Lab 3

Addressing Modes

Objectives

- Students will learn basic addressing modes
- Students will come to know about little-endian and big-endian notations
- Students will be able to access any physical location of memory for data

When we run a program, the operating system locates the complete program on disk and loads (copies) it to the RAM. It also initializes the value of the CS and SS registers with the starting addresses of the code and stack segments. However, it does not initialize the DS segment. The DS value (and ES if used) must be initialized by the program to access memory for data. This is done as follows:

```
MOV AX, @DATA
MOV DS, AX
```

Here, @DATA refers to the start of the data segment and is replaced by a number. Since we cannot assign a number directly to segment registers, therefore, we must first assign it to a general-purpose register and then from that general-purpose register to a segment register.

Addressing Modes

Immediate addressing mode

In the immediate addressing mode, the source operand is a constant. In immediate addressing mode, as the name implies, when the instruction is assembled, the operand comes immediately after the opcode. For this reason, this addressing mode executes quickly. However, in programming, it has limited use. Immediate addressing mode can be used to load information into general-purpose registers.

Examples:

```
MOV AX,2550H
MOV CX,625
MOV BL,40H
```

Register addressing mode

The register addressing mode involves the use of registers to hold the data to be manipulated. Memory is not accessed when this addressing mode is executed; therefore, it is relatively fast.

Examples:

```
ADD BX, DX
MOV ES, AX
MOV AL, BH
```

Direct Addressing mode

In the direct addressing mode, the data is in some memory location(s) and the address of the data in memory comes immediately after the instruction. Note that in immediate addressing, the operand itself is provided with the instruction, whereas in direct addressing mode, the address of the operand is provided with the instruction. This address is an offset.

```
MOV DL, [2400h]        ; move contents of DS:2400H into DL
```

The physical address is calculated by DS x 10h + 2400h.

Register indirect addressing mode

In the register indirect addressing mode, the address of the memory location where the operand resides is held by a register. The registers used for this purpose are SI, DI, and BX.

For example:

```
MOV AL, [BX]        ;moves the contents of the memory location into AL, the
                    memory location pointed to  DS:BX.
```

- The physical address is calculated by DS x 10h + BX. The same rules apply when using register SI or DI.
- BP register can also be used as a pointer register. However, the physical address will be calculated by SS x 10h + BP.
- AX, CX, DX, and SP cannot be used as pointer registers/ offset.

We can override the segment register while using BX, SI, DI, and BP registers as pointers by writing the preferred segment register name with it as shown below.

```
MOV AL, ES:[BX]
MOV AL, DS:[BP]
MOV AL, CS:[SI]
MOV AL, SS:[DI]
```

The size of a register in an instruction specifies how many bytes will be read or written from or to memory. If a register is not there, one byte is accessed from memory by default in Emu8086. However, to avoid confusion, one must specify the number of bytes to read or write to memory using the following:

- **byte ptr** - for byte.
- **word ptr** - for word (two bytes).

For example:

MOV byte ptr [2400h], 1

ADD word ptr [2400h], 2

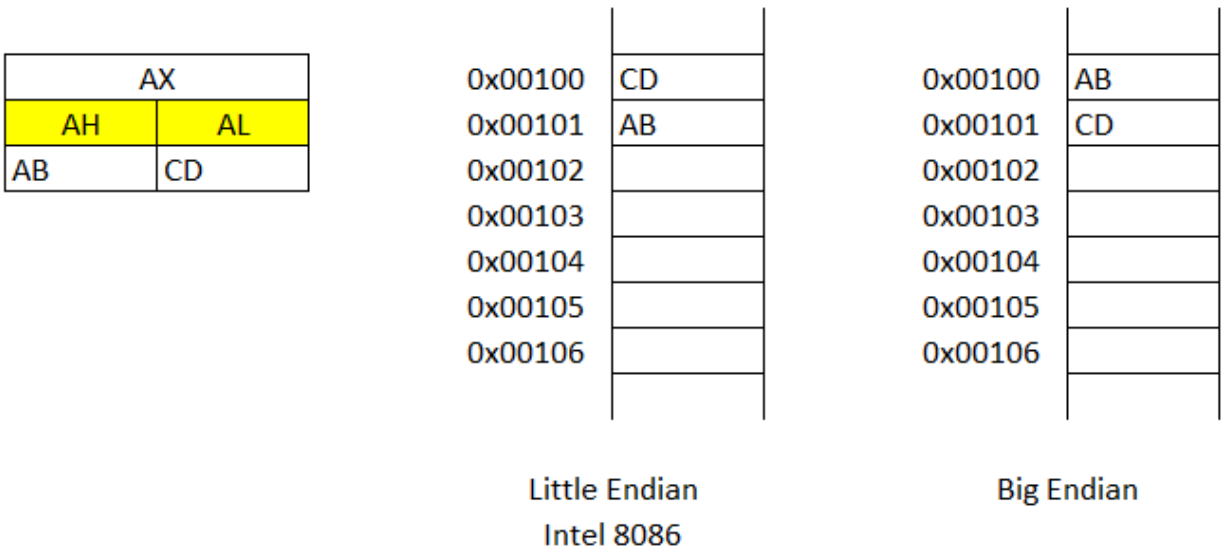
Storing multi-byte data in RAM

Little-endian and big-endian are the two ways of storing data in memory. In little-endian, the lower byte is stored at a lower address and the higher byte is stored at a higher address. However, in big-endian, the lower byte is stored at the higher address and the higher byte is stored at the lower address.

Let the AX register contain '0xABCD' and the DS register contain "0000h". The following instruction will write the contents of the AX register into the memory and start writing from physical location 0x00100 onwards. Since the Intel 8086 processor is based on little-endian, it will store the lower byte 'CD' at 0x00100 and higher byte 'AB' at 0x00101 as shown in the following figure. However, the processor that is based on big-endian will store it the other way around. Memory will be read in the same way.

Instruction:

MOV [0100h], AX



Accessing a specific physical location of RAM

Using the concepts above, one can access any physical location of memory. The physical address is a 20-bit value that has to be converted into Segment: Offset in a way such that when the processor combines it to form a physical address, it should be in the same location.

To write a byte “0x12” to the physical address 0xABCDE of RAM, one needs to break the physical address into Segment and Offset parts each of 16-bit. One combination is:

Segment : Offset
0ABCDh : 000Eh.

```
.code

   mov ax,0abcdh
   mov ds,ax
   mov bx,000eh
   mov byte ptr [bx],012h
```

Similarly, to write a word with the physical address “0xABCDE”, the prefix: byte ptr will be replaced with word ptr.

```
   mov ax,0abcdh
   mov ds,ax
   mov bx,000eh
   mov word ptr [bx],012h
```

Program to write 0x1234 physical memory "0xABCDE".

```
.model small

.data

.code

mov bx,0abcdh
mov es,bx
mov word ptr es:[000eh],01234h

.exit
```

The description of the above program is shown in the following table.

Instruction	Description
mov bx,0abcdh	Moving segment address to bx before moving it to ES
mov es,bx	Moving segment address to ES
mov word ptr es:[000eh]	Writing word "0x1234" to physical memory 0xABCDE

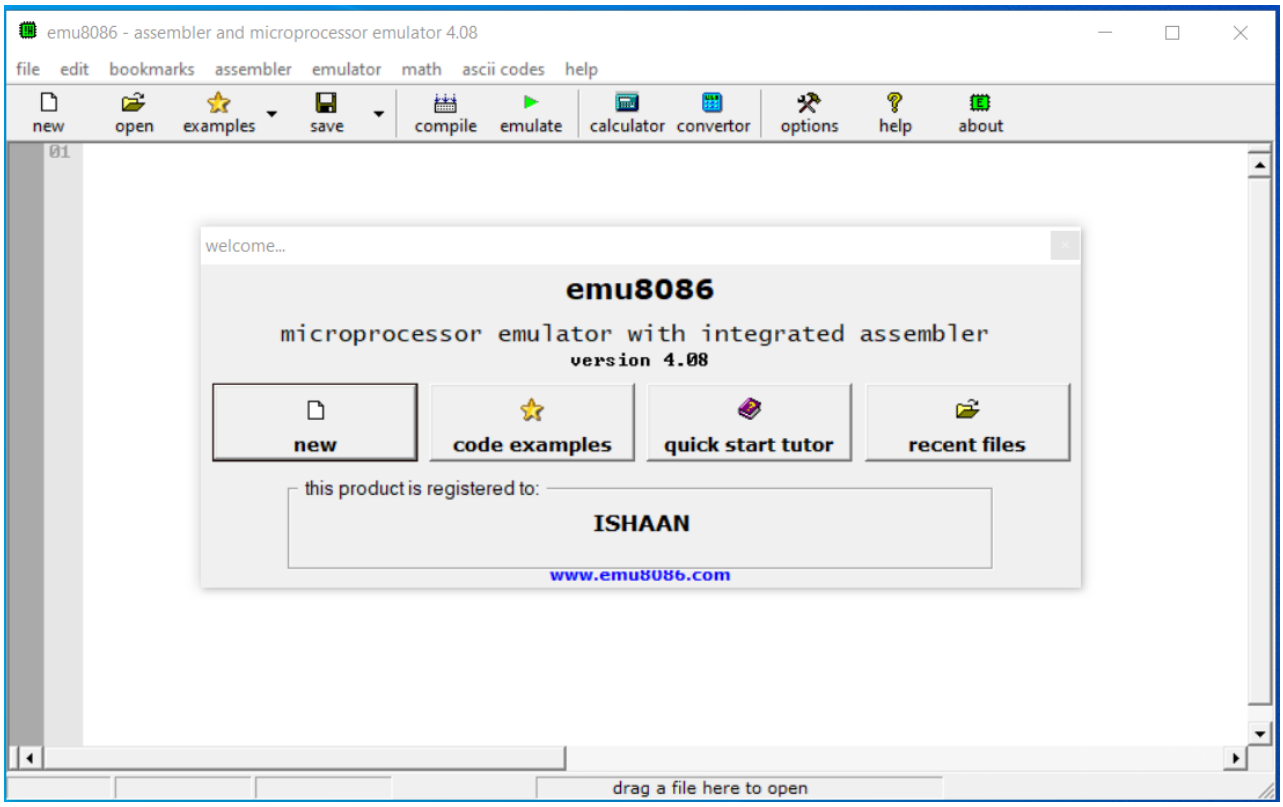
Emu8086 Tutorial Step by Step

Step-1

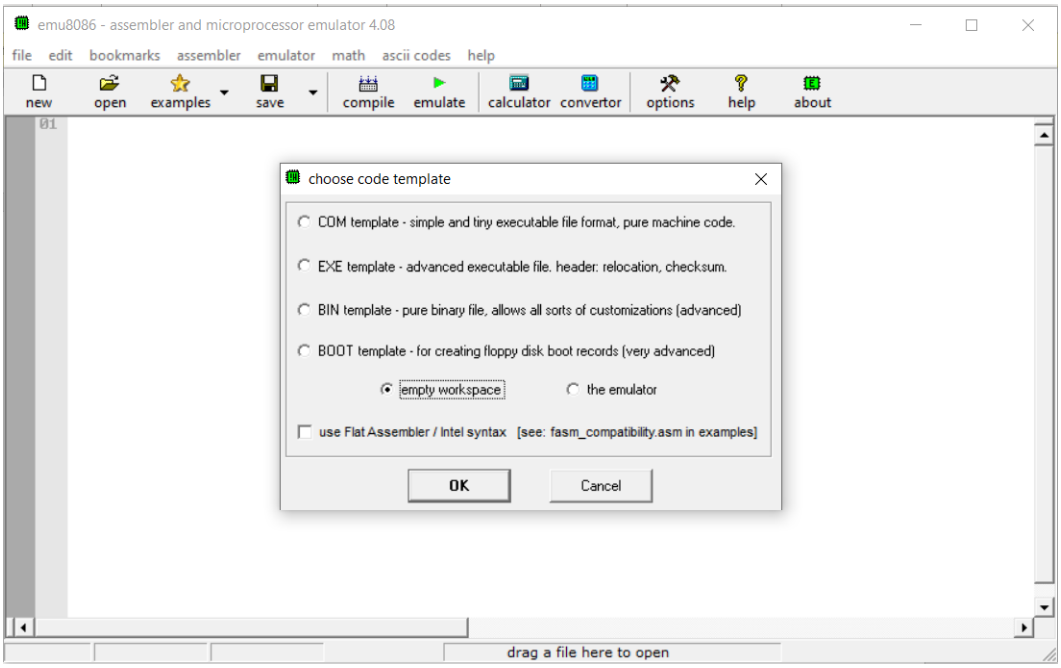
Double-click on the icon on the desktop 

Step-2

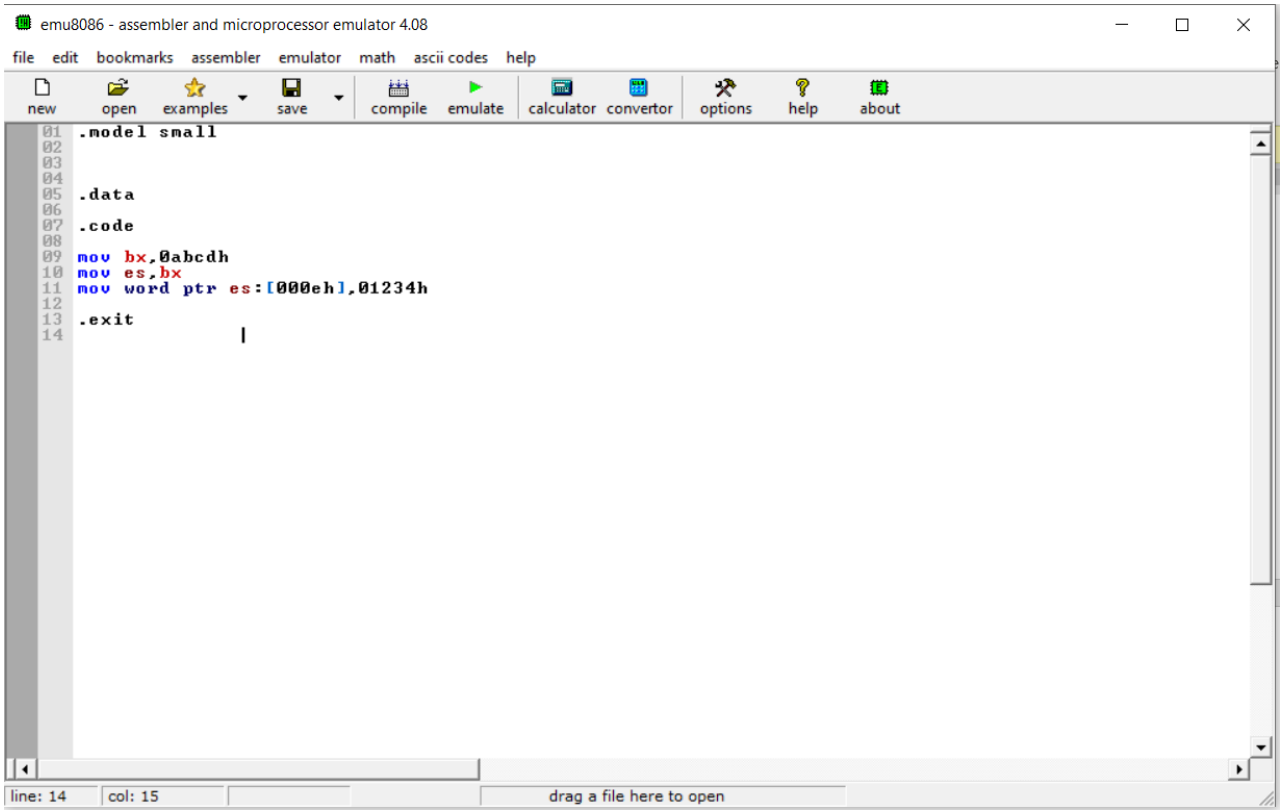
The following window will appear. Click on new.



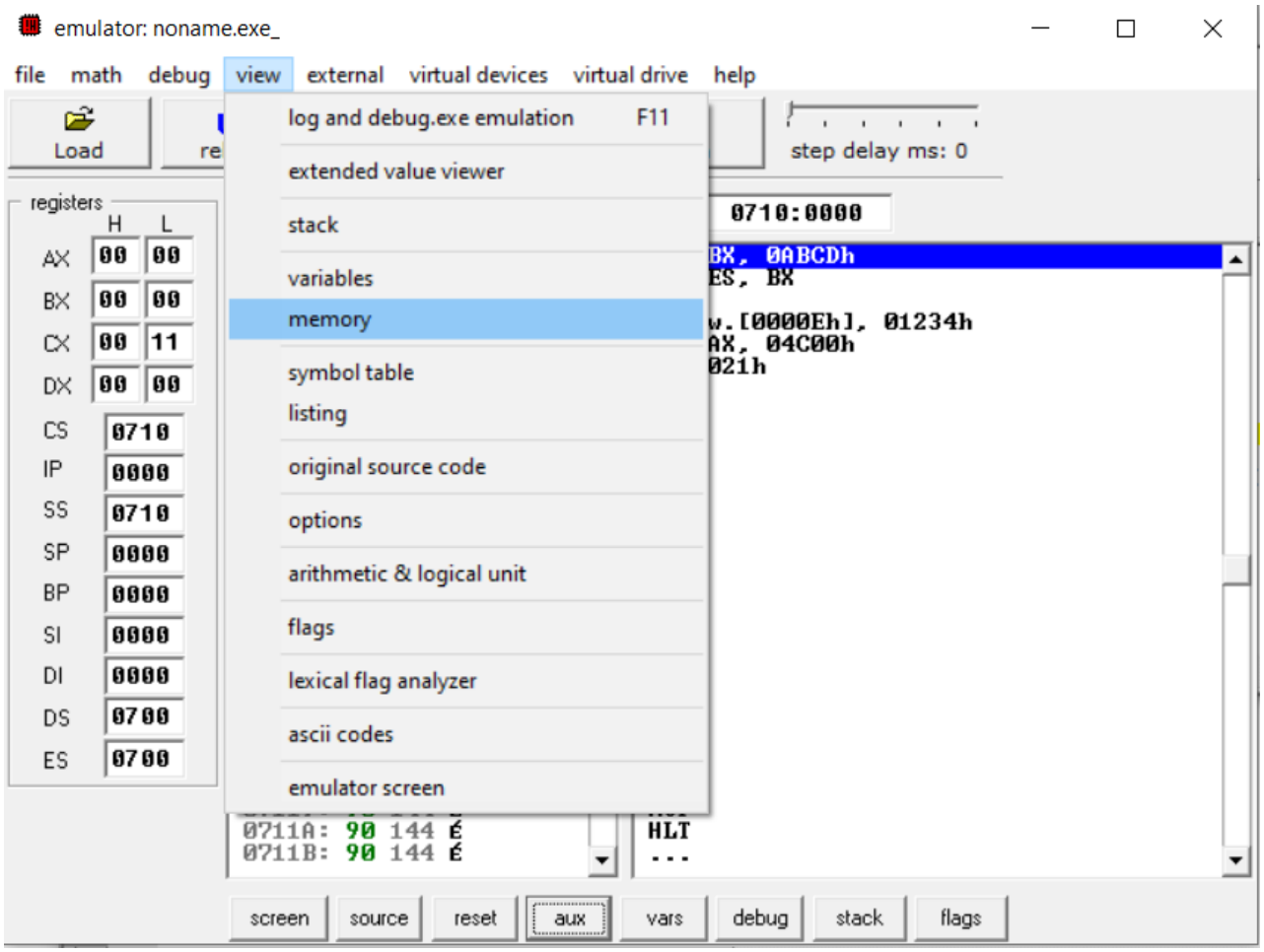
Step-3
Click on the empty workspace and press OK.



Step-4
Type the code given above and click on emulate.

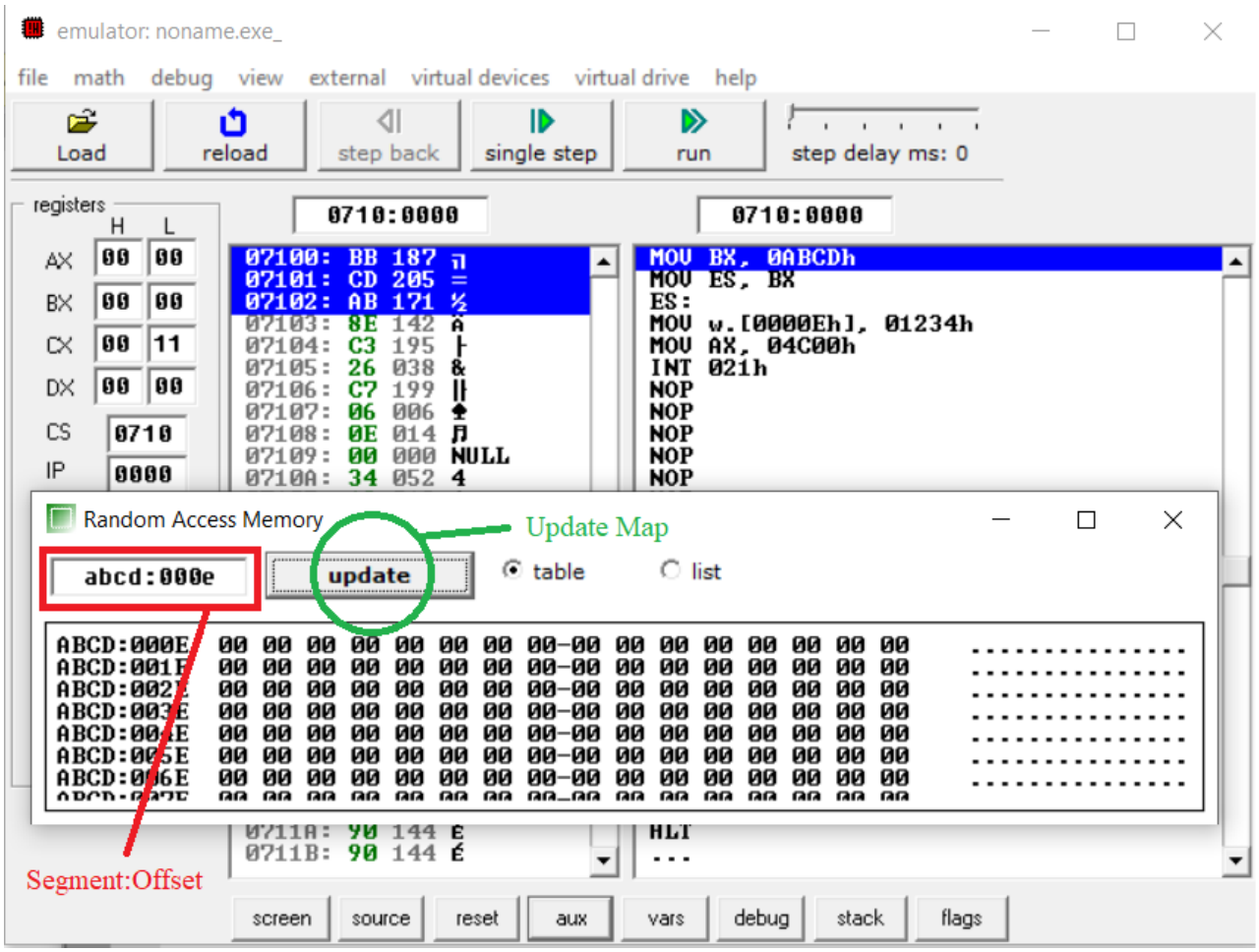


Step-5
Click on “Memory” from the view menu!

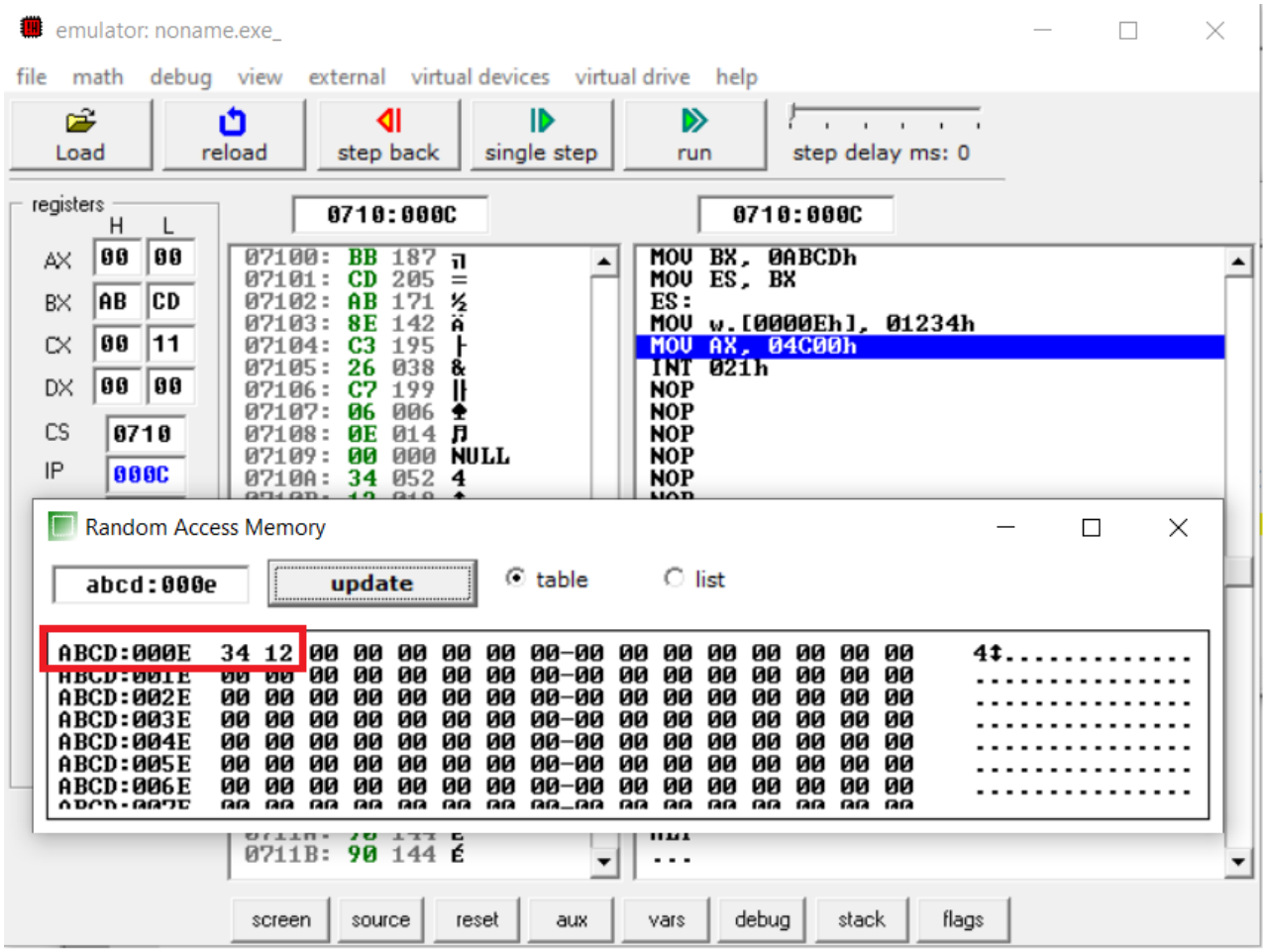


Step-6

Write the logical address of the desired part of the segment that you want to view.



Step-7
Keep clicking on "Single step" to execute program instructions one by one.
(The mov word ptr es:[000eh],01234h instruction will write two bytes to memory.)



Observation:

- Which byte is written to what address?

Practice Exercise

Task-1

Write a program that stores the following numbers into the current data segment at offset: 0x1000 onwards. The program then calculates the sum of these numbers and stores it at the physical location: 0xCD1F3.

Numbers: 0x1F00, 0xA0B1, 0x1254, 0x34EF, 0x8700

Task-2

The "HLT" has a machine code of "0xF4". Write a code to replace the first instruction of your program with this instruction.