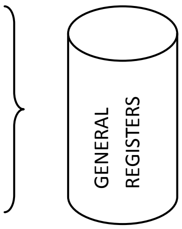
Contents

PRACTICAL NO.1

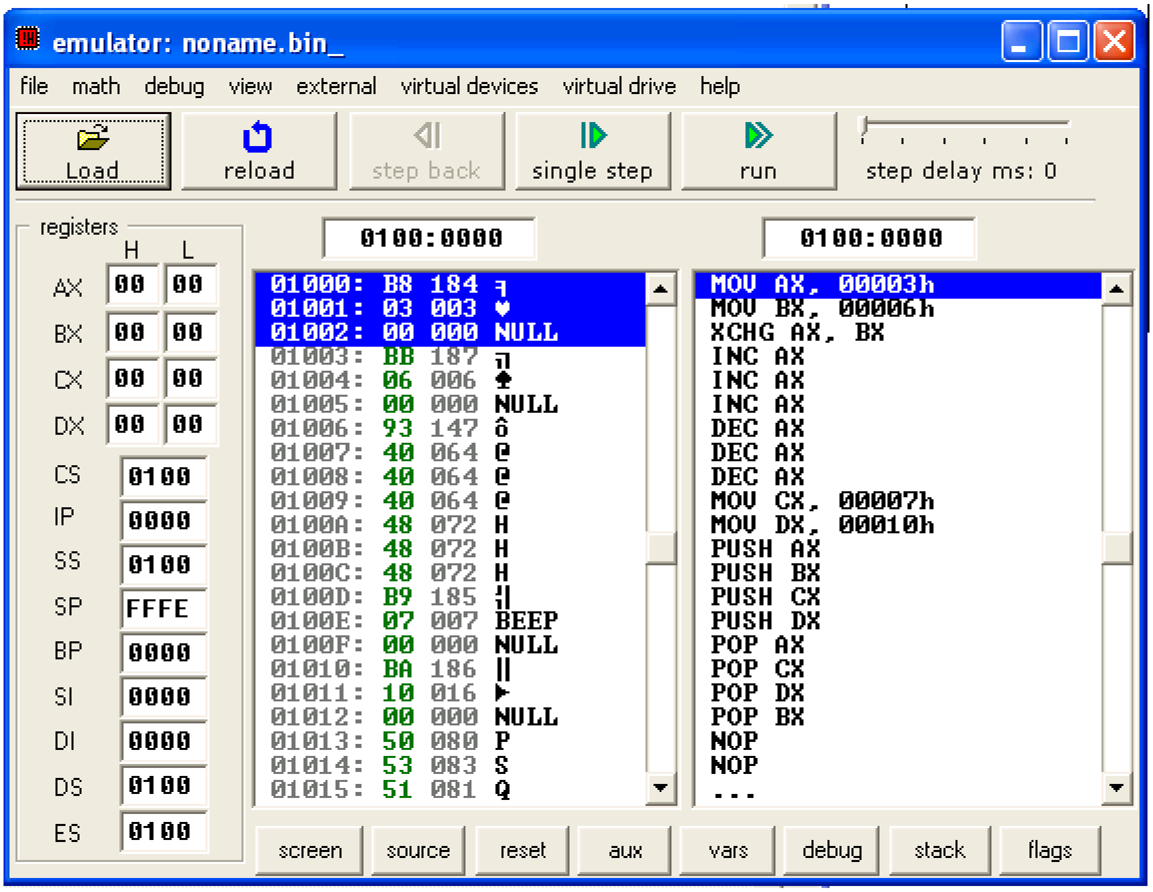
UNDERSTANDING THE REGISTERS OF 8086 MICROPROCESSOR

THE 8086 microprocessor has total of 14 registers which can be classified as:

REGISTER NAME	DESCRIPTION	SIZE
AX 	GENERAL PURPOSE REGISTER USED TO STORE	16 BIT S
BX	DATA .	16 BITS
CX		16 BITS
DX		16 BITS
CS (CODE SEGMENT)	STORES BASE ADRESS OF CODE SEGMENT	16 BITS
IP (INSTRUCTION POINTER)	STORES OFFSET ADRESS OF CODE SEGMENT	16 BITS
SS (STACK SEGMENT)	STORES BASE ADRESS OF STACK SEGMENT	16 BITS
SP (STACK POINTER)	STORES OFFSET ADRESS OF STACK POINTER	16 BITS
BP (BASE POINTER)		16 BITS
SI (SOURCE INDEX)		16 BITS
DI (DESTINATION INDEX)		16 BITS
DS (DATA SEGMENT)		16 BITS
ES (EXTRA SEGMENT)		16 BITS
FLAG (FLAG REGISTER)		8 BITS

General purpose register can also be divided into two sets of 8 bits as higher bits H and lower bits L i.e least significant bits and most significant bits.

REGISTER	HIGHER BITS H	LOWER BITS L
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL



```
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
ES   0100
```

```
0100:0000

01000: B8 184 ╕
01001: 03 003 ♥
01002: 00 000 NULL
01003: BB 187 ╗
01004: 06 006 ♠
01005: 00 000 NULL
01006: 93 147 ô
01007: 40 064 @
01008: 40 064 @
01009: 40 064 @
0100A: 48 072 H
0100B: 48 072 H
0100C: 48 072 H
0100D: B9 185 ╣
0100E: 07 007 BEEP
0100F: 00 000 NULL
01010: BA 186 ║
01011: 10 016 ►
01012: 00 000 NULL
01013: 50 080 P
01014: 53 083 S
01015: 51 081 Q
```

```
0100:0000

MOV  AX, 00003h
MOV  BX, 00006h
XCHG AX, BX
INC  AX
INC  AX
INC  AX
DEC  AX
DEC  AX
DEC  AX
MOV  CX, 00007h
MOV  DX, 00010h
PUSH AX
PUSH BX
PUSH CX
PUSH DX
POP  AX
POP  CX
POP  DX
POP  BX
NOP
NOP
...
```

PRACTICAL NO.2

STUDY THE "MOV" INSTRUCTION

MOV instruction is a very common and basic command used in micro processor to copy contents from source to destination. The general syntax of MOV instruction is:

MOV <SPACE> DESTINATION, SOURCE

DESTINATION can be the name of any register or memory location

Source can be any register name or memory location or any value in decimal, hexadecimal or binary

PRACTICAL NO.3

UNDERSTANDING THE ADRESSING MODES OF 8086 MICRO PROCESSOR

The addressing mode in literature came from two words address which means to talk to share some information and mode which means method. So from addressing mode it means that by which method we communicate with 4micro processor to give instructions.

Addressing modes fall into three major categories'

1. Immediate addressing mode
2. Register addressing mode
3. Memory addressing mode.

In immediate addressing mode data is stored in register of 8086 microprocessor form input given by user and contents of register will be taken from instructions. The general syntax for immediate addressing mode is

MOV <SPACE> REGISTER, VALUE

Using immediate addressing mode, contents in any number system can be transferred to registers. Only need is to specify the number system suffix after the number to be entered, for example

MOV AX, 20 ⇨ Considers 20 as decimal and stores equallent hexadecimal in to AX

MOV BX, 10H ⇨ Stores 10 hexadecimal to BX

MOV CX, 01010101B ⇨ Considers it a binary number and stores hexadecimal

PRACTICAL NO.4

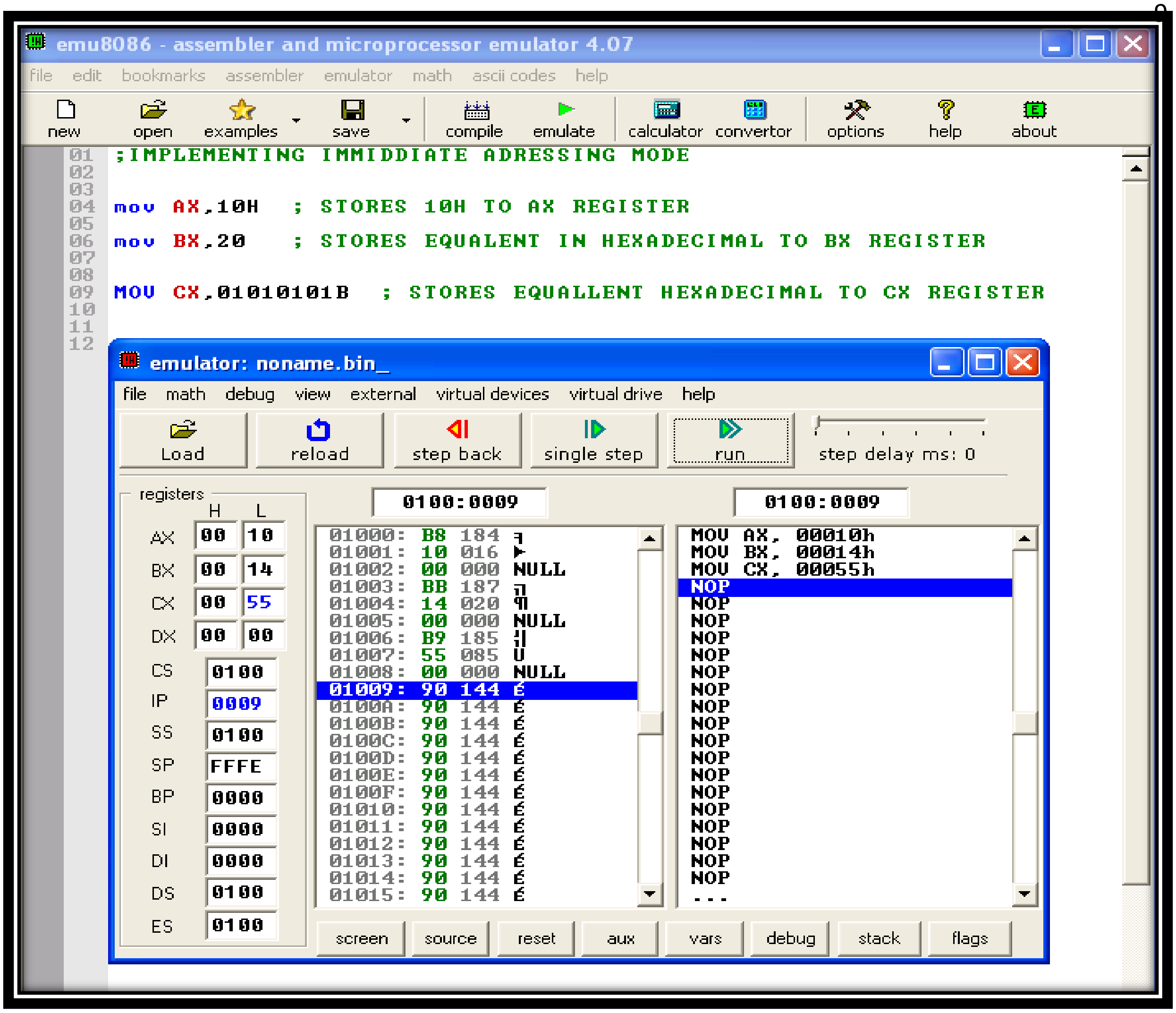
IMPLEMENTING THE IMMIDIATE ADRESSING MODES OF 8086 MICRO PROCESSOR USING EMU-8086 EMULATOR

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator
3. For transferring content of 10H to register ax write the command as **MOV AX, 10H** and write the comments as appropriate.
4. For transferring a binary number as 01010101using immediate addressing mode, write the command **MOV BX, 01010101B** and write comments as appropriate.
5. For transferring a decimal number of 20 write command as **MOV CX, 20** and comments as appropriate.
6. After completing the code click on **EMULATE** button and run the program in single steps.
7. Observe the output of following registers and fill the worksheet as given.

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
;IMPLEMENTING IMMIDDIATE ADRESSING MODE

mov AX,10H   ; STORES 10H TO AX REGISTER

mov BX,20    ; STORES EQUALENT IN HEXADECIMAL TO BX REGISTER

MOV CX,01010101B  ; STORES EQUALLENT HEXADECIMAL TO CX REGISTER
```

registers

	H	L
AX	00	10
BX	00	14
CX	00	55
DX	00	00

CS	0100
IP	0009
SS	0100
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0100
ES	0100

```
0100:0009

01000: B8 184 ╕
01001: 10 016 ►
01002: 00 000 NULL
01003: BB 187 ╗
01004: 14 020 ¶
01005: 00 000 NULL
01006: B9 185 ╣
01007: 55 085 U
01008: 00 000 NULL
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0009

MOV AX, 00010h
MOV BX, 00014h
MOV CX, 00055h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

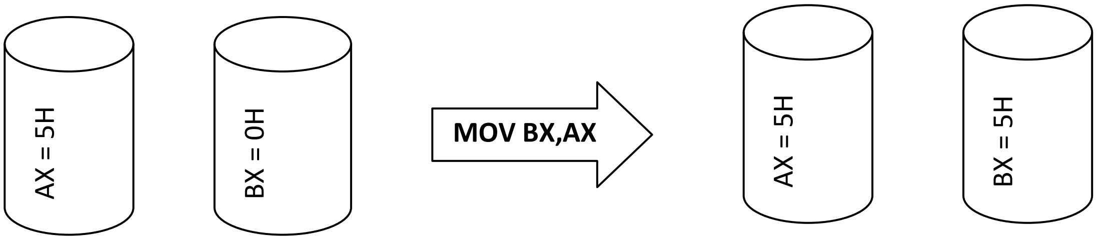
WORK SHEET:

S.NO	REGISTERS VALUES	INITIAL	1ST STEP	2ND STEP	3RD STEP	4TH STEP
1	AX					
2	BX					
3	CX					
4	CS					
5	IP					

PRACTICAL NO.5

IMPLEMENTING THE REGISTER ADRESSING MODES OF 8086 MICRO PROCESSOR USING EMU-8086 EMULATOR

In the register addressing mode using **MOV** instruction contents of one register is copied to the other register.



In the above example BX register is destination register and AX is source register. Recall from previous example that in MOV instruction uses Destination and Source as

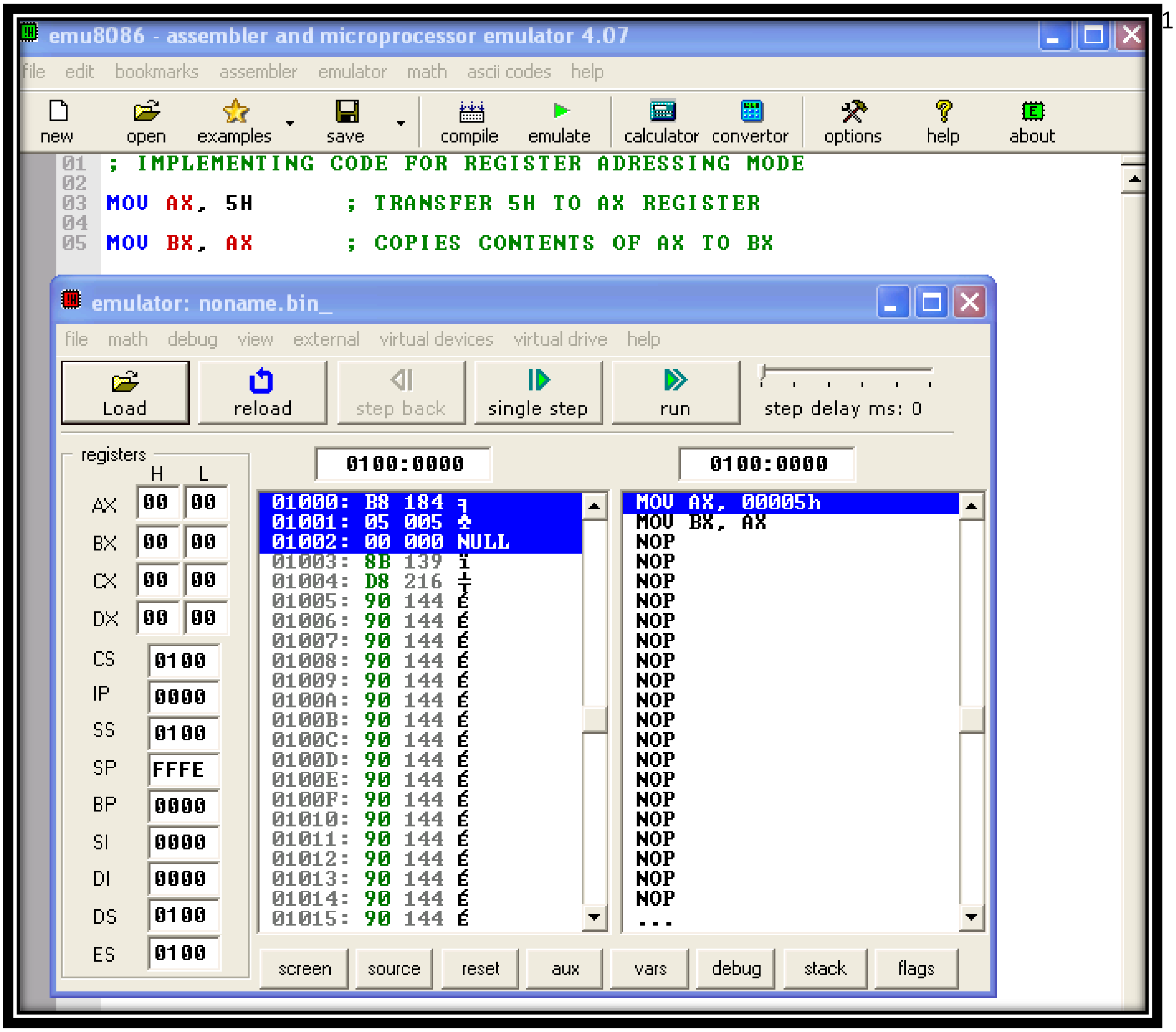
MOV BX,AX

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator
3. For transferring content of 5H to register AX write the command as **MOV AX, 5H** and write the comments as appropriate.
4. For transferring the contents of AX to BX write command **MOV BX,AX**
5. After completing the code click on **EMULATE** button and run the program in single steps.
6. Observe the output of following registers and fill the worksheet as given.

PRECAUSIONS:

7. Never use infinite loop in any coding.
8. Always emulate the code in single instruction.
9. Care fully observes the output of registers.


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
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
screen source reset aux vars debug stack flags

```
; IMPLEMENTING CODE FOR REGISTER ADRESSING MODE

MOV AX, 5H        ; TRANSFER 5H TO AX REGISTER

MOV BX, AX        ; COPIES CONTENTS OF AX TO BX
```

Register	H	L
AX	00	00
BX	00	00
CX	00	00
DX	00	00

Register	Value
CS	0100
IP	0000
SS	0100
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0100
ES	0100

0100:0000

```
01000: B8 184 ╕
01001: 05 005 ♣
01002: 00 000 NULL
01003: 8B 139 ï
01004: D8 216 ╪
01005: 90 144 É
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

0100:0000

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

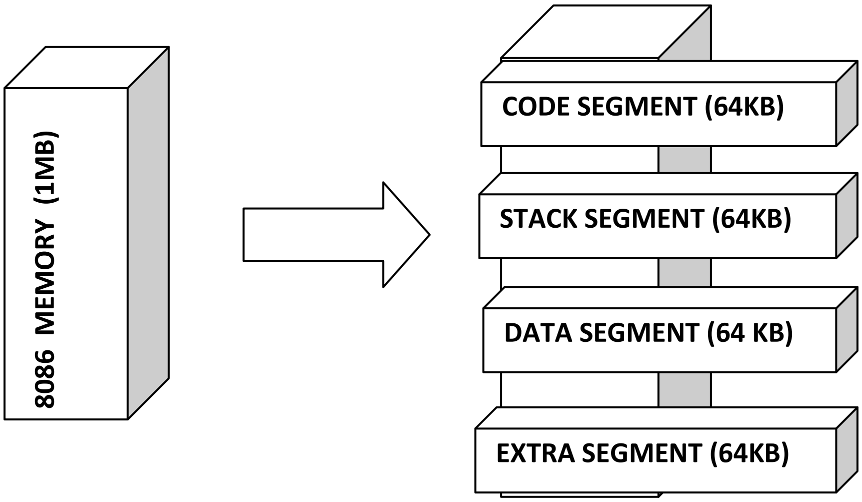
WORK SHEET:

S.NO	REGISTERS VALUES	INITIAL	1^{ST} STEP	2^{ND} STEP	3^{RD} STEP	4^{TH} STEP
1	AX					
2	BX					
3	CS					
4	IP					

PRACTICAL NO.6

UNDERSTANDING THE MEMORY ADRESSING MODE

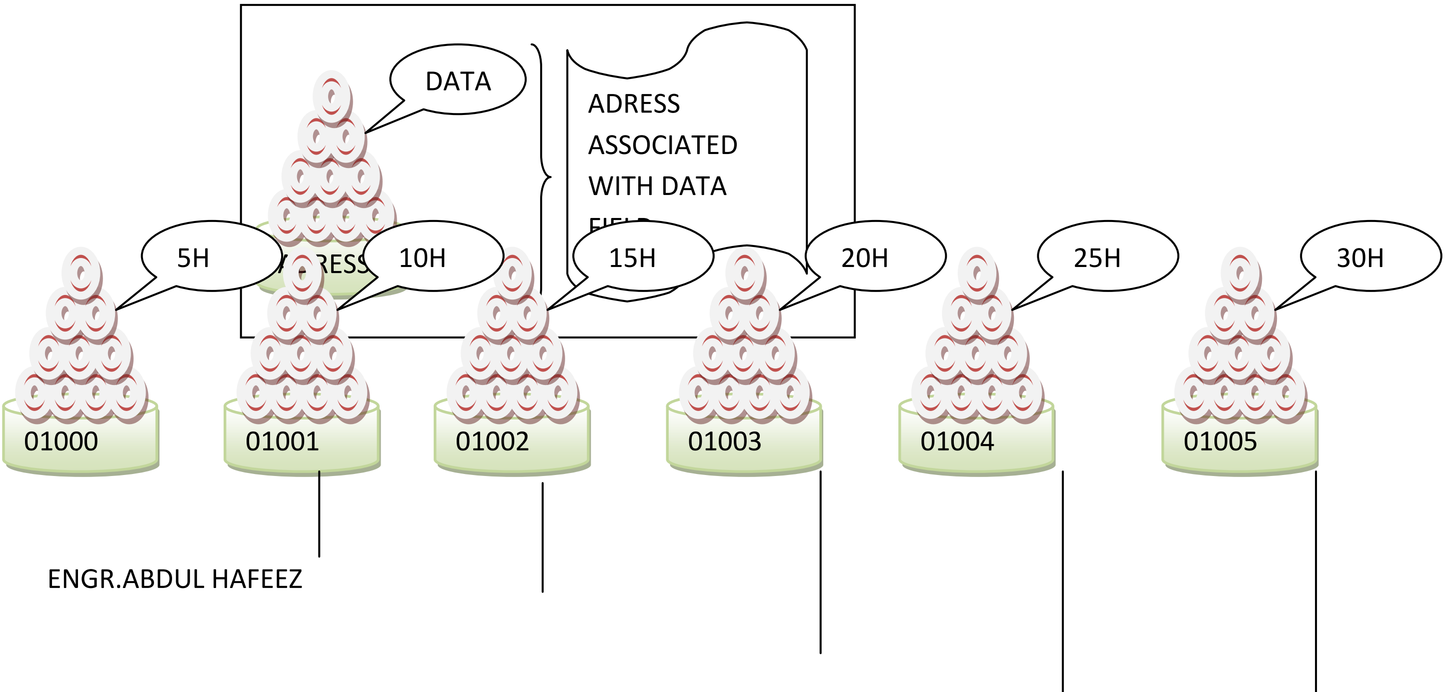
Memory addressing mode is used to transfer data from memory to register and from register to memory by using **MOV** instruction. The total memory of 8086 microprocessor is divided into four parts or segments that is

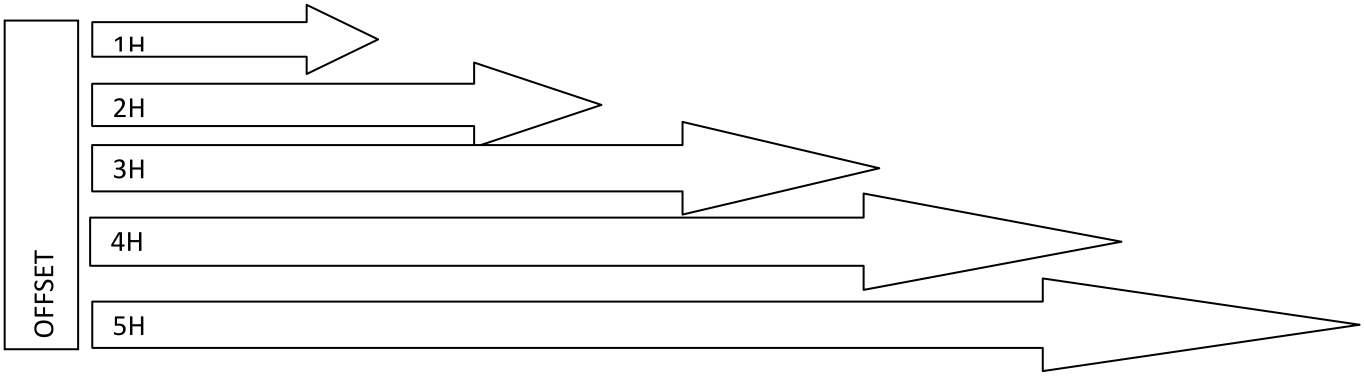


CONCEPT OF ADRESSES:

Address specifies the memory location. Each memory location is specified by a unique address. Accessing a specific memory location involves base addresses and offset addresses.

Base address is the address from which a specific segment starts and accessing any memory location within that segment involves offset address. Each memory location is accessed by that reference address called base address and by increasing offset value different memory fields are accessed.





Increment in base address to access a specific memory location is called offset address.

Memory addressing modes can be classified into five categories as:

1. Direct addressing mode
2. Register indirect addressing mode
3. Register relative addressing mode
4. Base index addressing mode
5. Base index relative addressing mode

Each memory addressing mode utilized the fact that the difference is that method of giving offset address is different for every addressing mode.

PRACTICAL NO.7

UNDERSTANDING MEMORY DIRECT ADDRESSING MODE

In memory direct addressing mode offset is provided directly in the instrution and contents of data segment memory is used.

Transfering data from register to memory:

Suppose data segment base address is 0100 as data. Physical address is calculated as

P.A = DS*10H

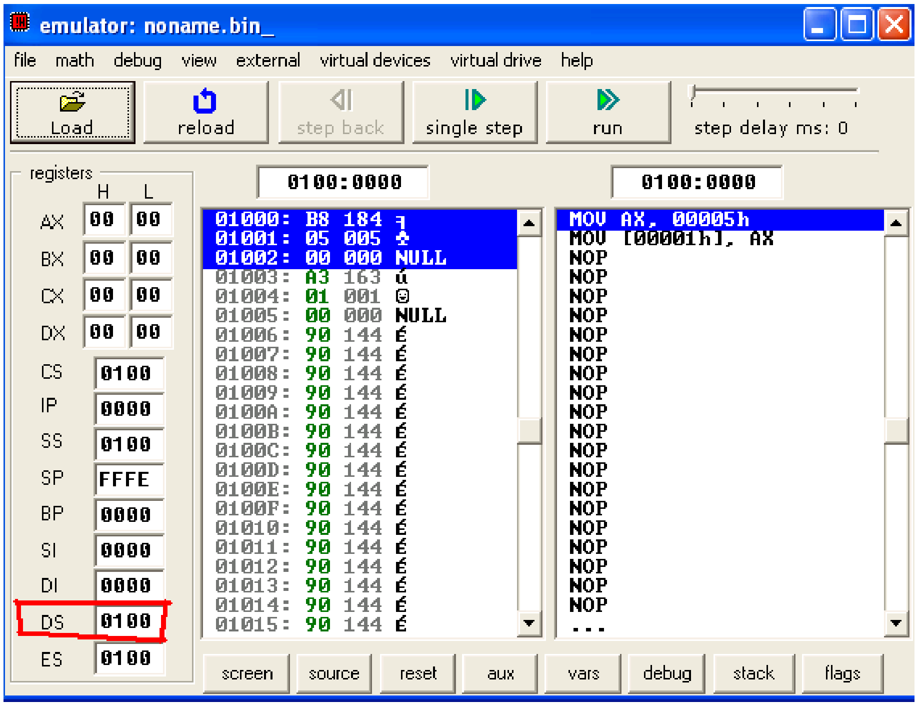
Memory location can be accesed by adding offset into physical address. MOV instruction for this case can be modified as:

MOV DESTINATION, SOURCE

MOV <SPACE> [OFFSET], REGISTER

By default memory of data segment DS is accessed.

DS shows base address of data segment and offset is given in the instruction field directly



```
    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
ES  0100
```

```
0100:0000

01000: B8 184 ╕
01001: 05 005 ♣
01002: 00 000 NULL
01003: A3 163 ú
01004: 01 001 ☺
01005: 00 000 NULL
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0000

MOV  AX, 00005h
MOV  [00001h], AX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.8

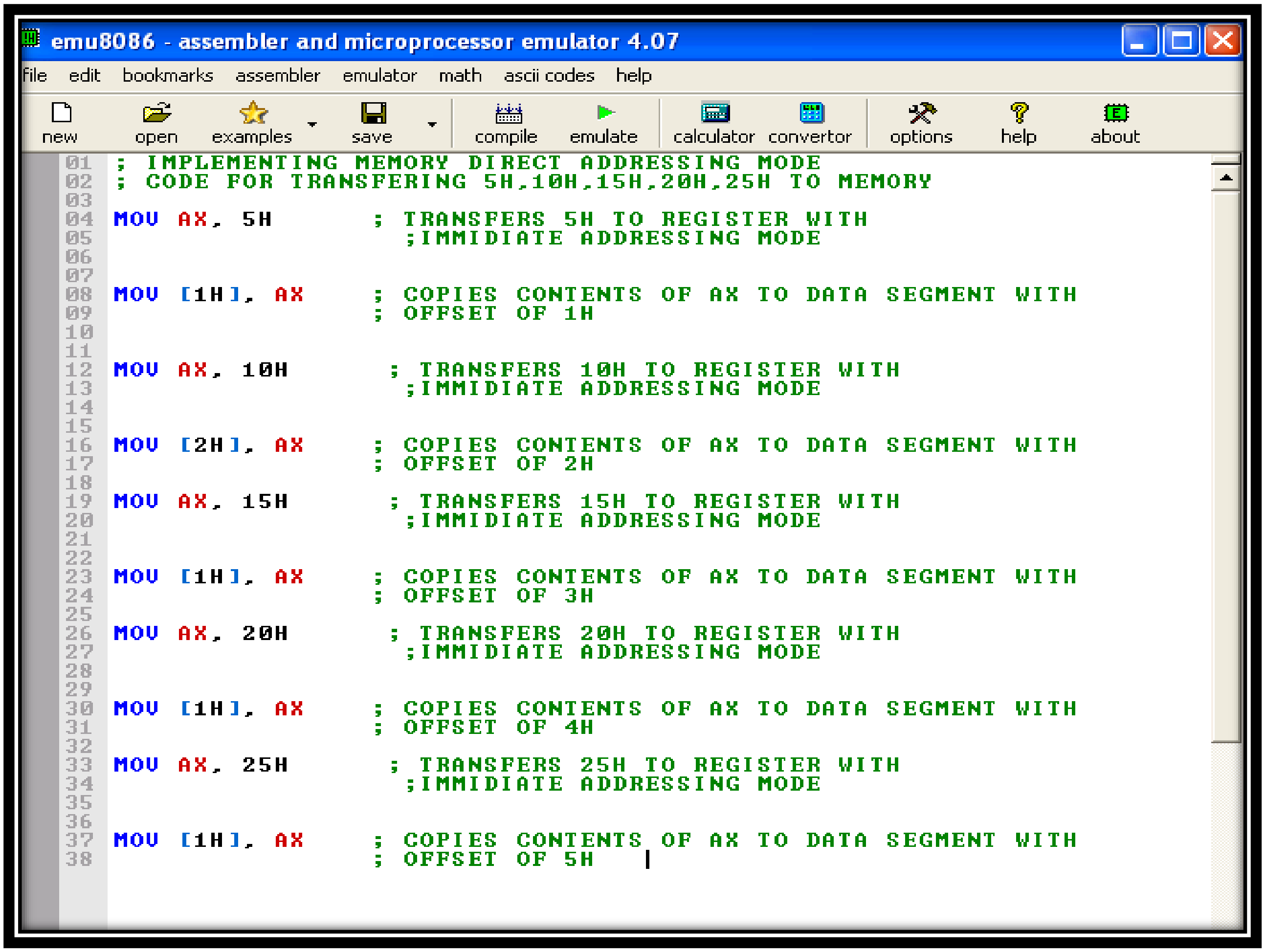
USING MEMORY DIRECT ADDRESSING MODE TO TRANSFER DATA (5H, 10H, 15H, 20H, 25H) TO MEMORY

PROCEDURE:

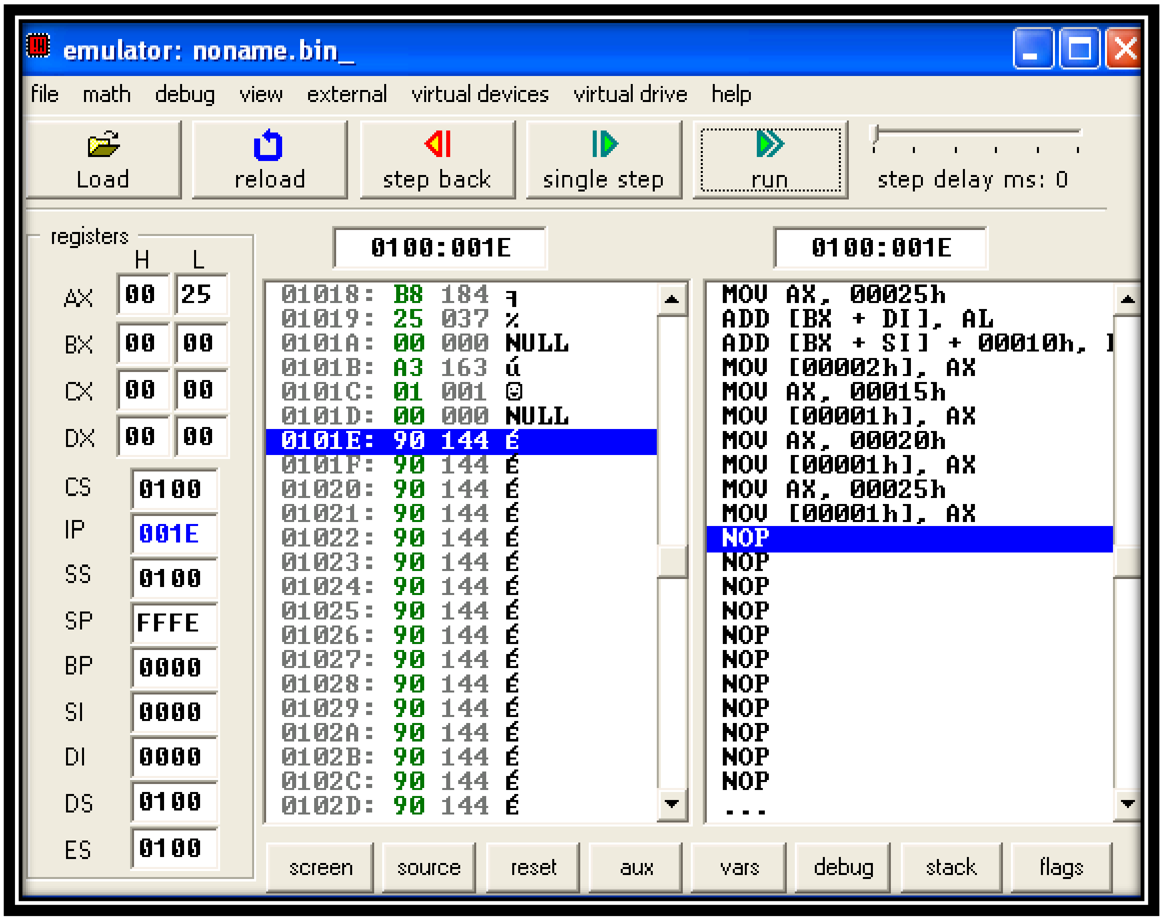
1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator
3. For transferring content of 5H to register AX write the command as **MOV AX, 5H** and write the comments as appropriate.
4. For transferring the contents of AX to memory in data segment with offset of 1H write command **MOV [1H],AX**
5. For transferring content of 10H to register AX write the command as **MOV AX, 10H** and write the comments as appropriate.
6. For transferring the contents of AX to memory in data segment with offset of 2H write command **MOV [2H],AX**
7. For transferring content of 15H to register AX write the command as **MOV AX, 15H** and write the comments as appropriate.
8. For transferring the contents of AX to memory in data segment with offset of 3H write command **MOV [3H],AX**
9. Repeat the code up to 25H data field with offset of 5H
10. After completing the code click on **EMULATE** button and run the program in single steps.
11. Observe the output of following registers and fill the worksheet as given.

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

```
; IMPLEMENTING MEMORY DIRECT ADDRESSING MODE
; CODE FOR TRANSFERING 5H,10H,15H,20H,25H TO MEMORY

MOV AX, 5H         ; TRANSFERS 5H TO REGISTER WITH
                    ;IMMIDIATE ADDRESSING MODE


MOV [1H], AX       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                   ; OFFSET OF 1H


MOV AX, 10H         ; TRANSFERS 10H TO REGISTER WITH
                     ;IMMIDIATE ADDRESSING MODE


MOV [2H], AX       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                   ; OFFSET OF 2H

MOV AX, 15H         ; TRANSFERS 15H TO REGISTER WITH
                     ;IMMIDIATE ADDRESSING MODE


MOV [1H], AX       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                   ; OFFSET OF 3H

MOV AX, 20H         ; TRANSFERS 20H TO REGISTER WITH
                     ;IMMIDIATE ADDRESSING MODE


MOV [1H], AX       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                   ; OFFSET OF 4H

MOV AX, 25H         ; TRANSFERS 25H TO REGISTER WITH
                     ;IMMIDIATE ADDRESSING MODE


MOV [1H], AX       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                   ; OFFSET OF 5H
```


emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

registers

	H	L
AX	00	25
BX	00	00
CX	00	00
DX	00	00

CS	0100
IP	001E
SS	0100
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0100
ES	0100

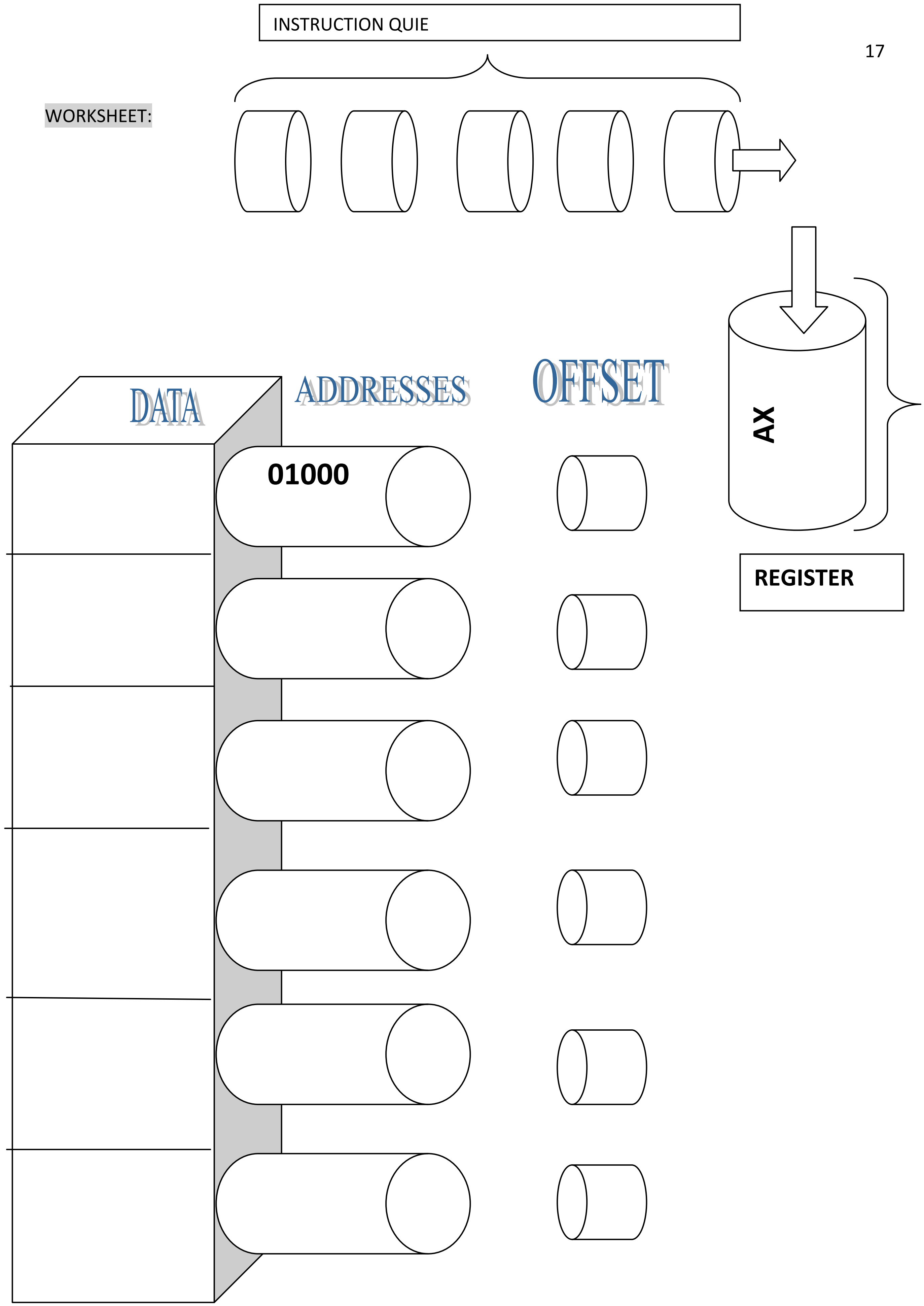
0100:001E

```
01018: B8 184 ╕
01019: 25 037 %
0101A: 00 000 NULL
0101B: A3 163 ú
0101C: 01 001 ☺
0101D: 00 000 NULL
0101E: 90 144 É
0101F: 90 144 É
01020: 90 144 É
01021: 90 144 É
01022: 90 144 É
01023: 90 144 É
01024: 90 144 É
01025: 90 144 É
01026: 90 144 É
01027: 90 144 É
01028: 90 144 É
01029: 90 144 É
0102A: 90 144 É
0102B: 90 144 É
0102C: 90 144 É
0102D: 90 144 É
```

0100:001E

```
MOV AX, 00025h
ADD [BX + DI], AL
ADD [BX + SI] + 00010h, I
MOV [00002h], AX
MOV AX, 00015h
MOV [00001h], AX
MOV AX, 00020h
MOV [00001h], AX
MOV AX, 00025h
MOV [00001h], AX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

WORKSHEET:



PRACTICAL NO.9

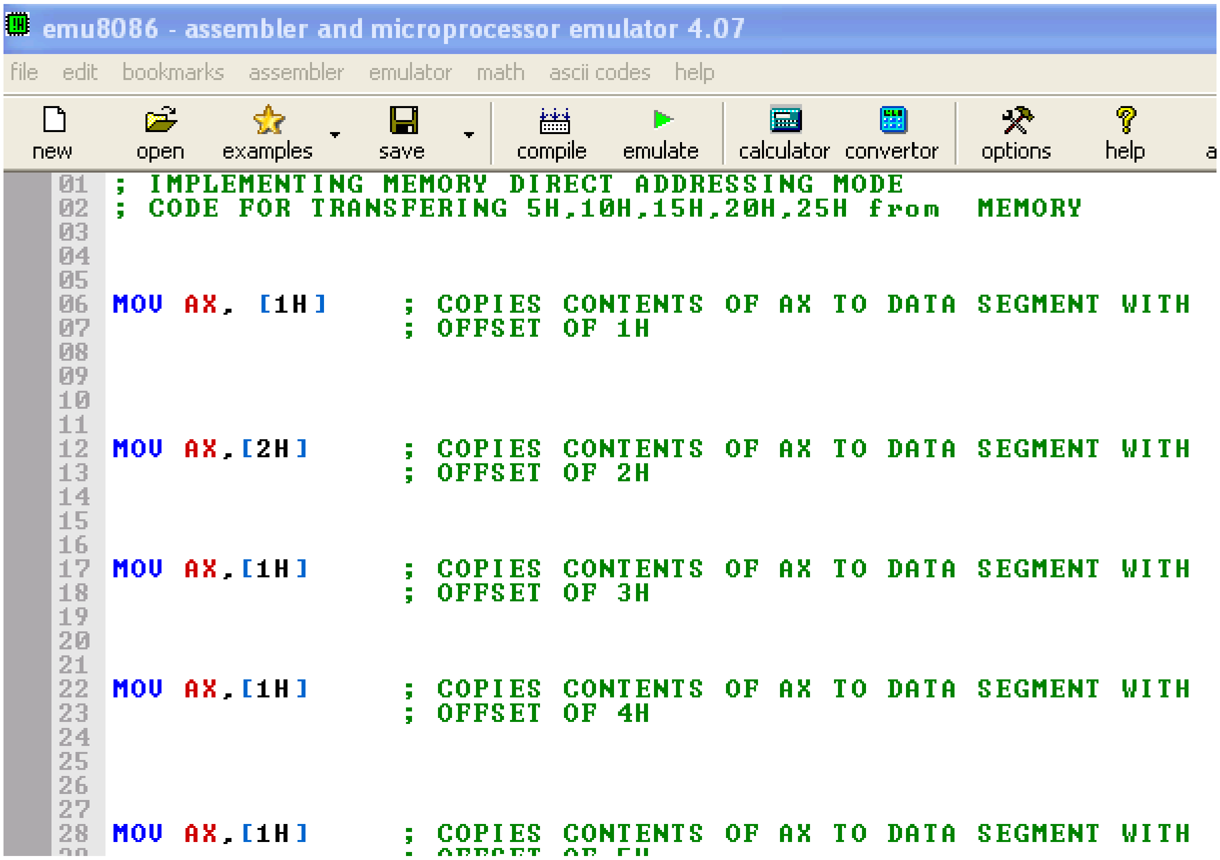
USING MEMORY DIRECT ADDRESSING MODE TO TRANSFER DATA (5H,10H,15H,20H,25H) FROM MEMORY TO REGISTER

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator
3. For transferring the contents of AX from memory in data segment with offset of 1H to AX write command **MOV AX,[1H]**
4. For transferring the contents of AX to memory in data segment with offset of 2H write command **MOV AX,[2H]**
5. For transferring the contents of AX to memory in data segment with offset of 3H write command **MOV AX,3[H]**
6. Repeat the code up to offset of 5H
7. After completing the code click on **EMULATE** button and run the program in single steps.
8. Observe the output of following registers and fill the worksheet as given.

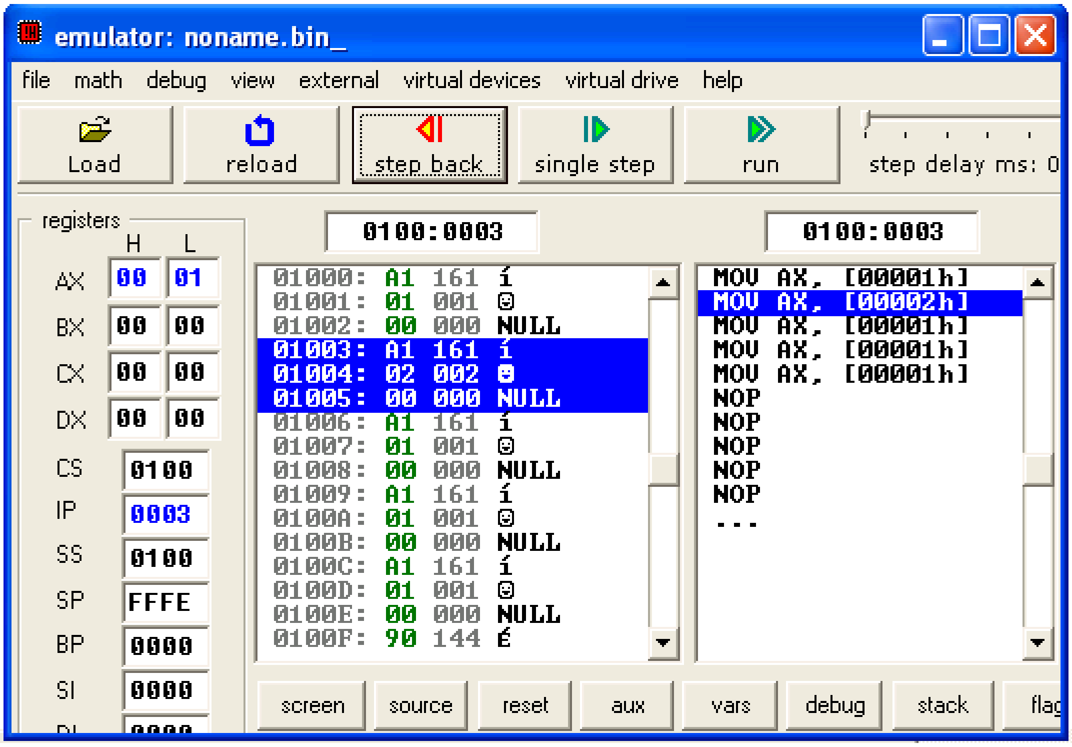
PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.



emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new
open
examples
save
compile
emulate
calculator
convertor
options
help
a

```
; IMPLEMENTING MEMORY DIRECT ADDRESSING MODE
; CODE FOR TRANSFERING 5H,10H,15H,20H,25H from  MEMORY



MOV AX, [1H]      ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 1H




MOV AX,[2H]       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 2H



MOV AX,[1H]       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 3H



MOV AX,[1H]       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 4H





MOV AX,[1H]       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 5H
```



	H	L
AX	00	01
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	0003	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	

```
0100:0003

01000: A1 161 í
01001: 01 001 ☺
01002: 00 000 NULL
01003: A1 161 í
01004: 02 002 ☻
01005: 00 000 NULL
01006: A1 161 í
01007: 01 001 ☺
01008: 00 000 NULL
01009: A1 161 í
0100A: 01 001 ☺
0100B: 00 000 NULL
0100C: A1 161 í
0100D: 01 001 ☺
0100E: 00 000 NULL
0100F: 90 144 É
```

```
0100:0003

MOV AX, [00001h]
MOV AX, [00002h]
MOV AX, [00001h]
MOV AX, [00001h]
MOV AX, [00001h]
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.10

USING REGISTER INDIRECT ADDRESSING MODE TO TRANSFER DATA(5H,10H,15H) MEMORY TO REGISTER

The general syntax for memory indirect addressing mode is

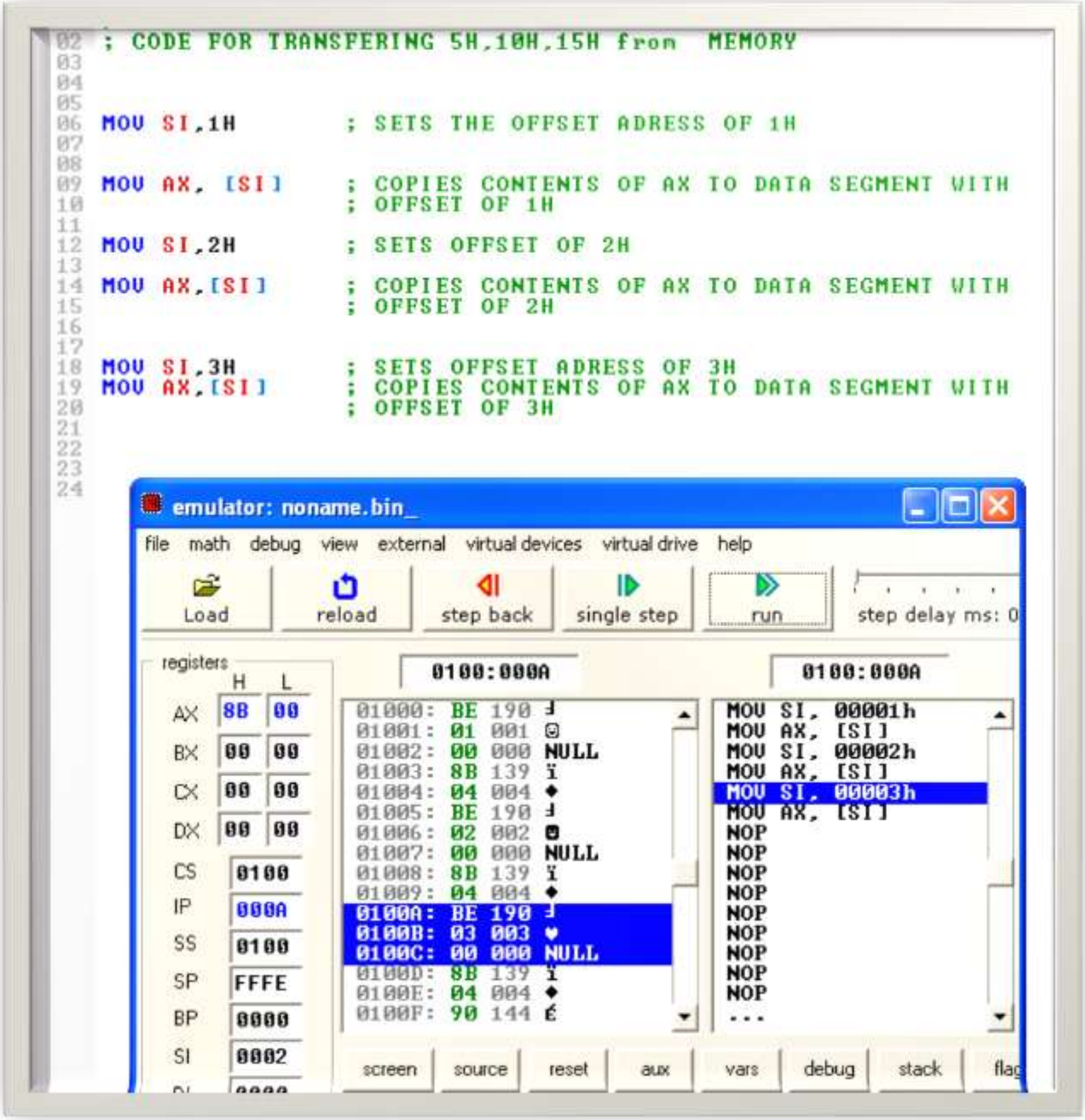
Mov "register", [SI OR DI OR BX]

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator
3. For transferring offset of 1H to SI write command
   MOV SI,1H
4. For transferring the contents of memory in data segment to AX with offset of 1H write command **MOV AX,[SI]**
5. Repeat the code up to offset of 3H FOR DATA 10H &15H
6. After completing the code click on **EMULATE** button and run the program in single steps.
7. Observe the output of following registers and fill the worksheet as given.

PRECAUSIONS:

8. Never use infinite loop in any coding.
9. Always emulate the code in single instruction.
10. Care fully observes the output of registers.



```
; CODE FOR TRANSFERING 5H,10H,15H from  MEMORY



MOV SI,1H         ; SETS THE OFFSET ADRESS OF 1H


MOV AX, [SI]      ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 1H

MOV SI,2H         ; SETS OFFSET OF 2H

MOV AX,[SI]       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 2H


MOV SI,3H         ; SETS OFFSET ADRESS OF 3H
MOV AX,[SI]       ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 3H
```

registers	H	L
AX	8B	00
BX	00	00
CX	00	00
DX	00	00
CS	0100	
IP	000A	
SS	0100	
SP	FFFE	
BP	0000	
SI	0002	

```
0100:000A

01000: BE 190 ╛
01001: 01 001 ☺
01002: 00 000 NULL
01003: 8B 139 ï
01004: 04 004 ♦
01005: BE 190 ╛
01006: 02 002 ☻
01007: 00 000 NULL
01008: 8B 139 ï
01009: 04 004 ♦
0100A: BE 190 ╛
0100B: 03 003 ♥
0100C: 00 000 NULL
0100D: 8B 139 ï
0100E: 04 004 ♦
0100F: 90 144 É
```

```
0100:000A

MOV SI, 00001h
MOV AX, [SI]
MOV SI, 00002h
MOV AX, [SI]
MOV SI, 00003h
MOV AX, [SI]
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.11

USING REGISTER RELATIVE ADDRESSING MODE TO TRANSFER DATA(5H,10H,15H) MEMORY TO REGISTER

The general syntax for register relative addressing mode is

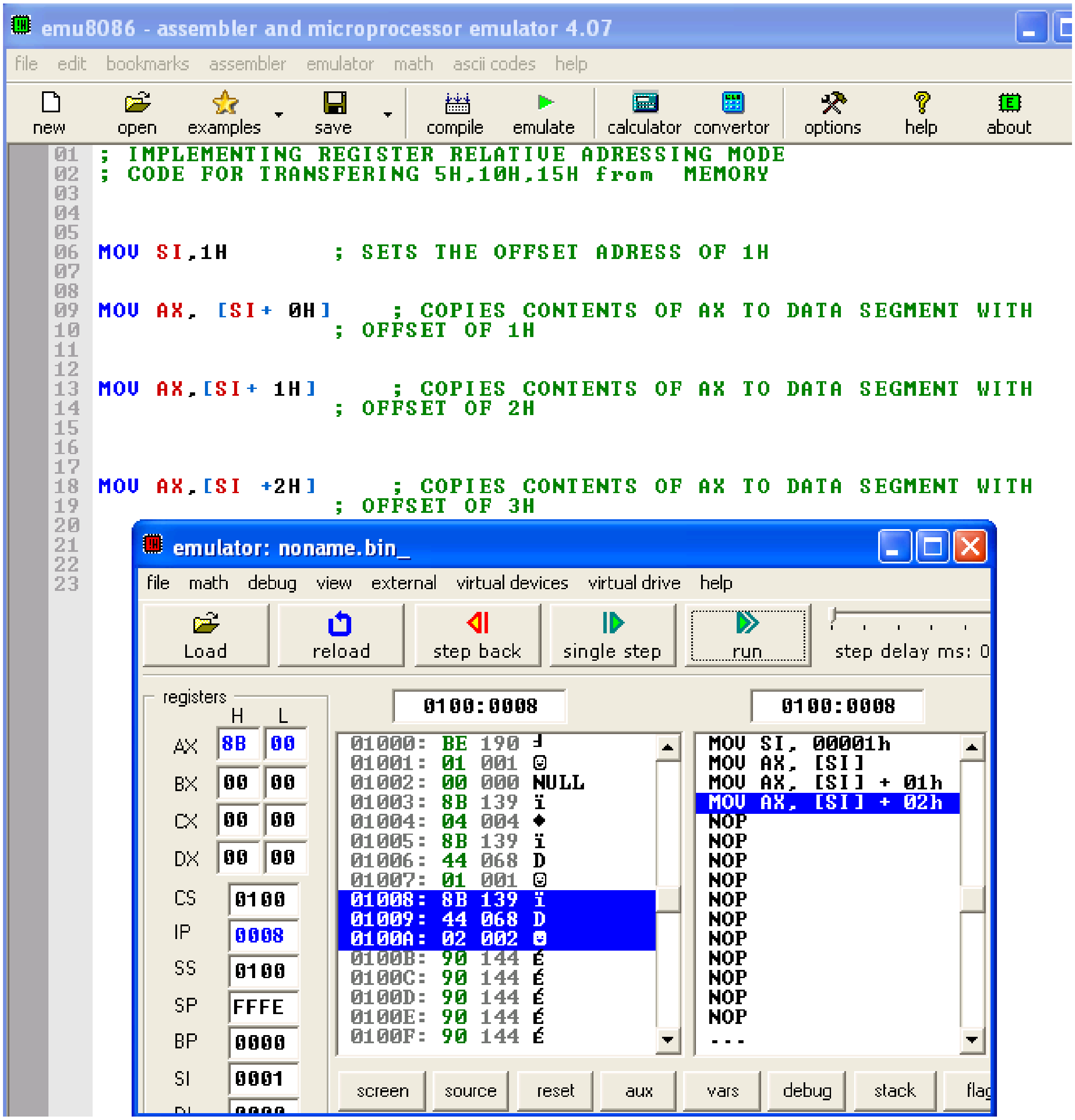
$$EA=\begin{bmatrix}(BX)\\(BP)\\(DI)\\(SI)\end{bmatrix}+\text{Displacement}\qquad\Big]$$

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator
3. For transferring offset of 1H to SI write command
   MOV SI,1H
4. For transferring the contents of memory in data segment to AX with offset of 1H write command **MOV AX,[SI + 0h]**
5. For transferring the contents of memory in data segment with offset of 2H write command **MOV AX,[SI + 1h]** so the total offset will be of 2h
6. For transferring the contents of memory in data segment with offset of 2H write command **MOV AX,[SI + 2h]** so the total offset will be of 3h
7. After completing the code click on **EMULATE** button and run the program in single steps.
8. Observe the output of following registers and fill the worksheet as given.

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.



```
; IMPLEMENTING REGISTER RELATIVE ADRESSING MODE
; CODE FOR TRANSFERING 5H,10H,15H from  MEMORY


MOV SI,1H         ; SETS THE OFFSET ADRESS OF 1H


MOV AX, [SI+ 0H]     ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 1H


MOV AX,[SI+ 1H]      ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 2H


MOV AX,[SI +2H]      ; COPIES CONTENTS OF AX TO DATA SEGMENT WITH
                  ; OFFSET OF 3H
```

```
registers
     H   L
AX   8B  00
BX   00  00
CX   00  00
DX   00  00
CS   0100
IP   0008
SS   0100
SP   FFFE
BP   0000
SI   0001
DI   0000
```

```
0100:0008

01000: BE 190 ╛
01001: 01 001 ☺
01002: 00 000 NULL
01003: 8B 139 ï
01004: 04 004 ♦
01005: 8B 139 ï
01006: 44 068 D
01007: 01 001 ☺
01008: 8B 139 ï
01009: 44 068 D
0100A: 02 002 ☻
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
```

```
0100:0008

MOV SI, 00001h
MOV AX, [SI]
MOV AX, [SI] + 01h
MOV AX, [SI] + 02h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.12

USING BASE INDEX ADDRESSING MODE TO TRANSFER DATA(5H,10H,15H) MEMORY TO REGISTER

The general syntax for BASE INDEX addressing mode is

$$EA=\begin{bmatrix}(BX)\\(BP)\end{bmatrix}+\begin{bmatrix}(DI)\\(SI)\end{bmatrix}$$

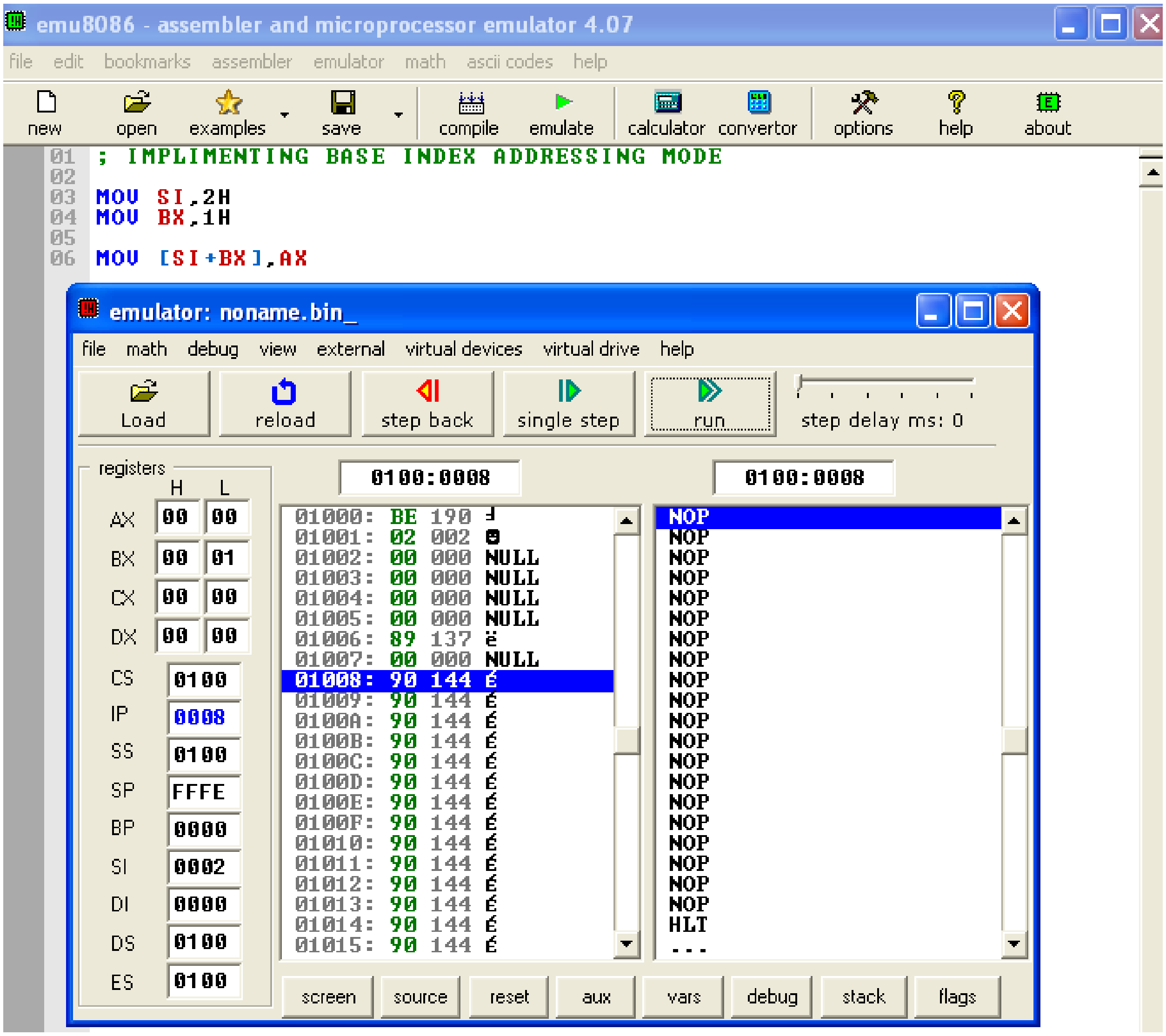
The offset is provided in the base register and index register

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. For transferring the offset of 3h break them into sum of two parts i.e 3h = 2h+1h
4. For transferring offset part of 2H to SI write command
   MOV SI,2H
5. For transferring the 2nd part of offset to BX write the command as mov BX,1h
6. Now for transferring contents to AX write command as Mov AX, 5h.
7. For transferring contents of AX register to memory location at offset of 3h write command as Mov [bx + SI], AX
8. After completing the code click on **EMULATE** button and run the program in single steps.
9. Observe the output of following registers and fill the worksheet as given.

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.



emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
; IMPLIMENTING BASE INDEX ADDRESSING MODE

MOV SI,2H
MOV BX,1H

MOV [SI+BX],AX
```

```
registers
     H   L
AX   00  00
BX   00  01
CX   00  00
DX   00  00
CS   0100
IP   0008
SS   0100
SP   FFFE
BP   0000
SI   0002
DI   0000
DS   0100
ES   0100
```

```
0100:0008
01000: BE 190 ╛
01001: 02 002 ☻
01002: 00 000 NULL
01003: 00 000 NULL
01004: 00 000 NULL
01005: 00 000 NULL
01006: 89 137 ë
01007: 00 000 NULL
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
...
```

PRACTICAL NO.13

USING BASE INDEX RELATIVE ADDRESSING MODE TO TRANSFER DATA(5H,10H,15H) MEMORY TO REGISTER

THEORY: The general syntax for register relative addressing mode is

$$EA=\begin{bmatrix}(BX)\\(BP)\end{bmatrix}+\begin{bmatrix}(DI)\\(SI)\end{bmatrix}+\begin{bmatrix}\text{DISPLACEMENT}\end{bmatrix}$$

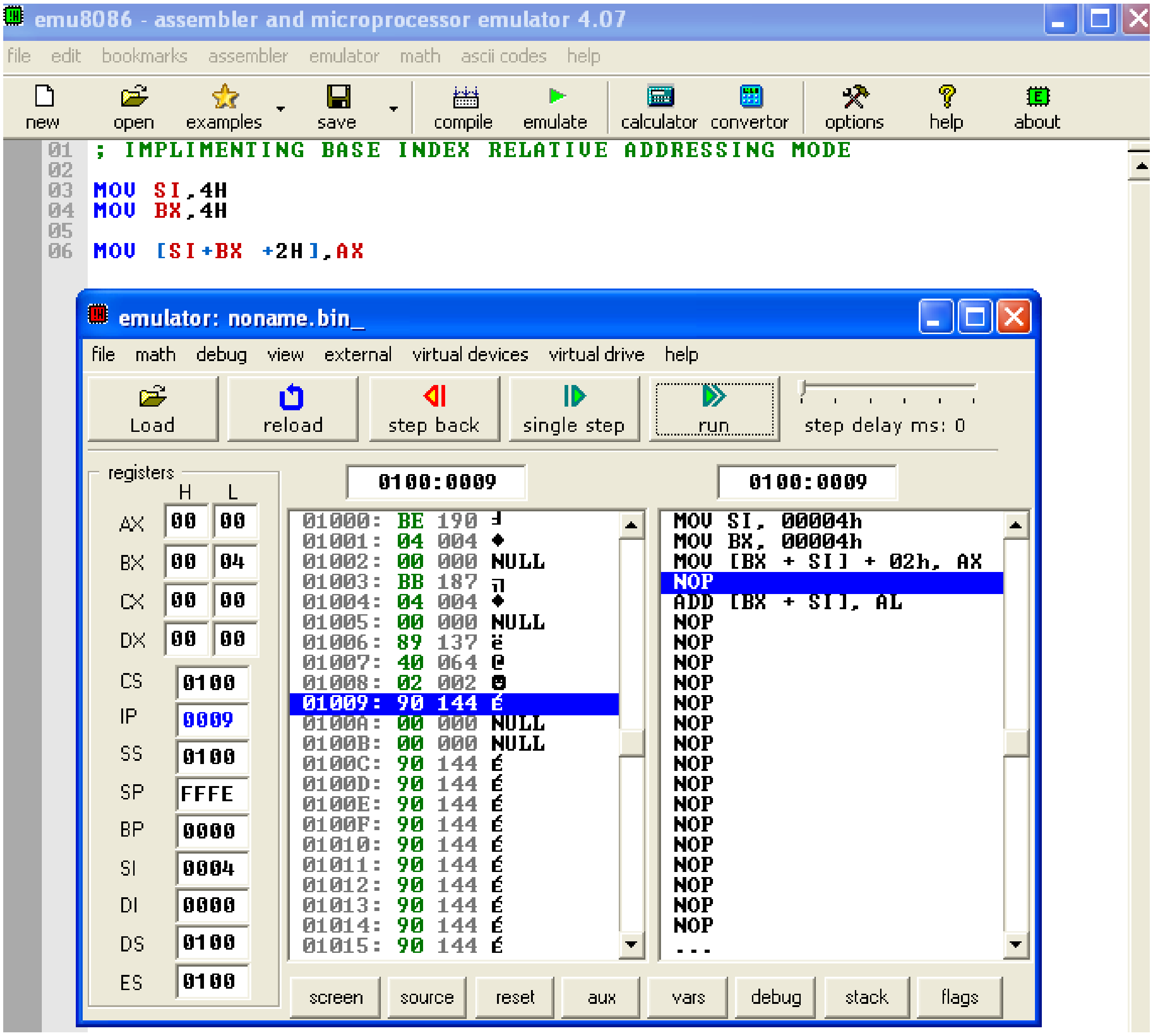
The offset is provided in the base register and index register + displacement

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. For transferring the offset of 10h break them into sum of two parts i.e 10h = 4h+4h+2h
4. For transferring offset part of 4H to SI write command MOV SI,4H
5. For transferring the 2nd part of offset to BX write the command as mov BX,4h
6. Now for transferring contents to AX write command as Mov AX, 5h.
7. For transferring contents of AX register to memory location at offset of 3h write command as Mov [bx + SI+2h], AX
8. After completing the code click on **EMULATE** button and run the program in single steps.
9. Observe the output of following registers and fill the worksheet as given.

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run
screen source reset aux vars debug stack flags

```
; IMPLIMENTING BASE INDEX RELATIVE ADDRESSING MODE

MOV  SI,4H
MOV  BX,4H

MOV  [SI+BX  +2H],AX
```

step delay ms: 0

```
registers
      H    L
AX    00   00
BX    00   04
CX    00   00
DX    00   00
CS    0100
IP    0009
SS    0100
SP    FFFE
BP    0000
SI    0004
DI    0000
DS    0100
ES    0100
```

```
0100:0009

01000: BE 190 ╛
01001: 04 004 ♦
01002: 00 000 NULL
01003: BB 187 ╗
01004: 04 004 ♦
01005: 00 000 NULL
01006: 89 137 ë
01007: 40 064 @
01008: 02 002 ☻
01009: 90 144 É
0100A: 00 000 NULL
0100B: 00 000 NULL
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0009

MOV  SI, 00004h
MOV  BX, 00004h
MOV  [BX + SI] + 02h, AX
NOP
ADD  [BX + SI], AL
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.14

DEMONSTATING MOV INSTRUCTION SET

The general syntax for MOV instruction set is

MOV DESTINATION,SOURCE

The source can be any register memory location or hexadecimal number. The destination can be any register or memory location

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write move instruction to verify the data transfer to different registers of 8086 microprocessor.
4. Mov AX,2H
5. MOV BX,3H
6. MOV CX,4H
7. MOV DX,10H
8. MOV CS,44H
9. MOV IP, 55H
10. MOV SS,41H
11. MOV SP, 42H
12. MOV BP,31H
13. MOV SI,32H
14. MOV DI,33H
15. MOV DS, 49H
16. MOV ES, 47H
17. MOV AH, 456H
18. MO DL, 44H

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.

READINGS

NOTE: Tick mark the command that executed and those that not executed write faults

S.NO	COMMAND	COMMAND EXECUTED	COMMAND NOT EXECUTED	FAULT
1.	1. Mov AX,2H 2. MOV BX,3H 3. MOV CX,4H 4. MOV DX,10H 5. MOV CS,44H 6. MOV IP, 55H 7. MOV SS,41H 8. MOV SP, 42H 9. MOV BP,31H 10. MOV SI,32H 11. MOV DI,33H 12. MOV DS, 49H 13. MOV ES, 47H 14. MOV AH, 456H 15. MO DL, 44H			

PRACTICAL NO.15

DEMONSTATING XCHANGE INSTRUCTION SET

The general syntax for XCHG instruction set is

```
XCHG REGISTER,REGISTER
```

Xchg instruction exchanges the contents between two memory locations or registers.

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to two registers
4. Write command Mov Ax, 44h
5. Write the command Mov Bx,33h
6. Write the command Xchg Ax,Bx
7. Fill out the readings given for results

PRECAUSIONS:

8. Never use infinite loop in any coding.
9. Always emulate the code in single instruction.
10. Care fully observes the output of registers.

READINGS

S.NO	COMMANDS	RESULTS
1	MOV AX, 44H	AX = BX=
2	MOV BX,33H	AX = BX=
3.	XCHG AX,BX	AX = BX=

PRACTICAL NO.16

DEMONSTATING PUSH INSTRUCTION SET

The general syntax for PUSH instruction set is

PUSH REGISTER

PUSH Instruction is used to send contents of register or memory to stack segment. In stack segment stack segment register (ss) and stack pointer register (sp) work together. Push instruction executes in following steps

1. Stack pointer register (sp) is decremented by 2H
2. Data is copied from register to stack memory location.

PROCEDURE:

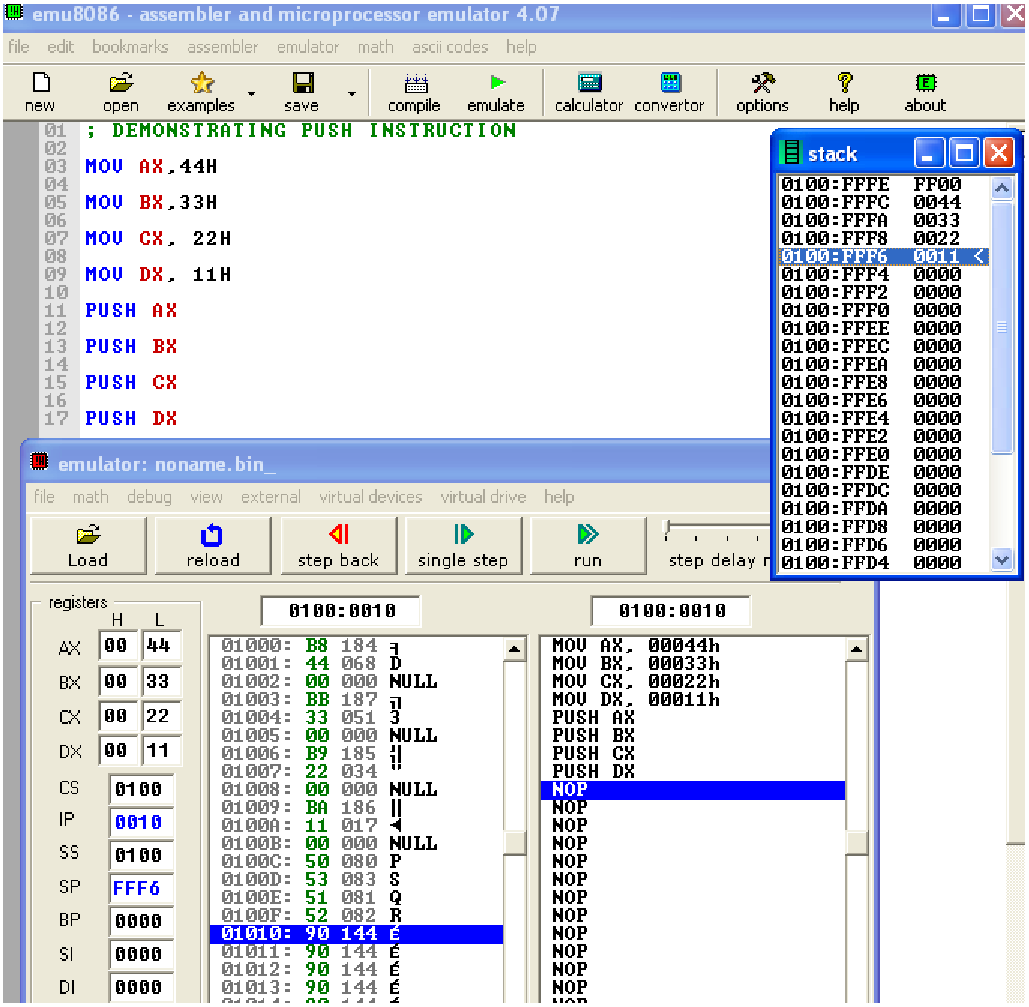
1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command Mov Ax, 44h
5. Write the command Mov Bx,33h
6. Write command Mov Cx, 22h
7. Write command mov Dx,11h
8. For transferring contents of these registers to stack write the commands as
9. Push Ax
10. Push Bx
11. Push cx
12. Push dx
13. Fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	SS	SP
1	MOV AX, 44H						
2	MOV BX,33H						
3	MOV CX,22H						
4	MOV DX,11H						
5	PUSH AX						
6	PUSH BX						
7	PUSH CX						
8	PUSH DX						


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay
stack
registers

```
; DEMONSTRATING PUSH INSTRUCTION

MOV AX,44H

MOV BX,33H

MOV CX, 22H

MOV DX, 11H

PUSH AX

PUSH BX

PUSH CX

PUSH DX
```

```
0100:FFFE   FF00
0100:FFFC   0044
0100:FFFA   0033
0100:FFF8   0022
0100:FFF6   0011 <
0100:FFF4   0000
0100:FFF2   0000
0100:FFF0   0000
0100:FFEE   0000
0100:FFEC   0000
0100:FFEA   0000
0100:FFE8   0000
0100:FFE6   0000
0100:FFE4   0000
0100:FFE2   0000
0100:FFE0   0000
0100:FFDE   0000
0100:FFDC   0000
0100:FFDA   0000
0100:FFD8   0000
0100:FFD6   0000
0100:FFD4   0000
```

	H	L
AX	00	44
BX	00	33
CX	00	22
DX	00	11

CS	0100
IP	0010
SS	0100
SP	FFF6
BP	0000
SI	0000
DI	0000

```
0100:0010

01000: B8 184 ╕
01001: 44 068 D
01002: 00 000 NULL
01003: BB 187 ╗
01004: 33 051 3
01005: 00 000 NULL
01006: B9 185 ╣
01007: 22 034 "
01008: 00 000 NULL
01009: BA 186 ║
0100A: 11 017 ◄
0100B: 00 000 NULL
0100C: 50 080 P
0100D: 53 083 S
0100E: 51 081 Q
0100F: 52 082 R
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
```

```
0100:0010

MOV AX, 00044h
MOV BX, 00033h
MOV CX, 00022h
MOV DX, 00011h
PUSH AX
PUSH BX
PUSH CX
PUSH DX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
```

PRACTICAL NO.17

DEMONSTATING POP INSTRUCTION SET

The general syntax for POP instruction set is

POP REGISTER

POP Instruction is used to copy contents from stack memory to register. In stack segment stack segment register (ss) and stack pointer register (sp) work together. POP instruction executes in following steps

1. Data is copied from stack memory to register
2. Stack Pointer is incremented by 2H

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command Mov Ax, 44h
5. Write the command Mov Bx,33h
6. Write command Mov Cx, 22h
7. Write command mov Dx,11h
8. For transferring contents of these registers to stack write the commands as
9. Push Ax
10. Push Bx
11. Push cx
12. Push dx

For copying data from stack to register write command as

13. Pop Ax
14. Pop Bx
15. Pop Cx
16. Pop Dx
17. Fill out the readings given for results

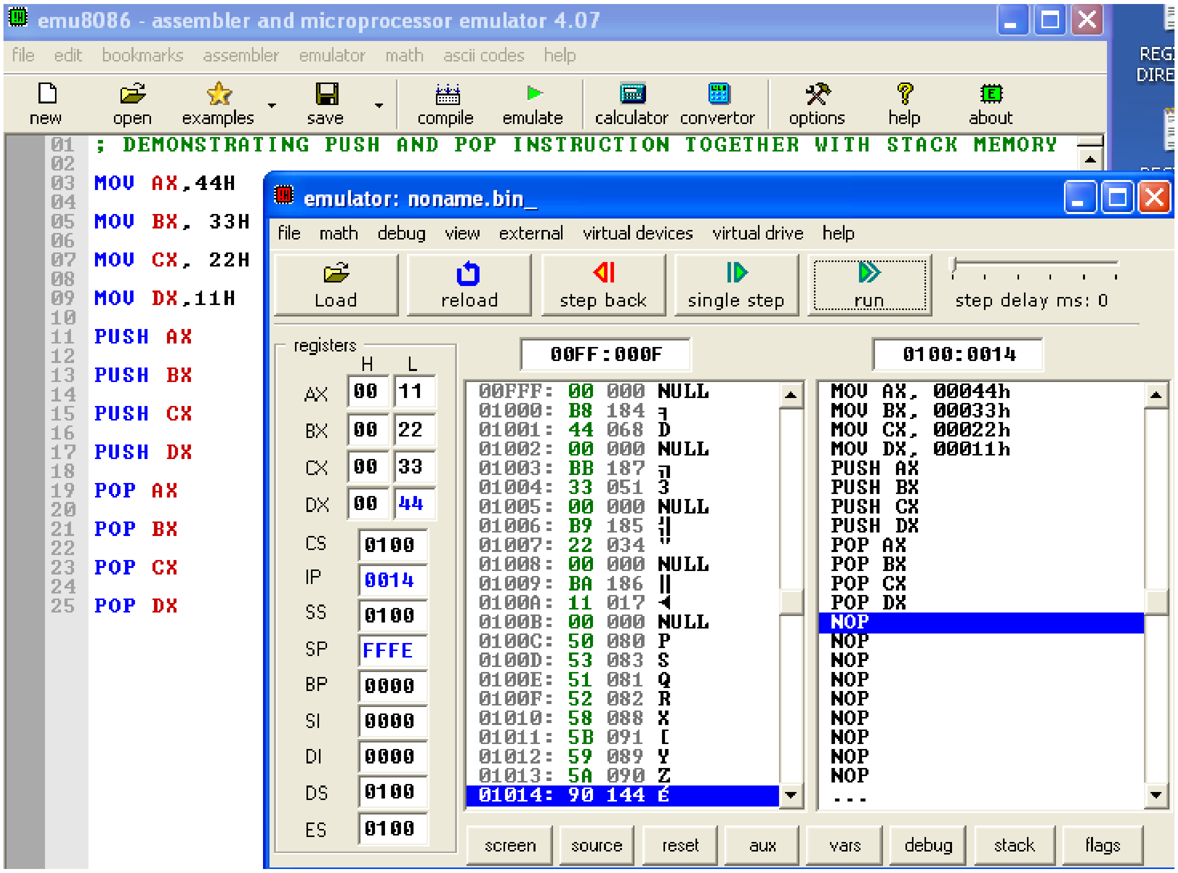
PRECAUSIONS:

1. Never use infinite loop in any coding.

2. Always emulate the code in single instruction

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	SS	SP
1	MOV AX, 44H						
2	MOV BX,33H						
3	MOV CX,22H						
4	MOV DX,11H						
5	PUSH AX						
6	PUSH BX						
7	PUSH CX						
8	PUSH DX						
9	Pop AX						
10	Pop BX						
11	POP CX						
12	POP DX						


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run
step delay ms: 0
screen source reset aux vars debug stack flags

```
; DEMONSTRATING PUSH AND POP INSTRUCTION TOGETHER WITH STACK MEMORY

MOV AX,44H

MOV BX, 33H

MOV CX, 22H

MOV DX,11H

PUSH AX

PUSH BX

PUSH CX

PUSH DX

POP AX

POP BX

POP CX

POP DX
```

registers

	H	L
AX	00	11
BX	00	22
CX	00	33
DX	00	44

CS	0100
IP	0014
SS	0100
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0100
ES	0100

```
00FF:000F

00FFF: 00 000 NULL
01000: B8 184 ╕
01001: 44 068 D
01002: 00 000 NULL
01003: BB 187 ╗
01004: 33 051 3
01005: 00 000 NULL
01006: B9 185 ╣
01007: 22 034 "
01008: 00 000 NULL
01009: BA 186 ║
0100A: 11 017 ◄
0100B: 00 000 NULL
0100C: 50 080 P
0100D: 53 083 S
0100E: 51 081 Q
0100F: 52 082 R
01010: 58 088 X
01011: 5B 091 [
01012: 59 089 Y
01013: 5A 090 Z
01014: 90 144 É
```

```
0100:0014

MOV  AX, 00044h
MOV  BX, 00033h
MOV  CX, 00022h
MOV  DX, 00011h
PUSH  AX
PUSH  BX
PUSH  CX
PUSH  DX
POP  AX
POP  BX
POP  CX
POP  DX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.18

DEMONSTATING PUSHF INSTRUCTION SET

The general syntax for PUSHF instruction set is

PUSHF

PUSHF Instruction is used to send the status of flag registers to stack memory. This command has no operands.

1. Stack pointer register (sp) is decremented by 2H
2. Data is copied from register to stack memory location.

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write the command as PUSHF.

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.
3. Care fully observes the output of registers.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	SS	SP
1	MOV AX,5H						
2	MOV BX,10H						
3.	Pushf						

PRACTICAL NO.19

DEMONSTATING POPF INSTRUCTION SET

The general syntax for POP instruction set is

POPF

POP Instruction is used to send status of flag registers from stack memory to flag register. This command has no operands

1. Data is copied from stack memory to register
2. Stack Pointer is incremented by 2H

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
3. Type the instruction on the coding area of simulator.
4. Write the instruction to move contents to registers
5. POPF
6. Fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	SS	SP
1	MOV AX,5H						
2	MOV BX,10H						
3.	Pushf						

PRACTICAL NO.20

DEMONSTATING ADD INSTRUCTION SET

The general syntax for ADD instruction set is

ADD REGISTER, REGISTER

ADD instruction is used to add the contents of two registers and result is stored into Ax register.

PROCEDURE:

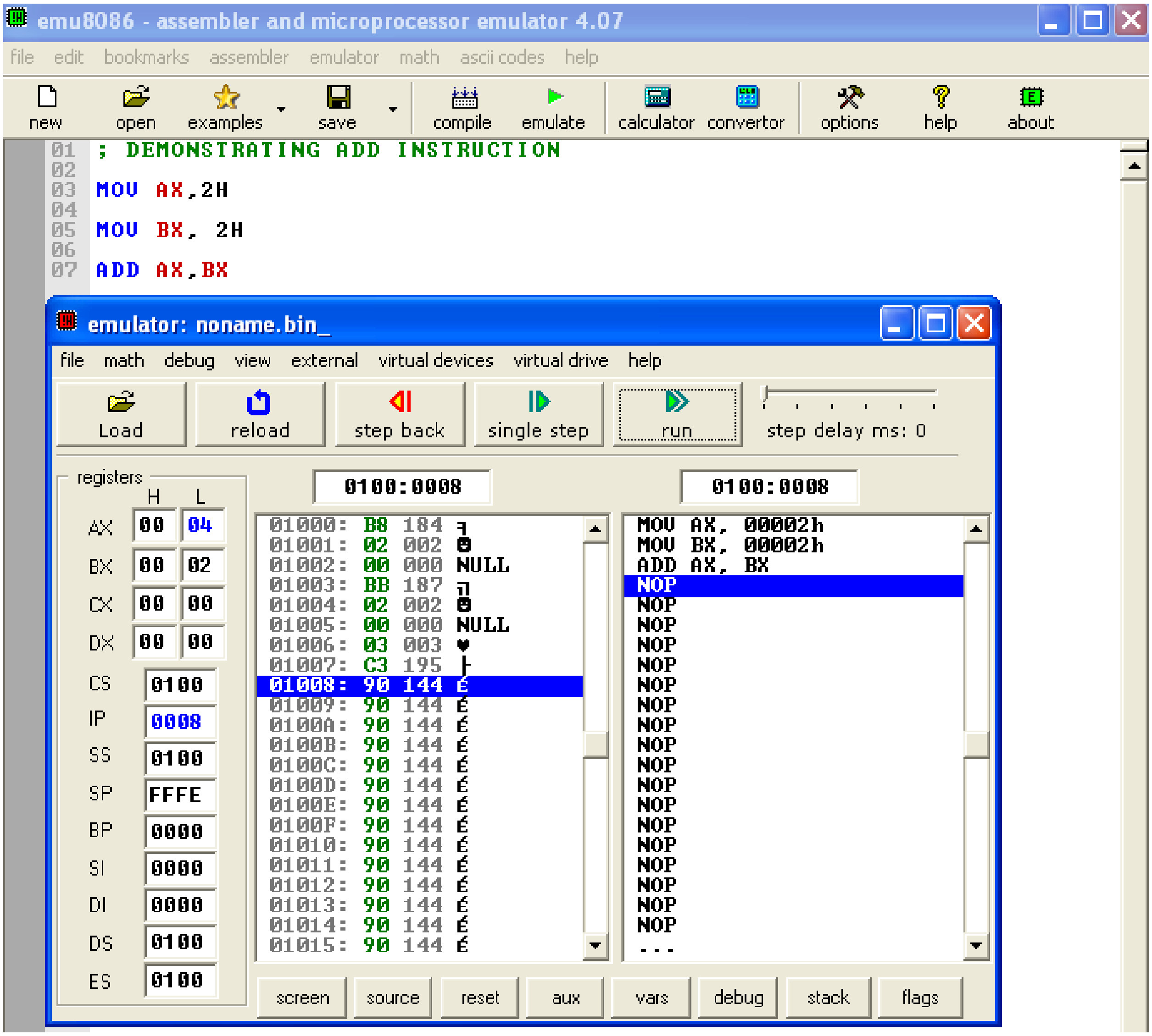
1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,2H
5. Write command MOV BX,2H
6. Write command ADD AX,BX
7. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	CS	IP
1	MOV AX,2H						
2	MOV BX,2H						
3.	ADD AX,BX						


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
; DEMONSTRATING ADD INSTRUCTION

MOV AX,2H

MOV BX, 2H

ADD AX,BX
```

registers

	H	L
AX	00	04
BX	00	02
CX	00	00
DX	00	00

CS	0100
IP	0008
SS	0100
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0100
ES	0100

```
0100:0008

01000: B8 184 ╕
01001: 02 002 ☻
01002: 00 000 NULL
01003: BB 187 ╗
01004: 02 002 ☻
01005: 00 000 NULL
01006: 03 003 ♥
01007: C3 195 ├
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008

MOV  AX, 00002h
MOV  BX, 00002h
ADD  AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.21

DEMONSTATING SUB INSTRUCTION SET

The general syntax for SUB instruction set is

SUB DESTINATION REGISTER, SOURCE REGISTER

SUB instruction is used to subtract the contents of two registers and result is stored into Ax register.

PROCEDURE:

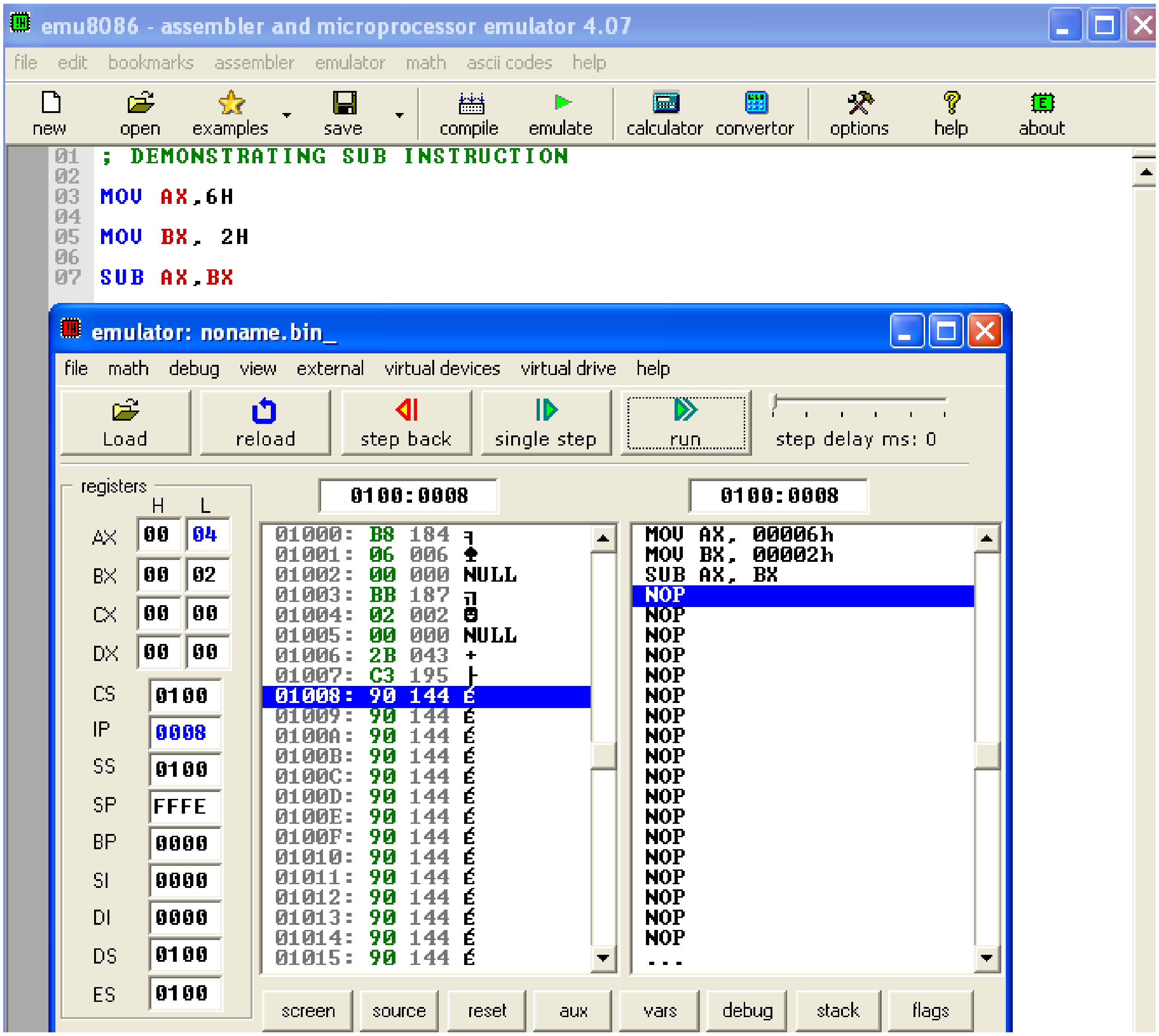
1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,6H
5. Write command MOV BX,2H
6. Write command SUB BX,AX
7. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	CS	IP
1	MOV AX,6H						
2	MOV BX,2H						
3.	SUB BX,AX						


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
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
screen source reset aux vars debug stack flags

```
; DEMONSTRATING SUB INSTRUCTION

MOV AX,6H

MOV BX, 2H

SUB AX,BX
```

registers	H	L
AX	00	04
BX	00	02
CX	00	00
DX	00	00
CS	0100	
IP	0008	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

```
0100:0008

01000: B8 184 ╕
01001: 06 006 ♠
01002: 00 000 NULL
01003: BB 187 ╗
01004: 02 002 ☻
01005: 00 000 NULL
01006: 2B 043 +
01007: C3 195 ├
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008

MOV  AX, 00006h
MOV  BX, 00002h
SUB  AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.22

DEMONSTATING MUL INSTRUCTION SET

The general syntax for MUL instruction set is

MUL REGISTER

MUL instruction is used to multiply contents of two registers. It uses only one register in the operand. For example if we want to multiply two values suppose 4H X 2H, 1st of all move any one value to AX register and 2nd value to any BX, CX or DX and apply command as MUL CX. The contents of AX register will be automatically multiplied by the contents of BX and result will be stored in AX.

PROCEDURE:

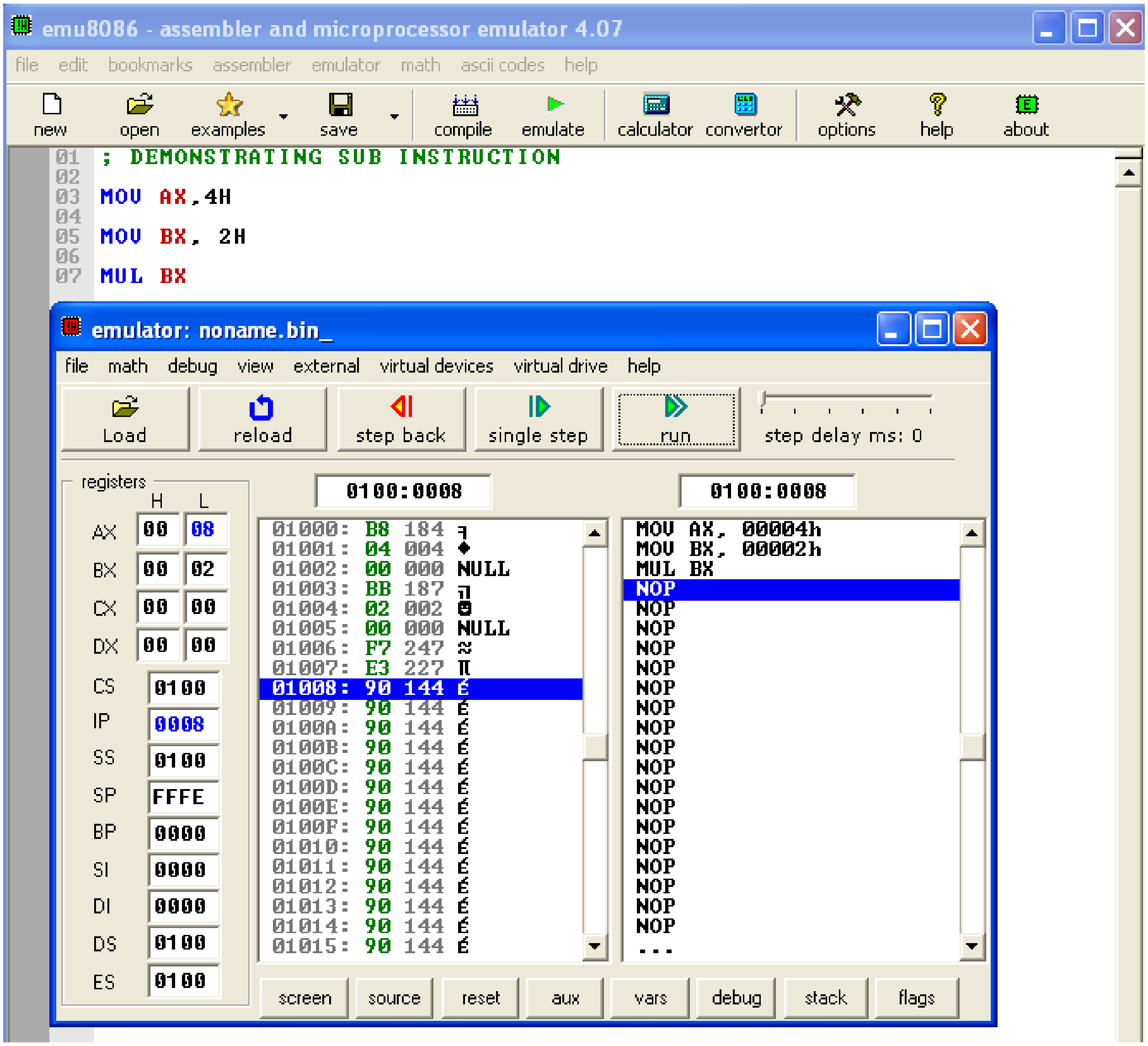
1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,4H
5. Write command MOV BX,2H
6. Write command MUL BX
7. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	CS	IP
1	MOV AX,4H						
2	MOV BX,2H						
3.	MUL BX						


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
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
screen source reset aux vars debug stack flags

```
; DEMONSTRATING SUB INSTRUCTION

MOV AX,4H

MOV BX, 2H

MUL BX
```

	H	L
AX	00	08
BX	00	02
CX	00	00
DX	00	00

CS	0100
IP	0008
SS	0100
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0100
ES	0100

```
0100:0008
01000: B8 184 ╕
01001: 04 004 ♦
01002: 00 000 NULL
01003: BB 187 ╗
01004: 02 002 ☻
01005: 00 000 NULL
01006: F7 247 ≈
01007: E3 227 π
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008
MOV AX, 00004h
MOV BX, 00002h
MUL BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.23

DEMONSTATING DIV INSTRUCTION SET

The general syntax for DIV instruction set is

DIV REGISTER

DIV instruction is used to divide contents of two registers. It uses only one register in the operand. For example if we want to divide two values suppose 5/2 , 1st of all move that number which will be divided (dividend) into AX register and that which will divide(divisor) into BX or CX. The quotient will be stored into AX and reminder will be stored into DX.

PROCEDURE:

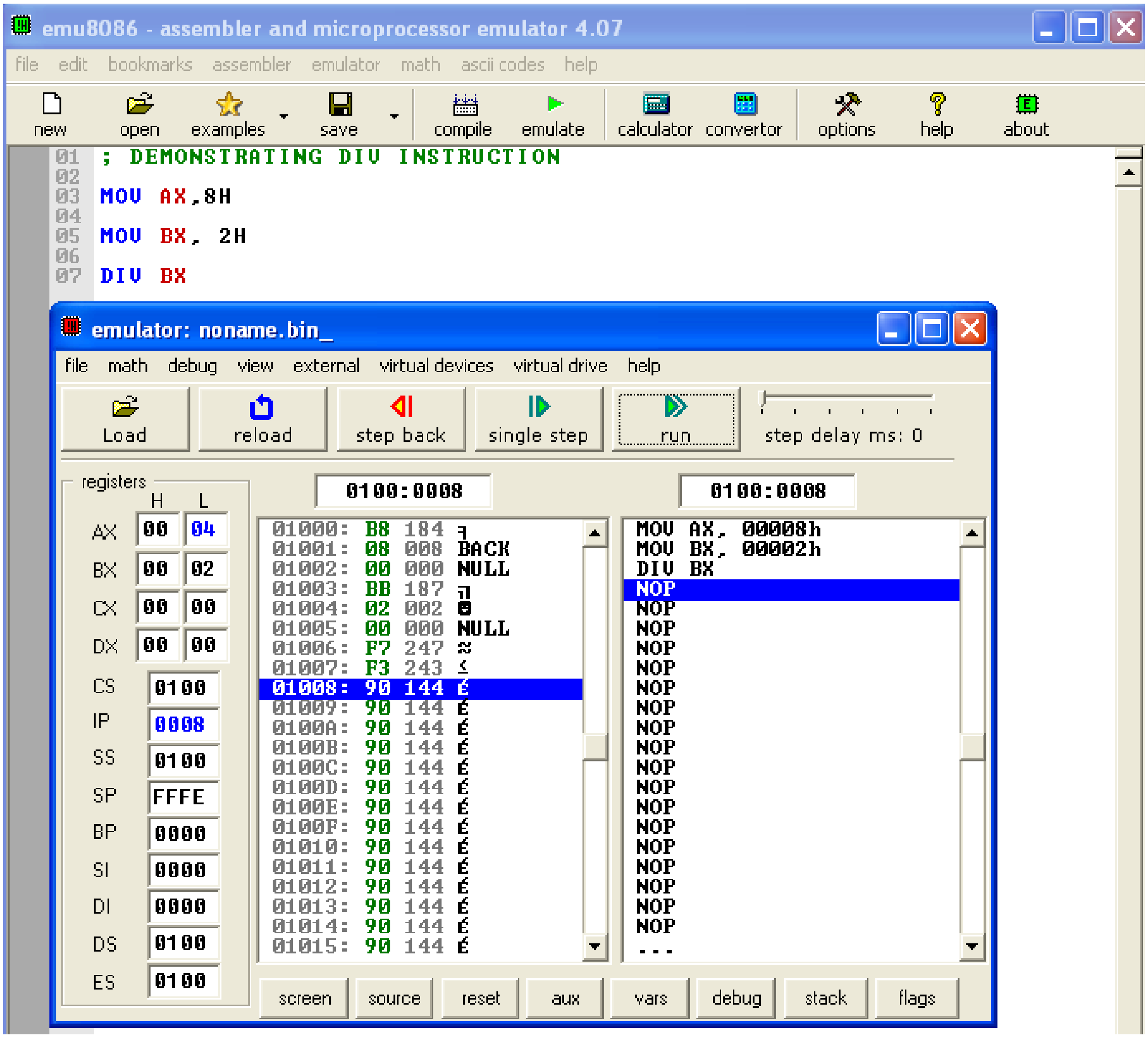
1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,8H
5. Write command MOV BX,2H
6. Write command DIV BX
7. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	CS	IP
1	MOV AX,8H						
2	MOV BX,2H						
3.	DIV BX						


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
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
screen source reset aux vars debug stack flags

```
; DEMONSTRATING DIV INSTRUCTION

MOV AX,8H

MOV BX, 2H

DIV BX
```

	H	L
AX	00	04
BX	00	02
CX	00	00
DX	00	00

CS	0100
IP	0008
SS	0100
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0100
ES	0100

```
0100:0008

01000: B8 184 ╕
01001: 08 008 BACK
01002: 00 000 NULL
01003: BB 187 ╗
01004: 02 002 ☻
01005: 00 000 NULL
01006: F7 247 ≈
01007: F3 243 ≤
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

```
0100:0008

MOV AX, 00008h
MOV BX, 00002h
DIV BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.24

DEMONSTATING INC INSTRUCTION SET

The general syntax for INC instruction set is

INC REGISTER

INC instruction increases the content of register or memory by 1H. Every single INC REGISTER increases the value of register by 1h.

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,8H
5. Write command MOV BX,2H
6. Write command INC AX
7. Write command INC AX
8. Write command INC BX
9. Write command INC BX
10. Write command INC CX
11. Write command INC DX
12. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	CS	IP
1	MOV AX,8H						
2	MOV BX,2H						
3.	INC AX						
4.	INC AX						
5.	INC BX						
6.	INC BX						
7.	INC CX						
8.	INC DX						

PRACTICAL NO.25

DEMONSTATING DEC INSTRUCTION SET

The general syntax for DEC instruction set is

DEC REGISTER

INC instruction decreases the content of register or memory by 1H. Every single INC REGISTER decreases the value of register by 1h.

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,8H
5. Write command MOV BX,2H
6. Write command DEC AX
7. Write command DEC AX
8. Write command DEC BX
9. Write command DEC BX
10. Write command DEC CX
11. Write command DEC DX
12. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction.

READINGS

S.NO	COMMANDS	RESULTS					
		AX	BX	CX	DX	CS	IP
1	MOV AX,8H						
2	MOV BX,2H						
3.	DEC AX						
4.	DEC AX						
5.	DEC BX						
6.	DEC BX						
7.	DEC CX						
8.	DEC DX						

PRACTICAL NO.26

DEMONSTATING NOT INSTRUCTION SET

The general syntax for NOT instruction set is

NOT REGISTER

NOT instruction fall into category of bit-manipulation instruction set. This command is used to take complement of binary bits.

PROCEDURE:

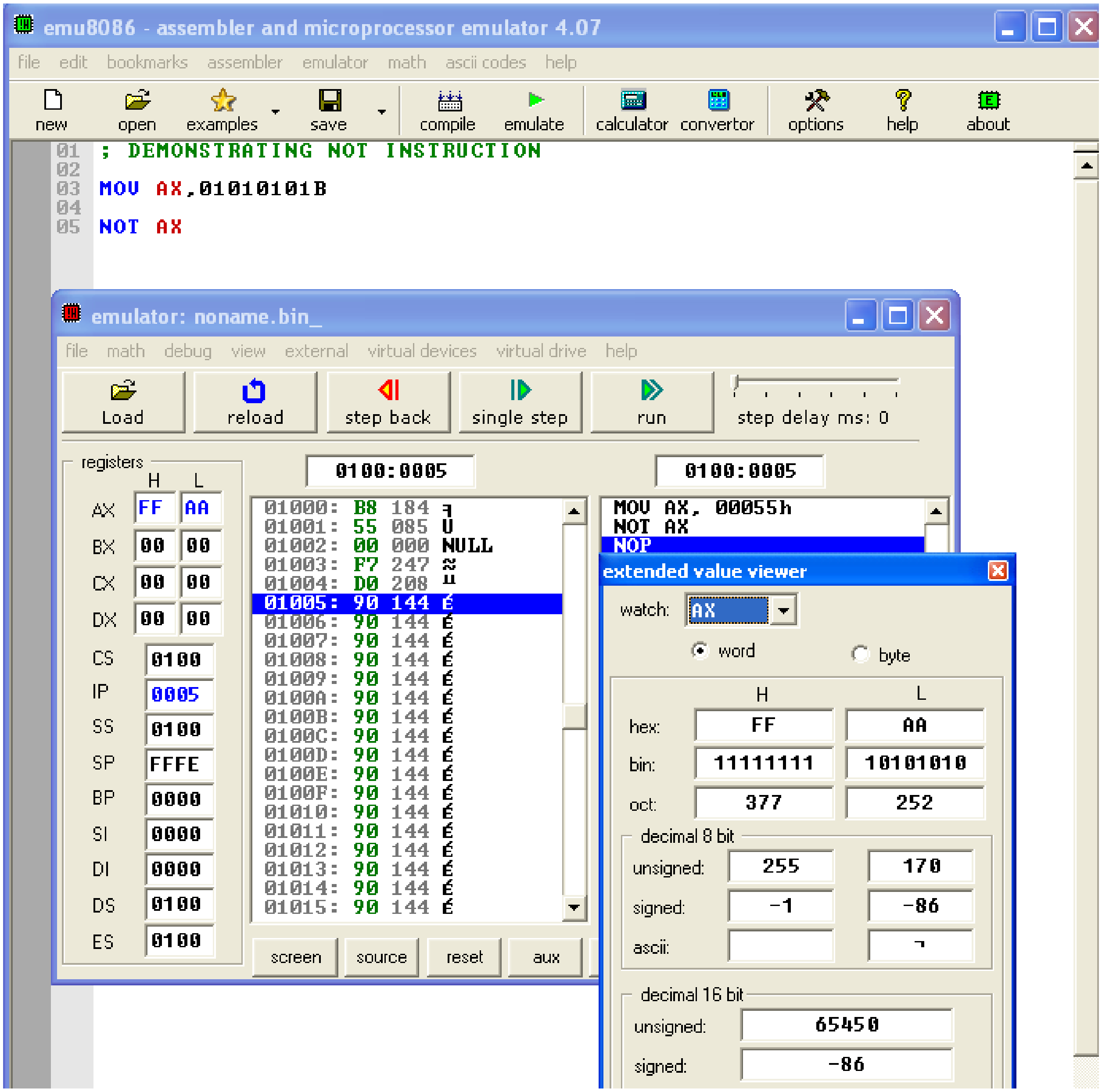
1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,01010101b
5. Write command NOT AX
6. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction

READINGS

S.NO	COMMANDS	RESULTS		
		AX	CS	IP
1	MOV AX,01010101B	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1		
2	NOT AX	1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0		


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
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
screen source reset aux
extended value viewer

```
; DEMONSTRATING NOT INSTRUCTION

MOV AX,01010101B

NOT AX
```

	H	L
AX	FF	AA
BX	00	00
CX	00	00
DX	00	00

CS 0100
IP 0005
SS 0100
SP FFFE
BP 0000
SI 0000
DI 0000
DS 0100
ES 0100

0100:0005

```
01000: B8 184 ╕
01001: 55 085 U
01002: 00 000 NULL
01003: F7 247 ≈
01004: D0 208 ╨
01005: 90 144 É
01006: 90 144 É
01007: 90 144 É
01008: 90 144 É
01009: 90 144 É
0100A: 90 144 É
0100B: 90 144 É
0100C: 90 144 É
0100D: 90 144 É
0100E: 90 144 É
0100F: 90 144 É
01010: 90 144 É
01011: 90 144 É
01012: 90 144 É
01013: 90 144 É
01014: 90 144 É
01015: 90 144 É
```

0100:0005

```
MOV AX, 00055h
NOT AX
NOP
```

watch: AX
● word ○ byte

	H	L
hex:	FF	AA
bin:	11111111	10101010
oct:	377	252

decimal 8 bit

unsigned:	255	170
signed:	-1	-86
ascii:		¬

decimal 16 bit
unsigned: 65450
signed: -86

PRACTICAL NO.27

DEMONSTATING AND INSTRUCTION SET

The general syntax for AND instruction set is

AND REGISTER, REGISTER

AND instruction is used to manipulate and logic in microprocessor.

PROCEDURE:

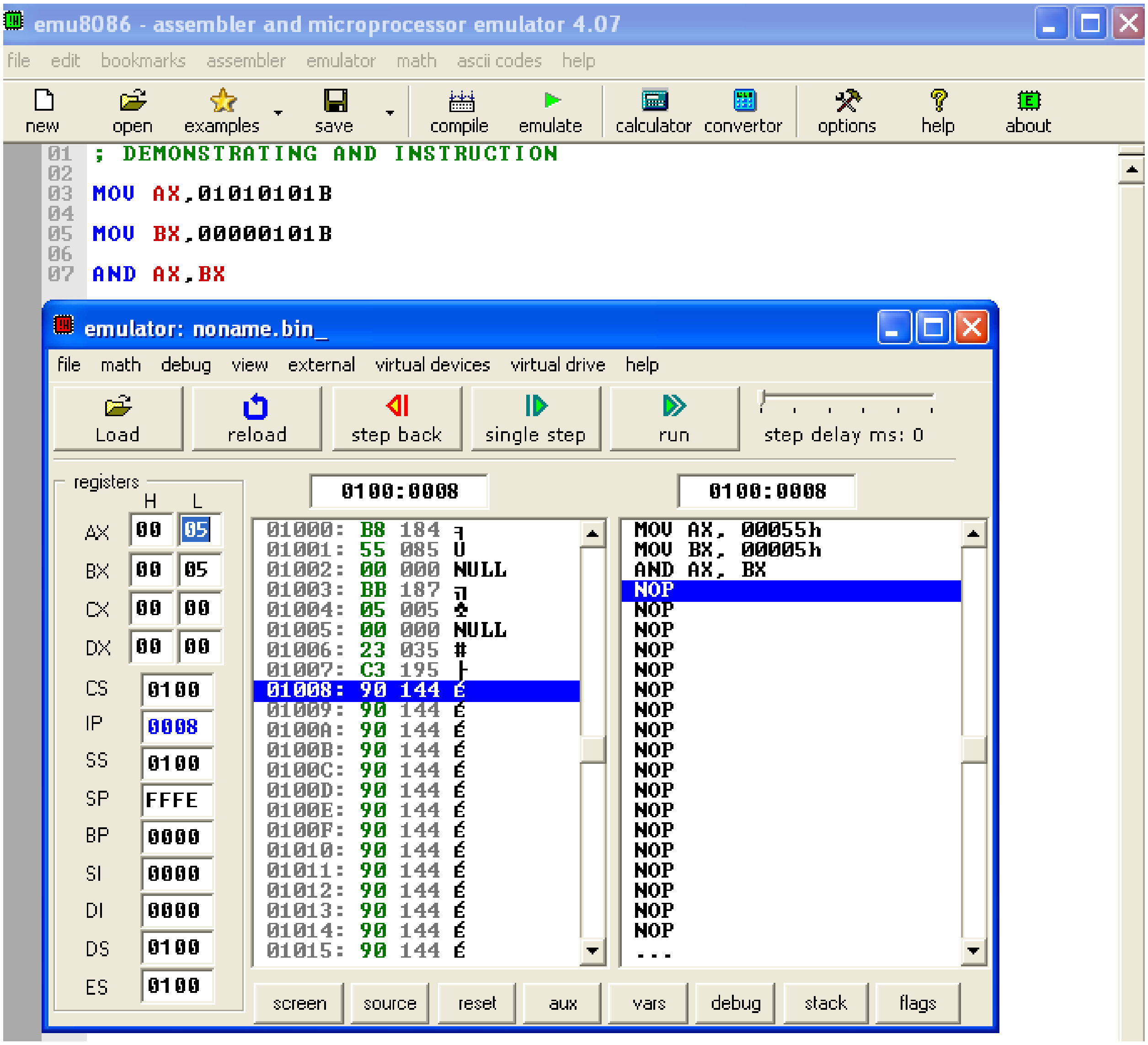
1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,01010101b
5. Write command MOV BX,00001010b
6. Write command AND AX,BX
7. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction

READINGS

S.NO	COMMANDS	RESULTS		
		AX	CS	IP
1	MOV AX,01010101B	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1		
2	MOV BX,00000101B	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1		
3	AND AX,BX	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1		


emu8086 - assembler and microprocessor emulator 4.07
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.bin_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
; DEMONSTRATING AND INSTRUCTION

MOV AX,01010101B

MOV BX,00000101B

AND AX,BX
```

registers

```
     H   L
AX   00  05
BX   00  05
CX   00  00
DX   00  00
CS   0100
IP   0008
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
0100:0008

01000:  B8 184 ╕
01001:  55 085 U
01002:  00 000 NULL
01003:  BB 187 ╗
01004:  05 005 ♣
01005:  00 000 NULL
01006:  23 035 #
01007:  C3 195 ├
01008:  90 144 É
01009:  90 144 É
0100A:  90 144 É
0100B:  90 144 É
0100C:  90 144 É
0100D:  90 144 É
0100E:  90 144 É
0100F:  90 144 É
01010:  90 144 É
01011:  90 144 É
01012:  90 144 É
01013:  90 144 É
01014:  90 144 É
01015:  90 144 É
```

```
0100:0008

MOV AX, 00055h
MOV BX, 00005h
AND AX, BX
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

PRACTICAL NO.28

DEMONSTATING OR INSTRUCTION SET

The general syntax for OR instruction set is

OR REGISTER, REGISTER

OR instruction is used to manipulate OR logic in microprocessor.

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,01010101b
5. Write command MOV BX,00001010b
6. Write command OR AX,BX
7. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction

READINGS

S.NO	COMMANDS	RESULTS		
		AX	CS	IP
1	MOV AX,01010101B	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1		
2	MOV BX,00000101B	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1		
3	OR AX,BX	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1		

PRACTICAL NO.29

DEMONSTATING XOR INSTRUCTION SET

The general syntax for XOR instruction set is

XOR REGISTER, REGISTER

XOR instruction is used to manipulate XOR logic in microprocessor.

PROCEDURE:

1. Open emu-8086 simulator and select empty work space from option.
2. Type the instruction on the coding area of simulator.
3. Write the instruction to move contents to registers
4. Write command MOV AX,01010101b
5. Write command MOV BX,00001010b
6. Write command XOR AX,BX
7. Observe output and fill out the readings given for results

PRECAUSIONS:

1. Never use infinite loop in any coding.
2. Always emulate the code in single instruction

READINGS

S.NO	COMMANDS	RESULTS		
		AX	CS	IP
1	MOV AX,01010101B	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1		
2	MOV BX,00000101B	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1		
3	XOR AX,BX	0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0		