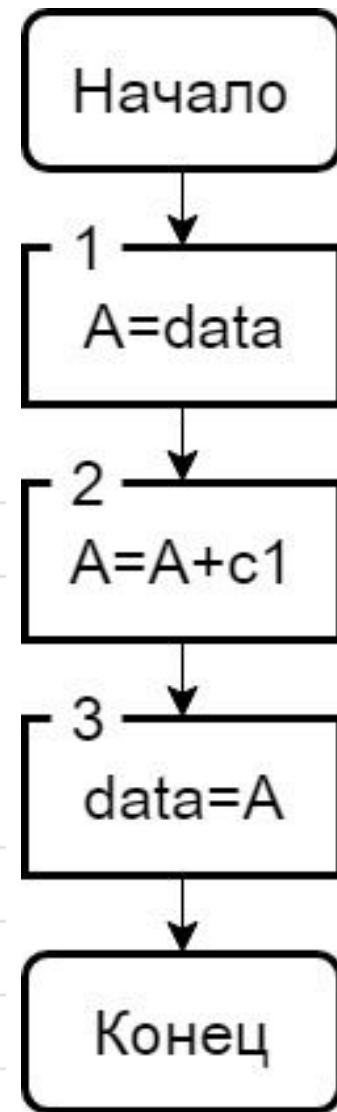


# Программирование в кодах ЦВМ

Разработать программу для увеличения переменной data на 1.

3) составление программы на специальном бланке (таблица с колонками «адрес», «код команды», «примечание»).

адрес	код команды						примечание
	код операции	адресация (в 1 формате)		адрес	операнда (в 1 формате)		
014	0 0	0 0	1 E			A=data	
017	1 8	0 0	2 1			A=a+c1	
01A	0 C	0 0	1 E			data=A	
01D	F F					Останов	



# Программирование в кодах ЦВМ

```
      | ; Пример 1
H inc      | inc      start    20
T 014 00001E |          lda      data
T 017 180021 |          add      c1
T 01A 0C001E |          sta      data
T 01D FF      |          hlt
      | ; Данные
```

T 01E  
T 021  
E 014

The screenshot shows a simulator window with two main panes. The left pane, titled 'TCom Bo...', contains assembly code and registers. The right pane, titled 'Memory Board', shows a memory dump grid. A red box highlights the memory addresses 01E through 021, and a blue box highlights address 014. The memory dump shows the following data:

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	*00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
010	00	00	00	00	00	00	1E	18	00	21	0C	00	1E	FF	00	00
020	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

The left pane shows the following registers and controls:

- CC: 0
- A: 000000
- X: 000000
- PC: 000
- L: 000000
- SA: 014
- MA: 000
- MD: (empty)
- STOP button (highlighted in blue)
- RESET, AUTO, START, STOP, CONTINUE buttons

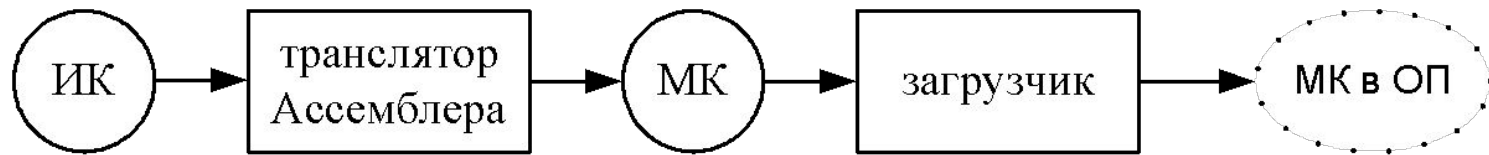
# Программирование на языке Ассемблер

**Ассемблер** – машинно-ориентированный язык, расширенный средствами управления трансляцией, средствами связывания программ и макросредствами.

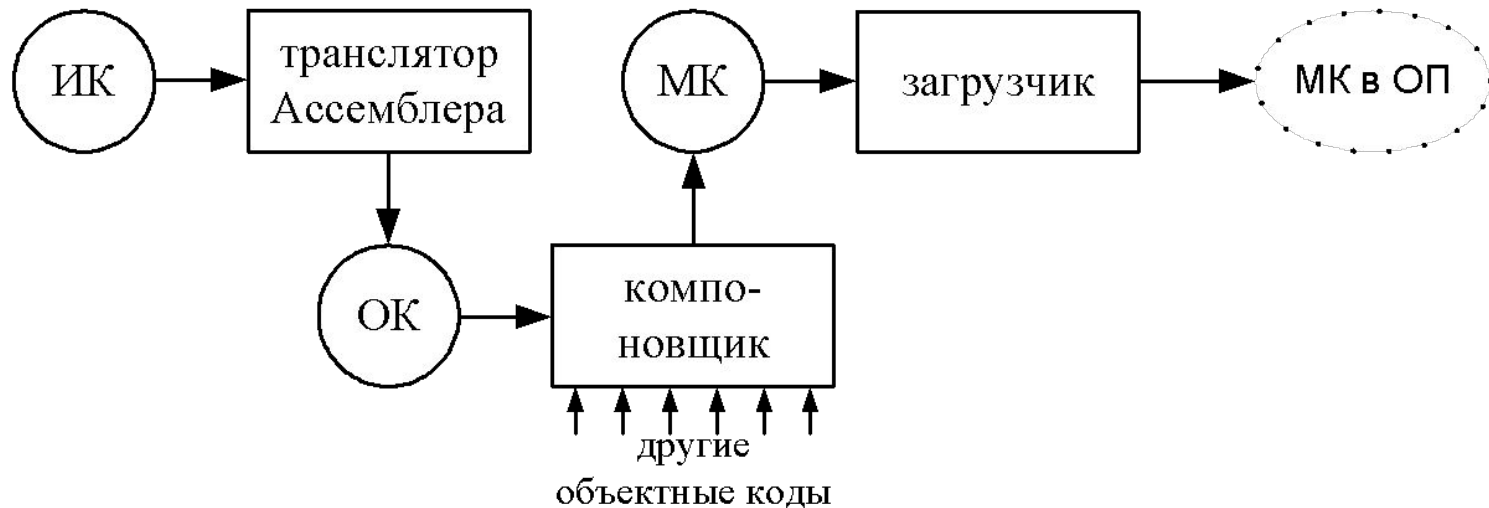
**Ассемблер** – это программа, генерирующая машинный код из исходного кода на языке Ассемблер.

# Программирование на языке Ассемблер

Упрощённая схема трансляции



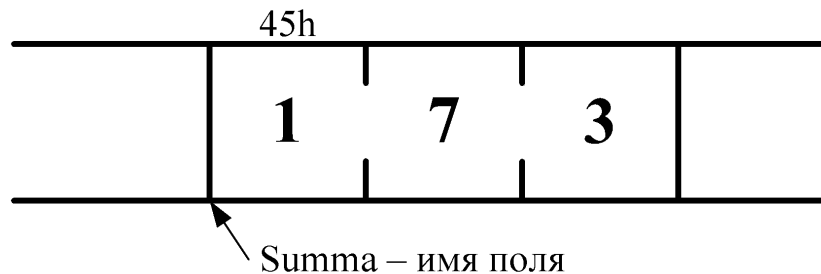
Реальная схема трансляции



# Программирование на языке Ассемблер

**Характерные черты языка Ассемблер:**

- 1) использование символических имён операций;**
- 2) использование символических имён полей памяти вместо адресов:**



**Имя поля заменяет его адрес,  
а не значение, т.е. Summa = 45h**

- 3) автоматическое распределение памяти;**
- 4) исходный текст программы на Ассемблере состоит из операторов, каждый из которых занимает отдельную строку.**



# Программирование на языке Ассемблер

**Операторы в УЦВМ:**

**1) оператор машинной команды – символическая запись  
машинной команды:**

**[<метка>] <симв. КОп> [<операнд>] [;<комментарий>]**

**2) оператор псевдокоманды (директива):**

**[<метка>] <директива> <операнд> [; <комментарий>]**

**Машинные команды управляют процессором,  
а псевдокоманды – транслятором.**

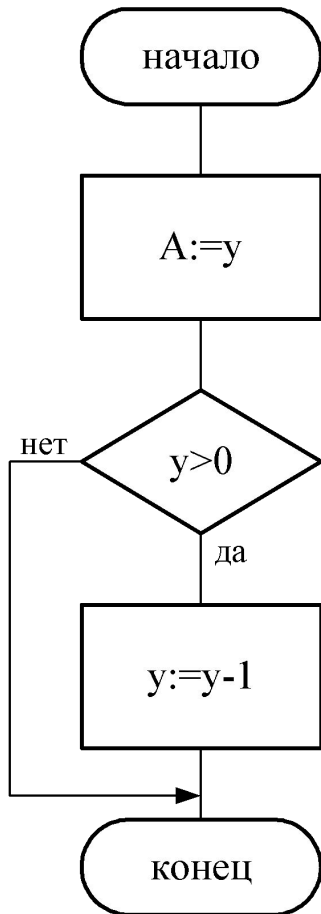
# Программирование на языке Ассемблер

Псевдокоманды в УЦВМ:

- 1) [**<имя>**] **start** **<адрес>** – первая запись программы,  
    <имя> – имя всей программы  
    <адрес> – адрес загрузки программы
- 2) [**<имя>**] **end** **<адрес>** – последний оператор программы,  
    <адрес> – пусковой адрес (точка входа в программу)
- 3) [**<метка>**] **word** **<операнд>** – резервирование места для  
    инициализированной переменной
- 4) [**<метка>**] **resb** **<операнд>** – резервирование определённого  
    операндом количества байтов памяти

# Программирование на языке Ассемблер

## Пример составления программы на Ассемблере



**; begin if y>0 then y:=y-1 end.**

**dec start 0 ; адрес загрузки программы = 0**

**lda y ; A=y**

**comp c0 ; (A-c0)>0?**

**jlt k ; если меньше переход на метку k**

**jeq k ; если равно переход на метку k**

**sub c1 ; A=A-1**

**k sta y; y=A**

**hlt ; останов**

**; данные**

**c1 word 1 ; поместить в слово по адресу c1 - 1**

**c0 word 0 ; поместить в слово по адресу c0 - 0**

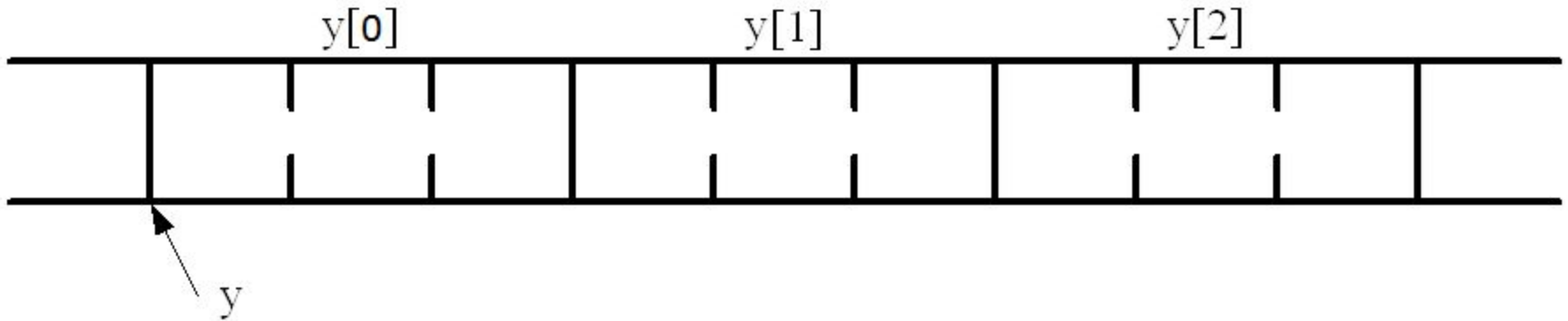
**y resb 3 ; выделить 3 байта под переменную y**

**end dec ; начать выполнение программы с**

**; метки dec**

# Программирование на языке Ассемблер

## Индексирование элементов массива



$y$  указывает на первый элемент массива.

Адрес элемента массива – адрес младшего байта элемента.

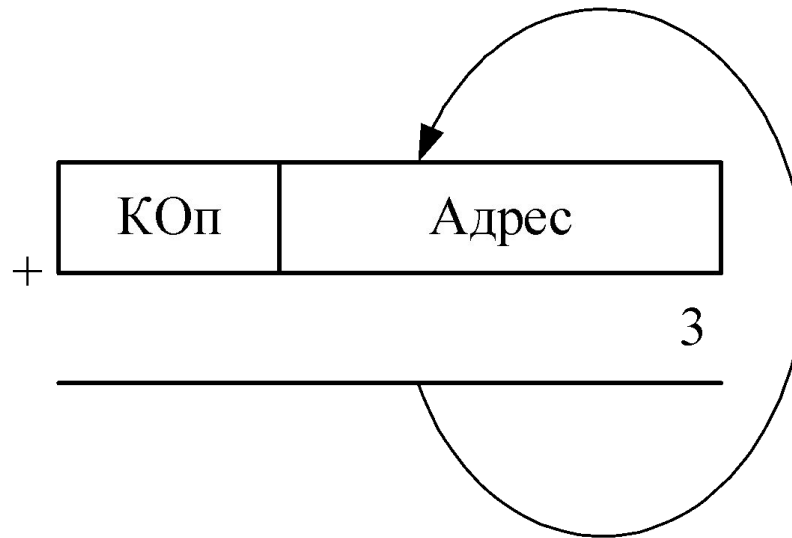
Для одномерного массива справедлива формула определения адреса заданного элемента:

$$E[i] = y + i * 3$$

# Программирование на языке Ассемблер

## Способы обращения к элементам массива

### 1. Способ модификации команд (нереентерабельные программы):



# Программирование на языке Ассемблер

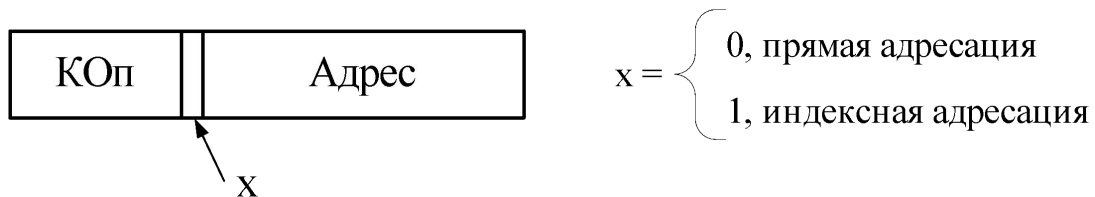
## Пример: Загрузка в аккумулятор 3-его элемента массива

H	Ex31		Ex31	start	0
T	000	00001C		lda	i
T	003	1C001F		sub	c1
T	006	200022		mul	c3
T	009	0C0025		sta	index
T	00C	000028		lda	com
T	00F	180025		add	index
T	012	0C0015		sta	read
T	015			read	resb 3
T	018	0C002B		sta	y
T	01B	FF		hlt	
			; данные		
T	01C	000003		i	word 3
T	01F	000001		c1	word 1
T	022	000003		c3	word 3
T	025			index	resb 3
T	028	00002E		com	lda d
T	02B			y	resb 3
T	02E	000020		d	word 32
T	031	000021			word 33
T	034	000022			word 34
T	037	000023			word 35
T	03A	000024			word 36
E	000			end	Ex31

# Программирование на языке Ассемблер

## Способы обращения к элементам массива

### 2. Использование индексного регистра:



### Вычисление исполнительного адреса:

$$ТА = \begin{cases} \text{Адрес, если } x=0 \\ \text{Адрес}+(X), \text{ если } x=1 \end{cases}$$

**При операциях с массивом в регистре  $X$  обычно хранится смещение адреса элемента массива относительно базового адреса этого массива.**

# Программирование на языке Ассемблер

Пример работы с одномерным массивом:

**ПОИСК МАКСИМАЛЬНОГО ЭЛЕМЕНТА МАССИВА**

```
program ExArray;  
var d: array[1..5] of integer;  
    max: integer;  
    i: integer;  
begin  
    max:=d[1];  
    for i:=2 to 5 do  
        if d[i]>max then max:=d[i];  
    end.
```

```
ExArray start 0  
    lda d  
    sta max  
    ldx c3  
rpt  lda d,x  
    comp max  
    jlt cont  
    jeq cont  
then sta max  
cont rmo x, a  
    add c3  
    rmo a, x  
    comp c15  
    jlt rpt  
    hlt
```

```
; данные  
c3   word 3  
c15  word 15  
max  resb 3  
d    resb 15  
    end ExArray
```



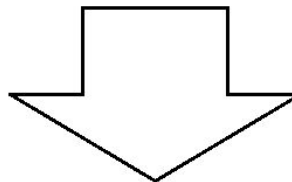
# Программирование на языке Ассемблер

## Многомерные массивы

При использовании в программе многомерных массивов производится линеаризация массива.

После линеаризации элементы многомерного массива располагаются в памяти друг за другом:

$Y_{00}$	$Y_{01}$	$Y_{0(n-1)}$
$Y_{10}$	$Y_{11}$	$Y_{1(n-1)}$
$Y_{(m-1)0}$	$Y_{(m-1)1}$	$Y_{(m-1)(n-1)}$



$$E[i,j] = y + i*3*l + j*3$$

$y$  – адрес начала массива  
 $l$  – кол-во элементов в строке  
 $i$  – номер строки  
 $j$  – номер столбца

	$Y_{00}$	$Y_{01}$	...	$Y_{0(n-1)}$	$Y_{01}$	$Y_{11}$	...	$Y_{1(n-1)}$	...	$Y_{(m-1)0}$	$Y_{(m-1)1}$	...	$Y_{(m-1)(n-1)}$	
--	----------	----------	-----	--------------	----------	----------	-----	--------------	-----	--------------	--------------	-----	------------------	--

# Программирование на языке Ассемблер

**Пример: Записать в массив n суммы элементов каждой строки матрицы m**

```
#include "stdafx.h"
int main()
{
    const int l = 4, h = 3;
    int m[h][l];
    int n[h];
    m[0][0] = 1;    m[0][1] = 2;    m[0][2] = 3;    m[0][3] = 4;
    m[1][0] = 5;    m[1][1] = 6;    m[1][2] = 7;    m[1][3] = 8;
    m[2][0] = 9;    m[2][1] = 0;    m[2][2] = 1;    m[2][3] = 2;
    for (int i = 0; i < h; i++)
        n[i] = 0;
    for (int i = 0; i < h; i++)
        for (int j = 0; j < l; j++)
            n[i] = n[i] + m[i][j];
    return 0;
}
```

# Программирование на языке Ассемблер

**Пример: Записать в массив n суммы элементов каждой строки матрицы m (данные)**

```

; Записать с массив n суммы элементов каждой строки
; матрицы m
H sumSt | sumSt start 0
; Данные
T 000 000004 | l1 word 4 ; кол-во элементов в строке
T 003 000003 | h word 3 ; кол-во элементов в столбце
T 006 000001 | m word 1
T 009 000002 | word 2
T 00C 000003 | word 3
T 00F 000004 | word 4
T 012 000005 | word 5
T 015 000006 | word 6
T 018 000007 | word 7
T 01B 000008 | word 8
T 01E 000009 | word 9
T 021 000000 | word 0
T 024 000001 | word 1
T 027 000002 | word 2
```

# Программирование на языке Ассемблер

**Пример: Записать в массив n суммы элементов каждой строки матрицы m (данные)**

T 02A		n	resb	9	
T 033		ln	resb	3	;длина массива n в байтах
T 036		lm	resb	3	;длина массива m в байтах
T 039		lmw	resb	3	;длина строки массива m в байтах
T 03C		in	resb	3	;смещение по n
T 03F		im	resb	3	;смещение по m
T 042		i	resb	3	;индекс текущего элемента в строке
T 045	000000		c0	word	0
T 048	000001		c1	word	1
T 04B	000003		c3	word	3

# Программирование на языке Ассемблер

**Пример: Записать в массив n суммы элементов каждой строки матрицы m (установка начальных значений)**

```
      ;Программа
T 04E 000003 | st      lda      h
T 051 20004B |         mul      c3
T 054 0C0033 |         sta      ln
T 057 040045 |         ldx      c0
T 05A AC10    |         rmo      x,a
T 05C 280033 | p2      comp     ln
T 05F 300072 |         jeq      p1
T 062 000045 |         lda      c0
T 065 0C802A |         sta      n,x
T 068 AC10    |         rmo      x,a
T 06A 18004B |         add      c3
T 06D AC01    |         rmo      a,x
T 06F 3C005C |         j        p2
T 072 000000 | p1      lda      l1
T 075 20004B |         mul      c3
T 078 0C0039 |         sta      lmw
T 07B 200003 |         mul      h
T 07E 0C0036 |         sta      lm
T 081 000045 |         lda      c0
T 084 0C0042 |         sta      i
T 087 0C003C |         sta      in
T 08A 0C003F |         sta      im
```

# Программирование на языке Ассемблер

**Пример: Записать в массив n суммы элементов каждой строки матрицы m**

```
T 08D 00003F | p5   lda   im
T 090 280036 |      comp  lm
T 093 3000D5 |      jeq   p4
T 096 000042 |      lda   i
T 099 280000 |      comp  l1
T 09C 3000C3 |      jeq   p3
T 09F 04003F |      ldx  im
T 0A2 008006 |      lda  m,x
T 0A5 04003C |      ldx  in
T 0A8 18802A |      add  n,x
T 0AB 0C802A |      sta  n,x
T 0AE 000042 |      lda  i
T 0B1 180048 |      add  c1
T 0B4 0C0042 |      sta  i
T 0B7 00003F |      lda  im
T 0BA 18004B |      add  c3
T 0BD 0C003F |      sta  im
T 0C0 3C008D |      j    p5
T 0C3 000045 | p3   lda  c0
T 0C6 0C0042 |      sta  i
T 0C9 00003C |      lda  in
T 0CC 18004B |      add  c3
T 0CF 0C003C |      sta  in
T 0D2 3C008D |      j    p5
```

```
T 0D5 FF      | p4   hlt
E 04E          |      end   st
```









# Программирование на языке Ассемблер

**Пример: Записать в массив n суммы элементов каждого столбца матрицы m**

```
#include "stdafx.h"
int main()
{
    const int l = 4, h = 3;
    int m[h][l];
    int n[l];
    m[0][0] = 1;    m[0][1] = 2;    m[0][2] = 3;    m[0][3] = 4;
    m[1][0] = 5;    m[1][1] = 6;    m[1][2] = 7;    m[1][3] = 8;
    m[2][0] = 9;    m[2][1] = 0;    m[2][2] = 1;    m[2][3] = 2;
    for (int j = 0; j < 4; j++)
        n[j] = 0;
    for (int i = 0; i < h; i++)
        for (int j = 0; j < l; j++)
            n[j] = n[j] + m[i][j];
    return 0;
}
```

# Программирование на языке Ассемблер

**Пример: Записать в массив *n* суммы элементов каждого столбца матрицы *m* (данные)**

H	sumSt		sumSt	start	0	
			; Данные			
T	000	000004		l1	word	4 ; кол-во элементов в строке
T	003	000003		h	word	3 ; кол-во элементов в столбце
T	006	000001		m	word	1 ; массив из 12 элементов
T	009	000002			word	2
T	00C	000003			word	3
T	00F	000004			word	4
T	012	000005			word	5
T	015	000006			word	6
T	018	000007			word	7
T	01B	000008			word	8
T	01E	000009			word	9
T	021	000000			word	0
T	024	000001			word	1
T	027	000002			word	2

# Программирование на языке Ассемблер

**Пример: Записать в массив n суммы элементов каждого столбца матрицы m (данные)**

T 02A		n	resb	12 ;;;;массив из 4 элементов
T 036		ln	resb	3 ;длина массива n в байтах
T 039		lm	resb	3 ;длина массива m в байтах
T 03C		lmw	resb	3 ;длина строки массива m в байтах
T 03F		in	resb	3 ;смещение по n
T 042		im	resb	3 ;смещение по m
T 045		i	resb	3 ;индекс текущего элемента в строке
T 048	000000		c0	word 0
T 04B	000001		c1	word 1
T 04E	000003		c3	word 3

T 02A		n	resb	9
T 033		ln	resb	3 ;длина массива n в байтах

		;Программа			
T 051	000000	st	lda l1 ;;;;	T 04E	000003   st lda h
T 054	20004E		mul c3	<b>Вычисление длины массива n</b>	
T 057	0C0036		sta ln	<b>//ln=l1*3</b>	
T 05A	040048		ldx c0	<b>Заполнение массива n нулями</b>	
T 05D	AC10		rmo x,a	<b>for (int j = 0; j &lt; 4; j++) n[j] = 0;</b>	
T 05F	280036	p2	comp ln	<b>j &lt; 4; //X-ln&gt;0</b>	
T 062	300075		jeq p1		
T 065	000048		lda c0	<b>n[j] = 0;</b>	
T 068	0C802A		sta n,x		
T 06B	AC10		rmo x,a		
T 06D	18004E		add c3	<b>j++; //X=X+3</b>	
T 070	AC01		rmo a,x		
T 072	3C005F		j p2		
T 075	000000	p1	lda l1	<b>Установка длины строки</b>	
T 078	20004E		mul c3	<b>массива t в байтах</b>	
T 07B	0C003C		sta lmw	<b>//lmw=l1*3</b>	
T 07E	200003		mul h	<b>Установка длины</b>	
T 081	0C0039		sta lm	<b>массива t в байтах</b>	
T 084	000048		lda c0	<b>//mw=lmw*h</b>	
T 087	0C0045		sta i	<b>//i=0</b>	
T 08A	0C003F		sta in	<b>//in=0</b>	
T 08D	0C0042		sta im	<b>//im=0</b>	

T 090	000042	p5	lda	im	//im=lm	for (int i = 0; i < h; i++)
T 093	280039		comp	lm	если =,	for (int j = 0; j < l; j++)
T 096	3000DB		jeq	p4	то выход	n[j] = n[j] + m[i][j];
T 099	000045		lda	i	//i=l1	
T 09C	280000		comp	l1	если конец строки, то переходим	
T 09F	3000CF		jeq	p3	к следующей в блоке p3	
T 0A2	040042		ldx	im	n[j] = n[j] + m[i][j]; //X=im	
T 0A5	008006		lda	m,x	//A=W[m+X]	
T 0A8	04003F		ldx	in	//X=in	
T 0AB	18802A		add	n,x	//A=A+W[n+X]	
T 0AE	0C802A		sta	n,x	//W[n+X]=A	
T 0B1	000045		lda	i	//i++	
T 0B4	18004B		add	c1		
T 0B7	0C0045		sta	i		
T 0BA	00003F		lda	in ; ; ; ;	//in++	T 0B4 0C0042   sta I
T 0BD	18004E		add	c3 ; ; ; ;		T 0B7 00003F   lda im
T 0C0	0C003F		sta	in ; ; ; ;		
T 0C3	000042		lda	im	//im=im+3	
T 0C6	18004E		add	c3		
T 0C9	0C0042		sta	im		T 0C6 0C0042   sta i
T 0CC	3C0090		j	p5		T 0C9 00003C   lda in
T 0CF	000048	p3	lda	c0		T 0CC 18004B   add c3
T 0D2	0C0045		sta	i	//i=0	T 0CF 0C003C   sta in
T 0D5	0C003F		sta	in ; ; ; ;	//in=0	T 0D2 3C008D   j p5
T 0D8	3C0090		j	p5		

# Программирование на языке Ассемблер

**Пример: Записать в массив n суммы элементов  
каждого столбца матрицы m**

```
T 0DB FF      | p4 hlt
                |
E 051         |      end st
```





