



Microprocessor and Assembly Language Lab

Lab Material 3 for CSE 312 (M&AL Lab)

Dr. Shah Murtaza Rashid Al Masud

Associate Prof.

Dept. of CSE, UAP

Microprocessor Based Systems

**How to see the (Physical) Address, OP CODE,
operand and Memory**

BUS INTERFACE UNIT (BU)

The BIU performs all bus operations for EU .

Fetching instructions

Responsible for executing all external

bus cycles.

Read operands and write result.

EXECUTION UNIT (EU)

Execution unit contains the complete infrastructure required to execute an instruction

BCD

H = bit ?? 4 bit 1111H:0000H = PA?? (20 bit long)

1111H*10H+0000H=11110H+0000H=11110H=20 bit long

1000H*10H=10000H

BUS INTERFACE UNIT (BU)

BCD

H = bit ?? 4 bit 1111H:0000H = PA?? (20 bit long)

1111H*10H+0000H=11110H+0000H=11110H=20 bit long

1000H*10H=10000H

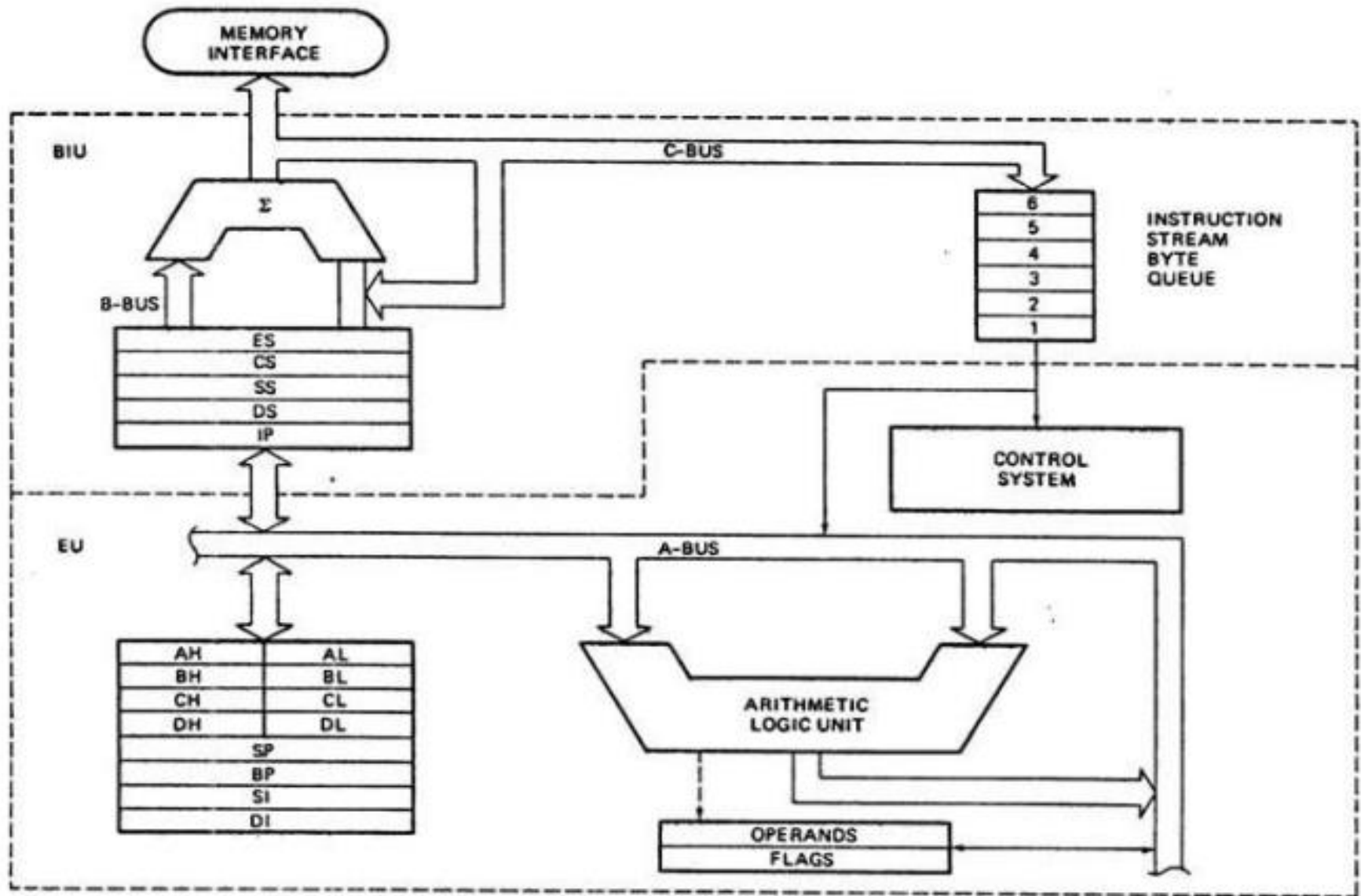
10h*10h=100h

20h*10h=200h

0100:0000

0100*10+0000=01000+0000=01000H(5*4=20bit long PA)

Internal Block Diagram



- CS-Code Segment Register points the code segment (**segment address**) in the memory
- IP-Instruction Pointer register points to the offset address of a code segment in a memory

IN 8086emu

DEFAULT

CODE SEGMENT

Offset Address

Segment Address

IP register

eg. 00H

CS register

eg. 100H

Physical Address = Segment Add. x 10H + Offset Add.

- If we know the segment address and offset address of a particular instruction then we can calculate the actual physical address of a location in a memory.

IN 8086emu

DEFAULT

CODE SEGMENT

Offset Address

IP register
eg. 00H

Segment Address

CS register
eg. 100H

Physical Address = Segment Add. x 10H + Offset Add.

- If we know the segment address and offset address of a particular instruction then we can calculate the actual physical address of a location in a memory.

```

01 MOV AX, 10H
02 MOV BX, 05H
03
04 ADD AX, BX

```

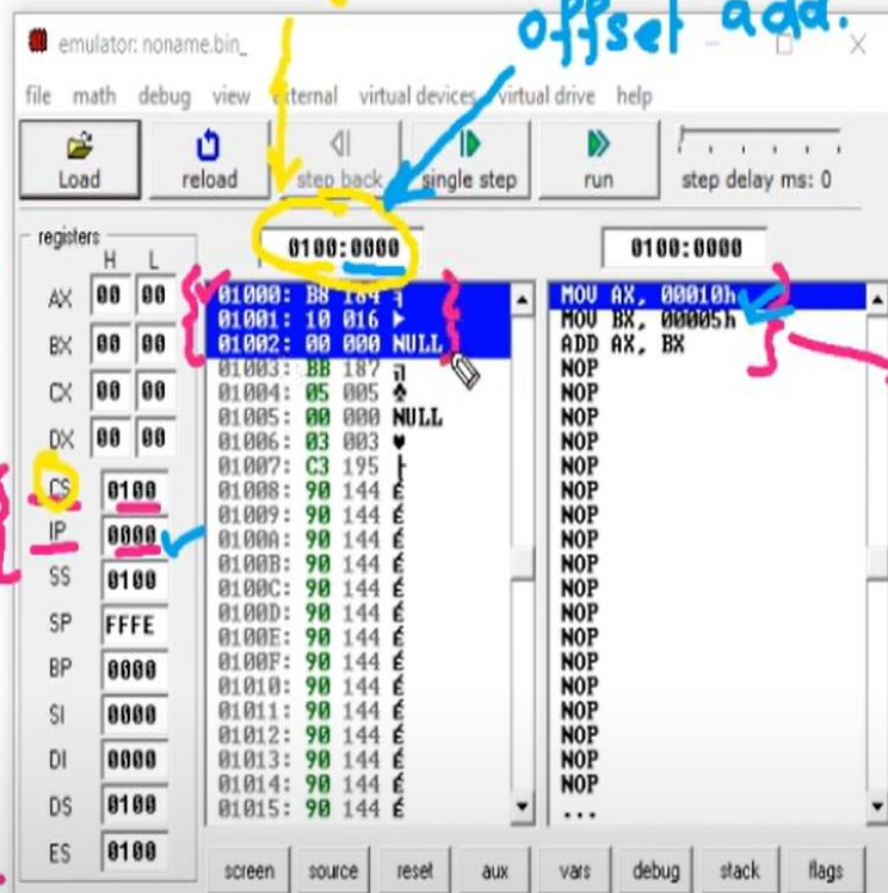
Seg Add.
off Add.

PA = Seg Add x 10H + off.add.

= 100H x 10H + 0H

PA = 1000H

Registers



- After press on the single step

```
01 MOV AX, 10H
02 MOV BX, 05H
03
04 ADD AX, BX
```

Press **ESC** to exit full screen

$$PA = 100H \times 10H + 3H \\ = 1003H$$

100
3 →
→

emulator: noname.bin_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

	H	L
AX	00	10
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	0003	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

0100:0003

01000: B8	184	1
01001: 10	016	1
01002: 00	000	NULL
01003: BB	187	1
01004: 05	005	1
01005: 00	000	NULL
01006: 03	003	1
01007: C3	195	1
01008: 90	144	E
01009: 90	144	E
0100A: 90	144	E
0100B: 90	144	E
0100C: 90	144	E
0100D: 90	144	E
0100E: 90	144	E
0100F: 90	144	E
01010: 90	144	E
01011: 90	144	E
01012: 90	144	E
01013: 90	144	E
01014: 90	144	E
01015: 90	144	E

0100:0003

```
MOV AX, 00010h
MOV BX, 00005h
ADD AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

screen source reset aux vars debug stack flags

- ```
03
04 ADD AX, BX
```

$$PA = 100H \times 10H + 6H$$

$$= \underline{\underline{1006H}}$$

100 →  
6 →



**OPCODE:** every instruction is converted into opcode. Opcode is the language that our MP speaks internally. We write in assembly language and our MP change it into OPCODE. **MOV AX, 0005H**

## OPCODE

What programmers write ?



Assembly Language

eg. MOV AX,BX      'Operand'



Gets converted into 'Opcode'

'Operand'

eg. MOV AX,1005H



Gets converted into 'Opcode'

In our 8086emu , the opcode is in the form of hexadecimal code.

**Opcode** or operation code is the command where the task is actually defined.

The **operand** is the 2nd part which is the data to be operated on. **MOV AX, 0005H**

## OPCODE

What programmers write ?



Assembly Language

eg. MOV AX,BX

Operand



Gets converted into 'Opcode'

eg. MOV AX,1005H

Operand



Gets converted into 'Opcode'

In our 8086emu , the opcode is in the form of hexadecimal code.



OPCODE: every instruction is converted into opcode

[illegible]

# OPCODE: every instruction is converted into opcode

ADD AX, BX

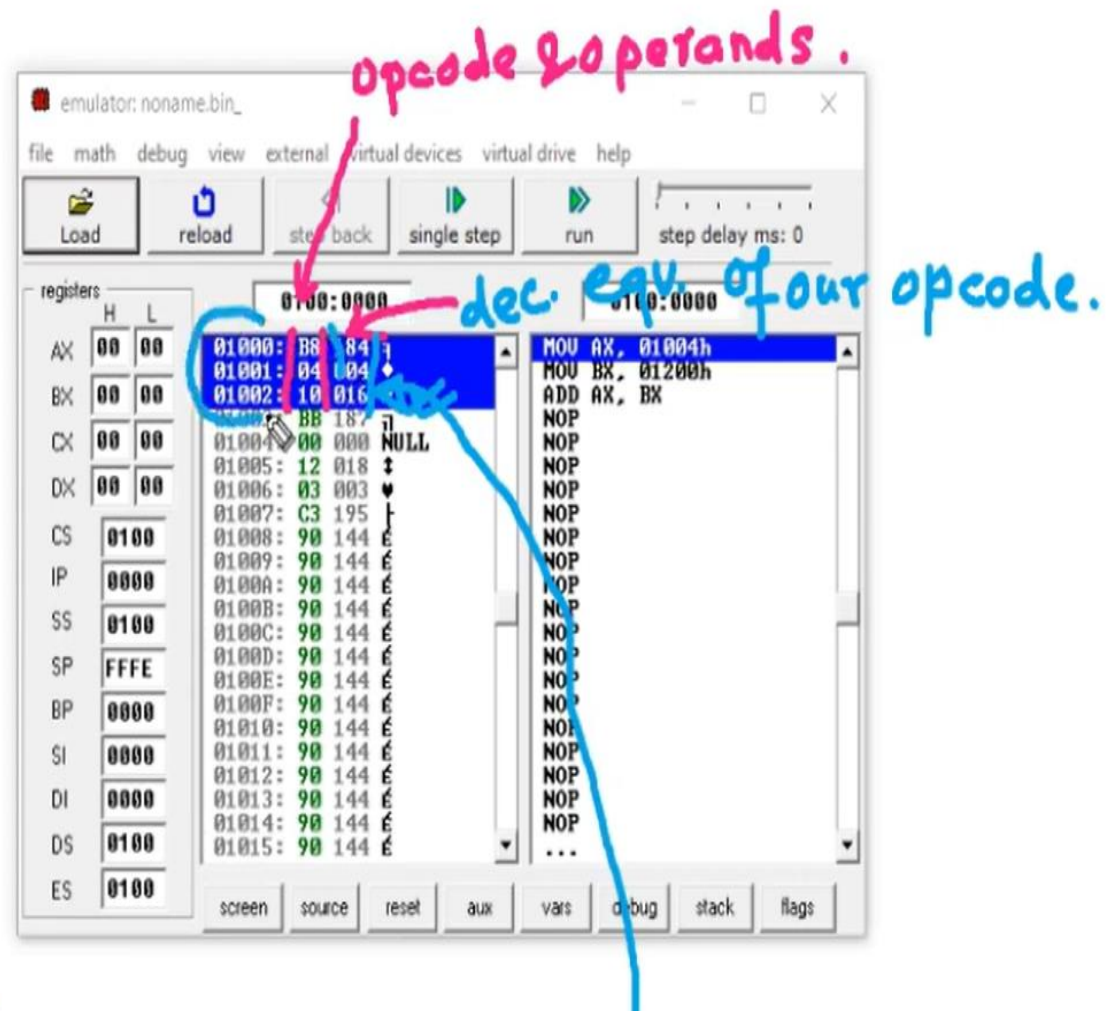
MOV AX, 1004H

opcode

B8  
04  
10

11  
8  
B8  
decimal  
16x8  
8

04  
04



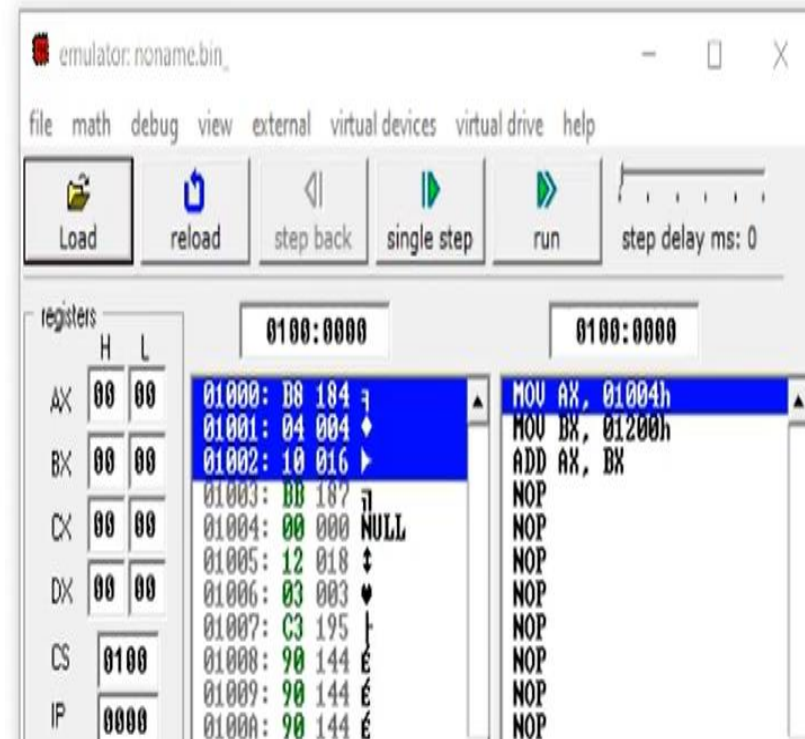


# OPCODE: every instruction is converted into opcode

```
01 MOV AX, 1004H
02 MOV BX, 1200H
03
04 ADD AX, BX
```

Summary

| Inst/<br>Mnemonics | Add.  | Opcode |
|--------------------|-------|--------|
| MOV AX, 1004H      | 1000H | B80410 |



OPCODE: every instruction is converted into opcode

**PHYSICAL ADDRESS**

**OPCODE**

**DECIMAL EQUIVALENT**

emulator: noname.bin\_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

|    | H    | L  |
|----|------|----|
| AX | 00   | 00 |
| BX | 00   | 00 |
| CX | 00   | 00 |
| DX | 00   | 00 |
| CS | 0100 |    |
| IP | 0000 |    |
| SS | 0100 |    |
| SP | FFFE |    |
| BP | 0000 |    |
| SI | 0000 |    |
| DI | 0000 |    |
| DS | 0100 |    |

0100:0000

| Physical Address | Opcode | Decimal Equivalent |
|------------------|--------|--------------------|
| 01000:           | B8     | 184                |
| 01001:           | 00     | 000                |
| 01002:           | 11     | 017                |
| 01003:           | BB     | 187                |
| 01004:           | 02     | 002                |
| 01005:           | 11     | 017                |
| 01006:           | 03     | 003                |
| 01007:           | C3     | 195                |
| 01008:           | 90     | 144                |
| 01009:           | 90     | 144                |
| 0100A:           | 90     | 144                |
| 0100B:           | 90     | 144                |
| 0100C:           | 90     | 144                |
| 0100D:           | 90     | 144                |
| 0100E:           | 90     | 144                |
| 0100F:           | 90     | 144                |
| 01010:           | 90     | 144                |
| 01011:           | 90     | 144                |
| 01012:           | 90     | 144                |
| 01013:           | 90     | 144                |
| 01014:           | 90     | 144                |
| 01015:           | 90     | 144                |

0100:0000

```
MOV AX, 01100h
MOV BX, 01102h
ADD AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

**Add the following two numbers**

**1AAA H**

**2222 H**

**See the changes in the registers**

**Observe the Physical Address (PA)**

**See the Opcode and give your comments and**

**See the decimal equ of the opcode and show the calculation if required (hex to dec conversion)**

**We see the (Physical) Address, OPCODE, and  
Memory**