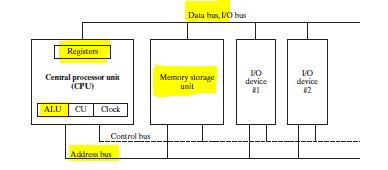
**Computer Organization and Assembly Language**

|  |  |
| --- | --- |
| **Lab 04** | |
| **Topic** | Memory addressing modes, Data movement, Array, addition and subtraction instructions |
| **Objective** | To learn iteration in an array and data movement instructions like mov, add and sub. |
| **Material** | Document on Arrays is uploaded for helping material. |

**Introduction**

* The *control unit* (CU) coordinates the sequencing of steps involved in executing machine instructions.
* The *arithmetic logic unit* (ALU) performs arithmetic operations such as addition and subtraction and logical operations such as AND, OR, and NOT.

The CPU is attached to the rest of the computer via pins attached to the CPU socket in the computer’s motherboard. Most pins connect to the data bus, the control bus, and the address bus. The *memory storage unit* is where instructions and data are held while a computer program is running. The storage unit receives requests for data from the CPU, transfers data from random access memory (RAM) to the CPU, and transfers data from the CPU into memory. All processing of data takes place within the CPU, so programs residing in memory must be copied into the CPU before they can execute. Individual program instructions can be copied into the CPU one at a time, or groups of instructions can be copied together.

A *bus* is a group of parallel wires that transfer data from one part of the computer to another. A computer system usually contains four bus types: data, I/O, control, and address. The *data bus* transfers instructions and data between the CPU and memory. The I/O bus transfers data between the CPU and the system input/output devices. The *control bus* uses binary signals to synchronize actions of all devices attached to the system bus. The *address bus* holds the addresses of instructions and data when the currently executing instruction transfers data between the CPU and memory.

Important Notes:

* ArrayName db 1,2,3,4,5,6,7,8,9,10; *This will reserve 10 byte sized values in consecutive memory locations.*
* ArrayName dw 1,2,3,4,5,6,7,8,9,10; *This will reserve 10 word sized values in consecutive memory locations.*
* Array1 db 100 dup(0); *Defines an array of 100 elements with zero as initial values.*

PART 1

**Observe the register values and memory contents in each code snippet.**

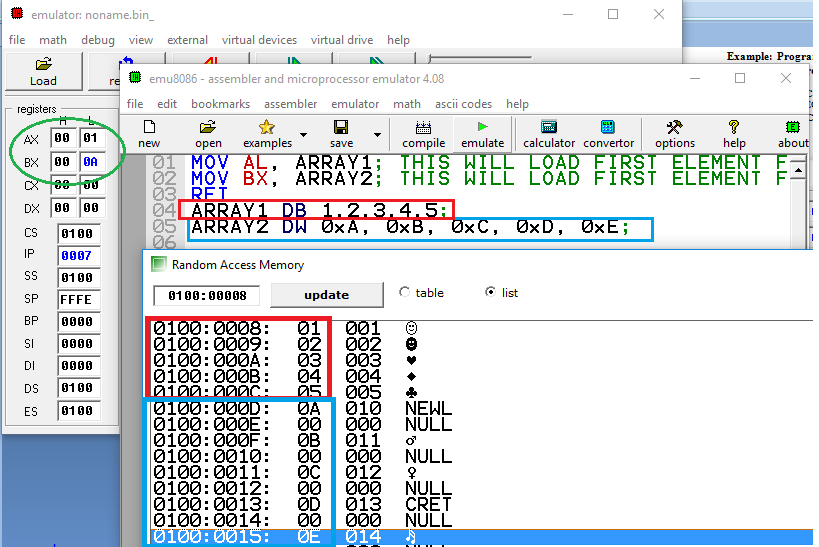
Sample Code 1: (Declaration and Access from arrays….)

MOV AL, ARRAY1; *THIS WILL LOAD FIRST ELEMENT FROM ARRAY1*

MOV BX, ARRAY2; *THIS WILL LOAD FIRST ELEMENT FROM ARRAY2*

RET

ARRAY1 DB 1,2,3,4,5;

ARRAY2 DW 0xA, 0xB, 0xC, 0xD, 0xE;

Sample Code 2: (Iteration in an array….)

**TO ACCESS NEXT ELEMENTS WITHIN AN ARRAY ADD OFFSETS TO ARRAY NAME DEPENDING UPON ARRAY TYPE.**

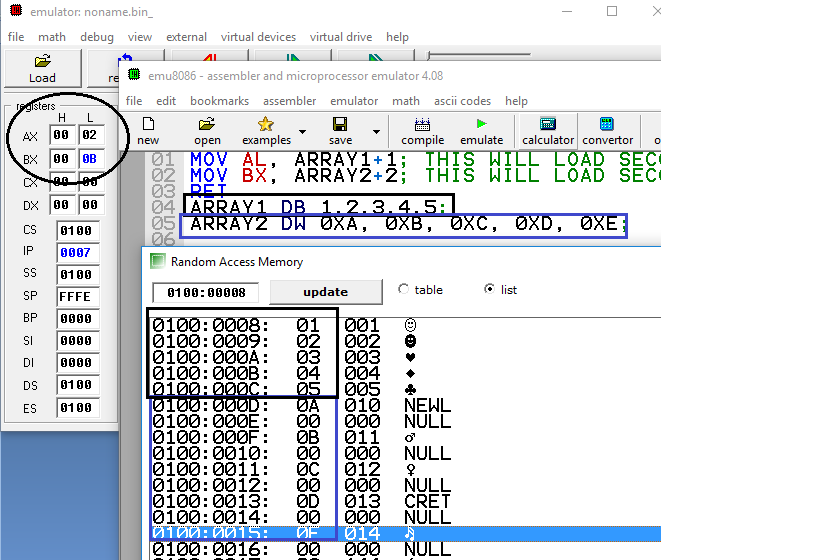
MOV AL, ARRAY1+1; *THIS WILL LOAD SECOND ELEMENT FROM ARRAY1*

MOV BX, ARRAY2+2; *THIS WILL LOAD SECOND ELEMENT FROM ARRAY2*

RET

ARRAY1 DB 1,2,3,4,5;

ARRAY2 DW 0XA, 0XB, 0XC, 0XD, 0XE;



Sample Code 3: (Accessing values with Iteration….)

**Example: Program below loads the values from VEC1 and VEC2, adds them and places their result in VEC3.**

LEA SI, VEC1 ;Loads the effective address of array VEC1 in SI

LEA BX, VEC2 ;Loads the effective address of array VEC2 in SBX

LEA DI, VEC3 ;Loads the effective address of array VEC3 in DI

MOV CX, 4

L1:

MOV AL, [SI]

ADD AL, [BX]

MOV [DI], AL

INC SI; ;adds ‘1’ to the destination operand.

INC BX;

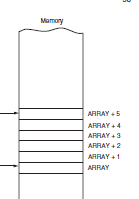
INC DI;

DEC CX; Subtracts ‘1’ from the destination operand.

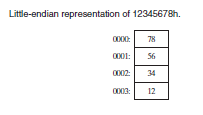
JNZ L1; Repeats executing instructions from label L1 until CX becomes zero.

RET

VEC1 DB 1, 2, 5, 6

VEC2 DB 3, 5, 6, 1

VEC3 DB ?, ?, ?, ?

****

**Higher Address**

**Lower Address**

**Use of Loop Instruction:**

It performs the following tasks.

* Decrement CX by 1 i.e., CX=CX-1
* Check if CX≠0, Repeat executing instructions from specified label

For example, if you use instruction **loop L1**, it would do following operations.

* DEC CX;
* CMP CX, 0; *It sets the Zero FLAG if both values are same.*
* JNZ L1; *This jump is taken if the last arithmetic operation did not produce a zero.*

Sample Code 4: (Accessing values with Iteration….)

**Example: Program below loads the values from VEC1 and VEC2, adds them and places their result in VEC3.**

LEA SI, VEC1

LEA BX, VEC2

LEA DI, VEC3

MOV CX, 4

**L1:**

MOV AL, [SI]

ADD AL, [BX]

MOV [DI], AL

INC SI; ; adds ‘1’ to the destination operand.

INC BX;

INC DI;

**LOOP L1;**

RET

VEC1 DB 1, 2, 5, 6

VEC2 DB 3, 5, 6, 1

VEC3 DB ?, ?, ?, ?

**PART 2**

**Problem #1:**

Multiplication can be performed by using the addition operation. For instance; If AL register has 3 value and AH register has 4 value. Then AL x AH means 3 x 4. This can be done by adding 3 four time i.e 3 + 3 + 3+3 = 12.

Write an assembly language program that multiplies the value of AL register with AH register and store the result in BX register where AL=8 and AH=5 in decimals.

Ans:

mov al,8

mov ah,5

mov ch,ah

mov ah,al

L1:

add al,ah

mov bh,al

dec ch

cmp ch,1

jnz L1

ret

**Problem #2:**

Declare an array of BYTE type with values:-

Array db 0xAB,0xCD,16h,61h,17h,64h,30h

Add all the values stored in the array and store the sum in a WORD type variable named “sum”.

Ans:

lea si,Array

lea di,sum

mov cx,7

mov al,0

L1:

mov al,[si]

add sum,ax

inc si

dec cx

cmp cx,0

jnz L1

ret

Array db 0xAB,0xCD,16h,61h,17h,64h,30h

sum dw 0

**Problem # 3:**

Convert the following high level language program to its equivalent assembly language program.

Int arr[] = {9,8,6,34,17,25,59,100};

Int sum = 0;

sum=sum+ arr[0]+8;

sum =sum+arr[1]+7;

sum= sum + arr[2] + 6 ;

sum= sum + arr[3]+ 8 ;

sum= sum + arr[4] + 4 ;

sum= sum + arr[5] + 9 ;

sum= sum + arr[6] + 8 ;

sum= sum + arr[7] +7 ;

Ans:

jmp start

Arr1 dw 9,8,6,34,17,25,59,100

sum dw 0

start:

lea si,Arr1

lea di,sum

mov bx,[si]

add sum,bx

add sum,8

mov bx,[si+2]

add sum,bx

add sum,7

mov bx,[si+4]

add sum,bx

add sum,6

mov bx,[si+6]

add sum,bx

add sum,8

mov bx,[si+8]

add sum,bx

add sum,4

mov bx,[si+10]

add sum,bx

add sum,9

mov bx,[si+12]

add sum,bx

add sum,8

mov bx,[si+14]

add sum,bx

add sum,7

***NOTE: Assembly language do not support memory to memory mov,add,sub instructions***

**Problem #4:**

Consider the following arrays and write an assembly language program to perform the operations in a loop.

Odd index of third array should be the mapping of the odd index of first array and even index of third array should be the mapping of the even index of second array.

Let Array 1 (byte size array) :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Value | 05 | EF | 36 | 3D | 38 | 0D | 84 | 15 | 16 |

Let Array 2 (byte size array):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Value | 0C | 64 | 9A | 55 | AA | 12 | 87 | 24 | 35 |

Calculate Array 3(byte size array) such that:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Value | 0C | EF | 9A | 3D | AA | 0D | 87 | 15 | 35 |

After merging, find the sum of all elements of third array in a memory variable of size word.

***NOTE: Assembly language do not support memory to memory mov,add,sub instructions***

**Problem #5:**

Consider the following arrays and write an assembly language program to perform the following operation in a loop.

Values of third array should be equal to the sum of elements of array1 and array2.

First and Last values are solved for you.

Let Array 1 (byte size array) :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Value | 05 | EF | DE | 64 | 38 | 0D | 84 | 15 |

Let Array 2 (byte size array):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Value | 0C | 64 | 9A | 55 | AA | 12 | 87 | 24 |

Let Array 3(word size array):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Value | 11 | ? | ? | ? | ? | ? | ? | 39 |

After merging find the sum of all elements of third array in a memory variable of size word.

***NOTE: Assembly language do not support memory to memory mov,add,sub instructions***

**Problem # 6:**

Consider the following arrays and write an assembly language program to perform the operations in a loop. Use **Direct addressing mode only** for referencing.

Array3[0]=Array1[0]+Array2[10]

Array3[1]=Array1[1]+Array2[9]

Array3[2]=Array1[2]+Array2[8]

:::::::::::::::::::::::::::::::::::::::::::::

Array3[10]=Array1[10]+Array2[0]

First and Last values are solved for you.

Let Array 1 (byte size array) :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Value | 05 | EF | DE | 64 | 38 | 0D | 84 | 15 | 88 |

Let Array 2 (byte size array):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Value | 0C | 64 | 9A | 55 | CC | 13 | 87 | 24 | 0F |

Let Array 3(word size array):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Value | 14 | ? | ? | ? | ? | ? | ? | ? | 94 |

***NOTE: Assembly language do not support memory to memory mov,add,sub instructions***

**Problem #7:**

Write an assembly language program that combines two lists by alternatively taking elements by using a loop? Use **register indirect addressing mode only** for referencing.

Let:

Array1 [A,B,C,D,E]

Array2 [0,1,2,3,4,5,6,7,8,9]

Array3 [A,0,1,B,2,3,C,4,5,D,6,7,E,8,9].

**Problem #8:**

Write an assembly language program that reverses the whole array of numbers. Use temporary variable to perform the desired operation.

Let Array 1 (byte size):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Value | 5A | 6B | 26 | 8D | 35 | 0F | 91 | 34 | 73 | 64 |

After reverse Array 1 (byte size) should be:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Value | 64 | 73 | 34 | 91 | 0F | 35 | 8D | 26 | 6B | 5A |

***NOTE: Assembly language do not support memory to memory mov,add,sub instructions***