Microprocessor and Interfacing Lab Experiment 2

Name: Om Ashish Mishra
Registration Number: 16BCE0789
Slot: B2

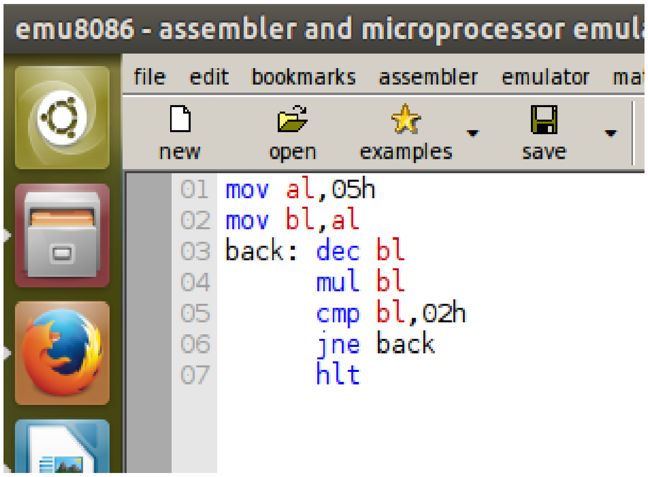
1. Write an ALP to find the factorial of a number with and without using loop instruction

LOOP

Factorial

Write a program to find the factorial of a number with /without using loop instruction.

Without Loop



```
mov al,05h
mov bl,al
back: dec bl
      mul bl
      cmp bl,02h
      jne back
      hlt
```

Aim: Factorial without loop

Algorithm:

Step 1: First we store the value of the 5 in al.

Step 2: Then we store the value of al in bl which will calculate the factorail

Step 3: Then we decrease the value of bl.

Step 4: The multiplication takes place between bl prevoius value and bl.

Step 5: Compare the new value with 02h

Step 6: if not euql then we back to dec bl and to multiply with bl.

Step 7: else we got to hlt

Step 8: We get the result.

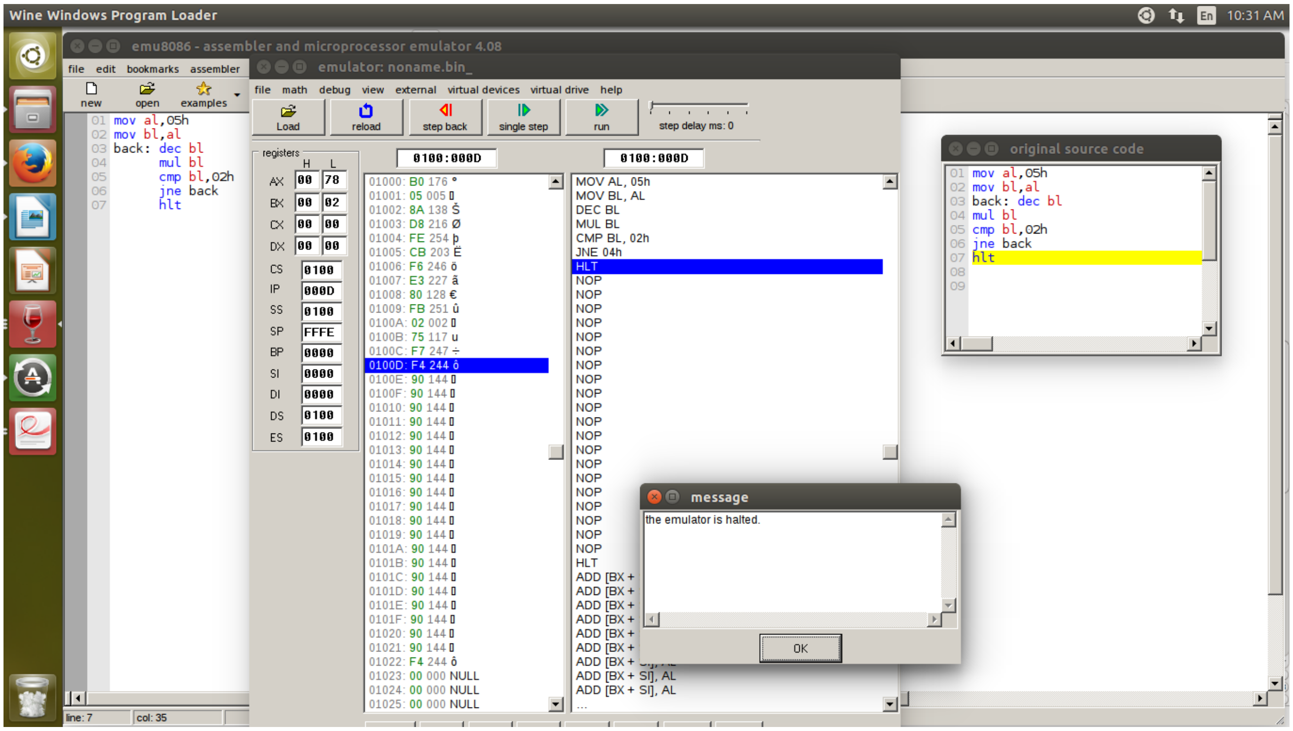
Memory Address	Instruction	Hex code
1000 1001	Mov al,05h	B0,05
1002 1003	Mov bl,al	8A,D8
1004 1005	Back: dec bl	FE,CB
1006 1007	Mul bl	F6,E3
1008 1009 100A	Cmp bl,02h	80,FB,02
100B 100C	Jne back	75,F7
100D	hlt	F4

Sample Input: 5h

Sample Output: 78h

Result:

The result of factorial is 78h.



```
mov al,05h
mov bl,al
back: dec bl
      mul bl
      cmp bl,02h
      jne back
      hlt
```

registers	H	L
AX	00	78
BX	00	02
CX	00	00
DX	00	00
CS	0100	
IP	000D	
SS	0100	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0100	
ES	0100	

```
0100:000D
01000: B0 176 °
01001: 05 005 ▯
01002: 8A 138 Š
01003: D8 216 Ø
01004: FE 254 þ
01005: CB 203 Ë
01006: F6 246 ö
01007: E3 227 ã
01008: 80 128 €
01009: FB 251 û
0100A: 02 002 ▯
0100B: 75 117 u
0100C: F7 247 ÷
0100D: F4 244 ô
0100E: 90 144 ▯
0100F: 90 144 ▯
01010: 90 144 ▯
01011: 90 144 ▯
01012: 90 144 ▯
01013: 90 144 ▯
01014: 90 144 ▯
01015: 90 144 ▯
01016: 90 144 ▯
01017: 90 144 ▯
01018: 90 144 ▯
01019: 90 144 ▯
0101A: 90 144 ▯
0101B: 90 144 ▯
0101C: 90 144 ▯
0101D: 90 144 ▯
0101E: 90 144 ▯
0101F: 90 144 ▯
01020: 90 144 ▯
01021: 90 144 ▯
01022: F4 244 ô
01023: 00 000 NULL
01024: 00 000 NULL
01025: 00 000 NULL
```

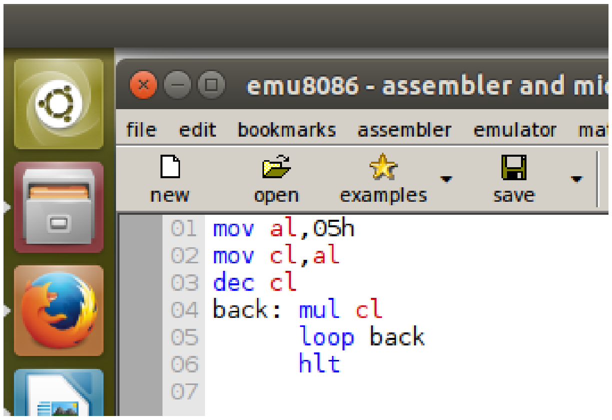
```
0100:000D
MOV AL, 05h
MOV BL, AL
DEC BL
MUL BL
CMP BL, 02h
JNE 04h
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
ADD [BX +
ADD [BX +
ADD [BX +
ADD [BX +
ADD [BX +
ADD [BX +
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

original source code

```
mov al,05h
mov bl,al
back: dec bl
mul bl
cmp bl,02h
jne back
hlt
```

the emulator is halted.

With Loop



```
mov al,05h
mov cl,al
dec cl
back: mul cl
      loop back
      hlt
```

Aim: Factorial with loop

Algorithm:

Step 1: First we store the value of the 5 in al.

Step 2: Then we store the value of al in cl which will calculate the factorial

Step 3: Then we decrease the value of cl.

Step 4: The multiplication takes place between cl prevoius value and cl.

Step 5: We continue to loop in 01h

Step 8: We get the result.

Memory Address	Instruction	Hex code
1000 1001	Mov al,05h	B0,05
1002 1003	Mov cl,al	8A,C8
1004 1005	dec cl	FE,C9
1006 1007	Back: mul cl	F6,E1
1008 1009	Loop back	E2,FC
0110A	hlt	F4

Sample Input: 5

Sample Output: 78h

The Result:

The result of factorial is 78h.

2. Write ALP to Check whether the given number is prime or not.

Aim: Prime Number or not

Algorithm:
Step 1: First we ask the user to choose a number

Step 2: Then we store the data in data segement which is transferred it to accumulator in code segment

Step 3: Then we display the number.

Step 4: Then we run a loop till of the number to check whether it is division by the any of the numbers in the loop

Step 5: If it is divisible it is printed as not a print number

Step 6: Else it is printed as a prime number.

Step 7: We got to hlt

Step 8: We get the result.

Memory Address	Instructions	Hex Code
1000 1001	.data	B0,05
1002 1003	AHSAN DB "ENTER NUMBER BETWEEN 0 & 9 ::: $"	8A,C8
1004 1005	AHSAN1 DB 0DH,0AH,"IT IS PRIME NUMBER $"	FE,C9
1006 1007	AHSAN2 DB 0DH,0AH,"IT IS NOT PRIME NUMBER $"	F6,E1
1008 1009 100A	.CODE	E2,FC
100B 100C	LEA DX,AHSAN	F4, B4
100D	MOV AH,09H	B0
100E 100F	INT 21H	8A,C8
1010 1011	MOV AH,01H	FE,C9
1012 1013	INT 21H	F6,E1
1014 1015	SUB AL,30H	E2,FC
1016 1017 1018	MOV BL,AL	F4,1E,A8
1019 101A	CMP BL,02H	B0,05
101B	JE PRIME	8A
101C 101D	CMP BL,03H	FE,C9
101E	JE PRIME	F6
101F 1020	CMP BL,05H	E2,FC
1021	JE PRIME	F4
1022 1023	CMP BL,07H	B0,05
1024	JE PRIME	8A
1025	JMP NOTPRIME	FE
1026 1027	PRIME:	F6,E1
1026 1027	LEA DX,AHSAN1	F6,E1
1028 1029	MOV AH,09H	F4,D5
102A 102B	INT 21H	B0,05
102C	JMP PEND	8A
102D 102E	NOTPRIME:	FE,C9
102D 102E	LEA DX,AHSAN2	FE,C9
102F 1030	MOV AH,09H	E2,FC
1031	INT 21H	F4
1032	JMP PEND	B0
1033 1034	PEND:	8A,C8
1033 1034	MOV AH,4CH	FE,C9

1035 1036	INT 21H	F6,E1
1037	RET	E2

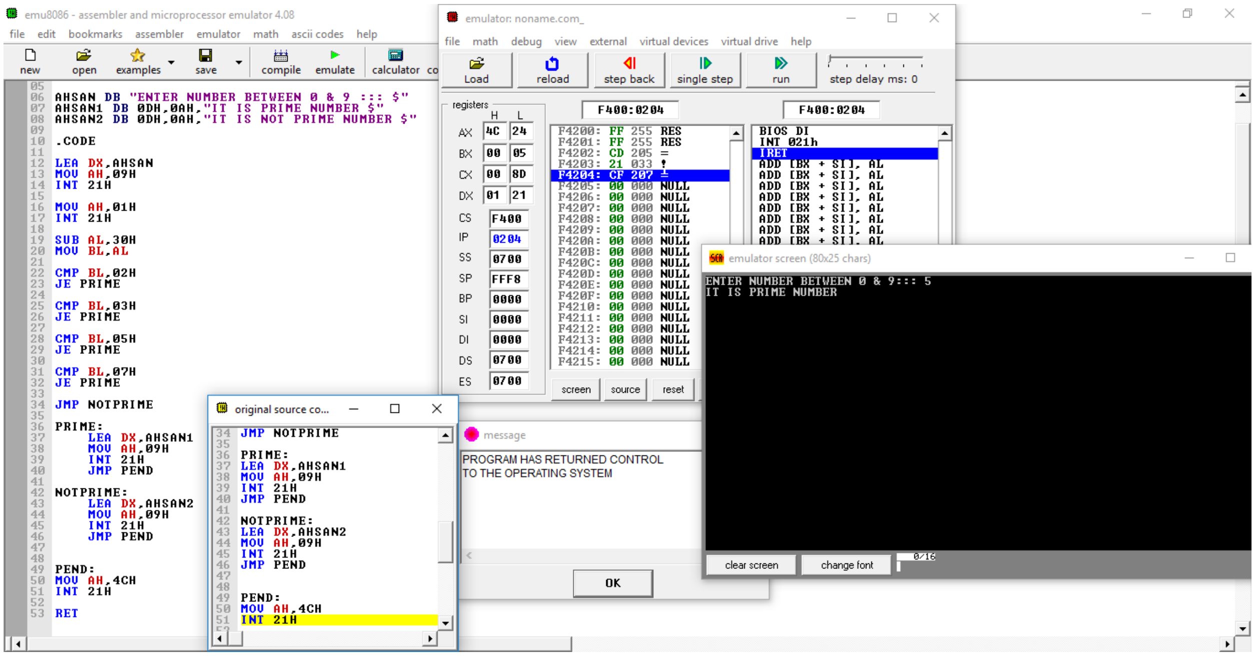
Sample Input: 5

Sample Output: Prime

Result:

Enter a number between 0 and 9: 5

It is a prime number.


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator
emulator: noname.com_
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
original source co...
message
emulator screen (80x25 chars)
clear screen change font

```
AHSAN DB "ENTER NUMBER BETWEEN 0 & 9 ::: $"
AHSAN1 DB 0DH,0AH,"IT IS PRIME NUMBER $"
AHSAN2 DB 0DH,0AH,"IT IS NOT PRIME NUMBER $"

.CODE

LEA DX,AHSAN
MOV AH,09H
INT 21H

MOV AH,01H
INT 21H

SUB AL,30H
MOV BL,AL

CMP BL,02H
JE PRIME

CMP BL,03H
JE PRIME

CMP BL,05H
JE PRIME

CMP BL,07H
JE PRIME

JMP NOTPRIME

PRIME:
     LEA DX,AHSAN1
     MOV AH,09H
     INT 21H
     JMP PEND

NOTPRIME:
     LEA DX,AHSAN2
     MOV AH,09H
     INT 21H
     JMP PEND


PEND:
MOV AH,4CH
INT 21H

RET
```

```
    H  L
AX  4C 24
BX  00 05
CX  00 8D
DX  01 21
CS  F400
IP  0204
SS  0700
SP  FFF8
BP  0000
SI  0000
DI  0000
DS  0700
ES  0700
```

```
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ±
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F4215: 00 000 NULL
```

```
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
```

```
ENTER NUMBER BETWEEN 0 & 9::: 5
IT IS PRIME NUMBER
```

PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM

OK

3. Write ALP to print the multiplication table for number 07.

Aim: Multiplication of table 7

Algorithm:
Step 1: First we store the value of the 7 in al.

Step 2: Then we store value of al in bl

Step 3: Then we store value of 01h in cl which is the counter.

Step 4: We call si(stack Initiator) to store the value of address 2000h.

Step 5: Now the looping starts and we multiply cl's value with bl.

Step 6: increase the value of the address location

Step 7: Refresh the value of the al from bl

Step 8: increase the counter value

Step 9: compare for equal to 10h

Step 10: if yes we hlt and get the result.

Step 11: else we go back into the loop

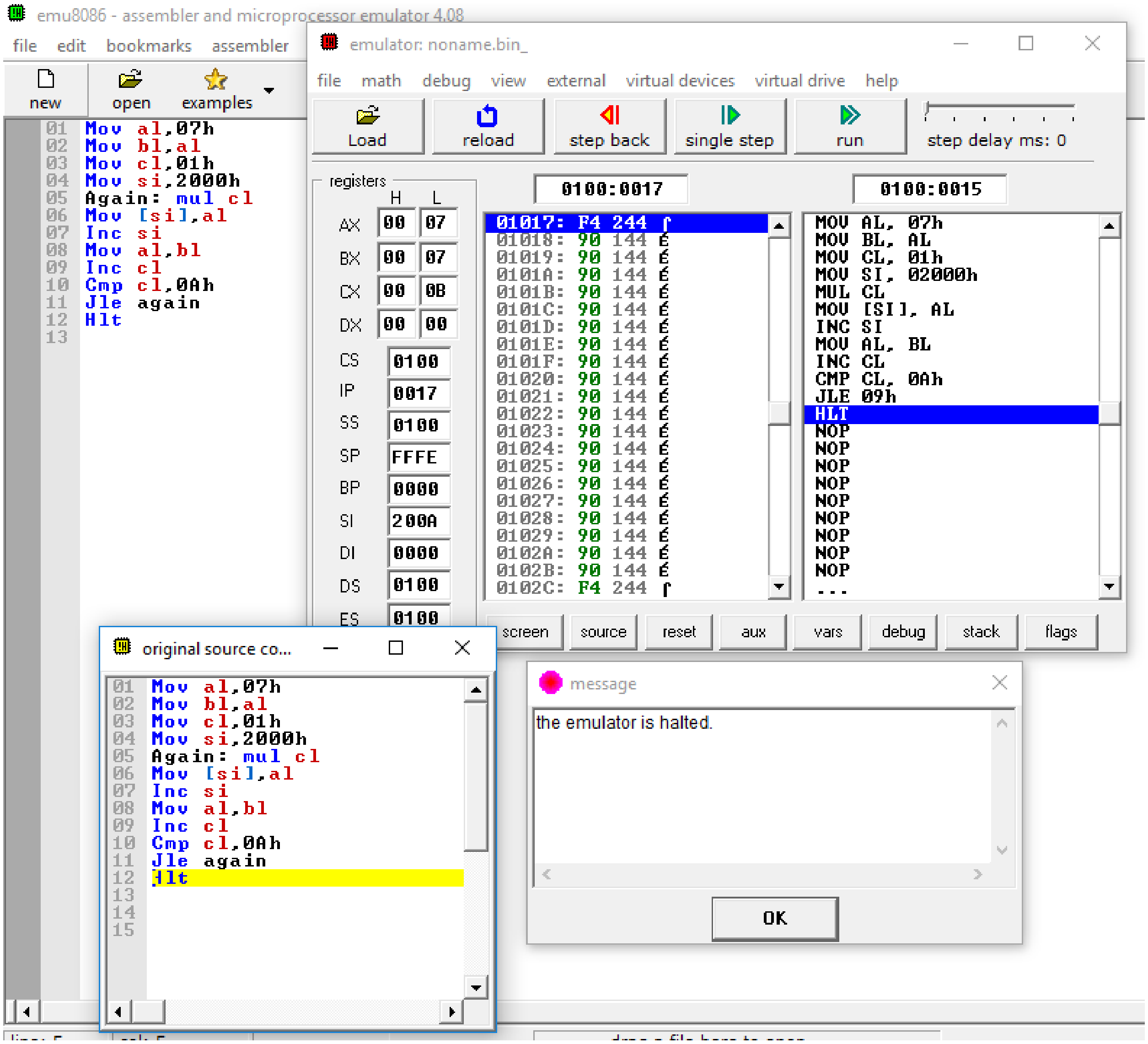
Memory Address	Instruction	Hex code
1000 1001	Mov al,07h	B0,05
1002 1003	Mov bl,al	8A,D8
1004 1005	Mov cl,01h	B1,01
1006 1007 1008	Mov si,2000h	BE,00,20
1009 100A	Agian: mul cl	F6,E1
0100B 0100C	Mov [si],al	88,04
0100D	Inc si	46
0100E 0100F	Mov al,bl	8A,C3
01010 01011	Inc cl	FE,C1
01012 01013 01014	Cmp cl,0Ah	80,F9,0A
01015 01016	Jle, again	7E,F2
01017	Hlt	F4

Sample Input: 7

Sample Output: Table of 7

Result:

The table represents multiplication of 7 from 1 to 10.



```
Mov al,07h
Mov bl,al
Mov cl,01h
Mov si,2000h
Again: mul cl
Mov [si],al
Inc si
Mov al,bl
Inc cl
Cmp cl,0Ah
Jle again
Hlt
```

	H	L
AX	00	07
BX	00	07
CX	00	0B
DX	00	00
CS	0100	
IP	0017	
SS	0100	
SP	FFFE	
BP	0000	
SI	200A	
DI	0000	
DS	0100	
ES	0100	

```
0100:0017
01017: F4 244 ⌠
01018: 90 144 É
01019: 90 144 É
0101A: 90 144 É
0101B: 90 144 É
0101C: 90 144 É
0101D: 90 144 É
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: F4 244 ⌠
```

```
0100:0015
MOV AL, 07h
MOV BL, AL
MOV CL, 01h
MOV SI, 02000h
MUL CL
MOV [SI], AL
INC SI
MOV AL, BL
INC CL
CMP CL, 0Ah
JLE 09h
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

the emulator is halted.

4. Write ALP to generate the Fibonacci series for N terms.

Aim: Fibonacci series

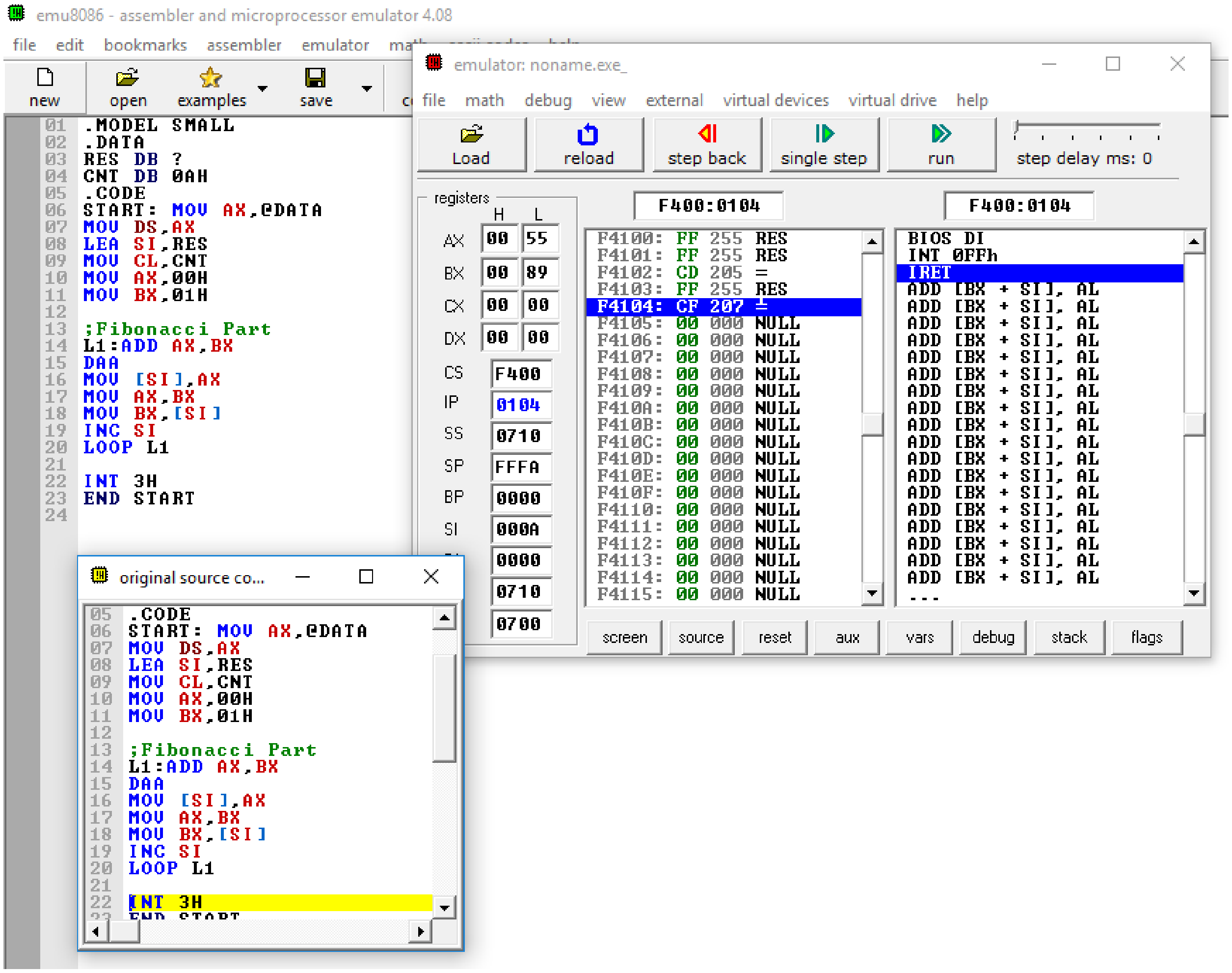
Algorithm:
Step 1: First we store the value of 1 and 0 in data segment and counter register is created here its value is 10.

Step 2: Then we store it in accumulator in code segment and do addition in loop for Fibonacci.

Step 3: First we run the loop and keep on adding and swapping the positions of numbers till we reach counter as 0.

Step 4: Show the output and stop.

Memory Address	Instructions	Hex Code
1000 1001	.MODEL SMALL	B0,05
1002 1003	.DATA	8A,D8
1004 1005	RES DB ?	B1,01
1006 1007 1008	CNT DB 0AH	BE,00,20
1009 100A	.CODE	F6,E1
0100B 0100C 0100D	START: MOV AX,@DATA	88,04
0100E 0100F	MOV DS,AX	46,C8
01010 01011	LEA SI,RES	8A,C3
01012 01013	MOV CL,CNT	FE,C1
01014 01015	MOV AX,00H	80,F9
01016 01017	MOV BX,01H	7E,F2
01018 01019	L1:ADD AX,BX	F4,C7
0101A	DAA	B0
0101B 0101C	MOV [SI],AX	8A,D8
0101D 0101E	MOV AX,BX	B1,01
0101F 01020	MOV BX,[SI]	BE,00
01021	INC SI	F6
01022	LOOP L1	88
01023	INT 3H	4B
01024	END START	8A


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save
emulator: noname.exe_
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
original source co...

```
.MODEL SMALL
.DATA
RES DB ?
CNT DB 0AH
.CODE
START: MOV AX,@DATA
MOV DS,AX
LEA SI,RES
MOV CL,CNT
MOV AX,00H
MOV BX,01H

;Fibonacci Part
L1:ADD AX,BX
DAA
MOV [SI],AX
MOV AX,BX
MOV BX,[SI]
INC SI
LOOP L1

INT 3H
END START
```

```
    H  L
AX  00 55
BX  00 89
CX  00 00
DX  00 00
CS  F400
IP  0104
SS  0710
SP  FFFA
BP  0000
SI  000A
    0000
    0710
    0700
```

```
F400:0104

F4100: FF 255 RES
F4101: FF 255 RES
F4102: CD 205 =
F4103: FF 255 RES
F4104: CF 207 ±
F4105: 00 000 NULL
F4106: 00 000 NULL
F4107: 00 000 NULL
F4108: 00 000 NULL
F4109: 00 000 NULL
F410A: 00 000 NULL
F410B: 00 000 NULL
F410C: 00 000 NULL
F410D: 00 000 NULL
F410E: 00 000 NULL
F410F: 00 000 NULL
F4110: 00 000 NULL
F4111: 00 000 NULL
F4112: 00 000 NULL
F4113: 00 000 NULL
F4114: 00 000 NULL
F4115: 00 000 NULL
```

```
F400:0104

BIOS DI
INT 0FFh
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

```
.CODE
START: MOV AX,@DATA
MOV DS,AX
LEA SI,RES
MOV CL,CNT
MOV AX,00H
MOV BX,01H

;Fibonacci Part
L1:ADD AX,BX
DAA
MOV [SI],AX
MOV AX,BX
MOV BX,[SI]
INC SI
LOOP L1

INT 3H
END START
```

Sample Input: 10

Sample Output: 55h

Result:

The sum of Fibonacci is 55h (88 in decimal).