Repository:

Assembly-Language-927

Name:

Junaid Ahmad

Assignment#1

Technical Example of

A) Input

B) Process

C) Output

Technical Example of

A) Input

Takes single character input from user

Stores input in BL register

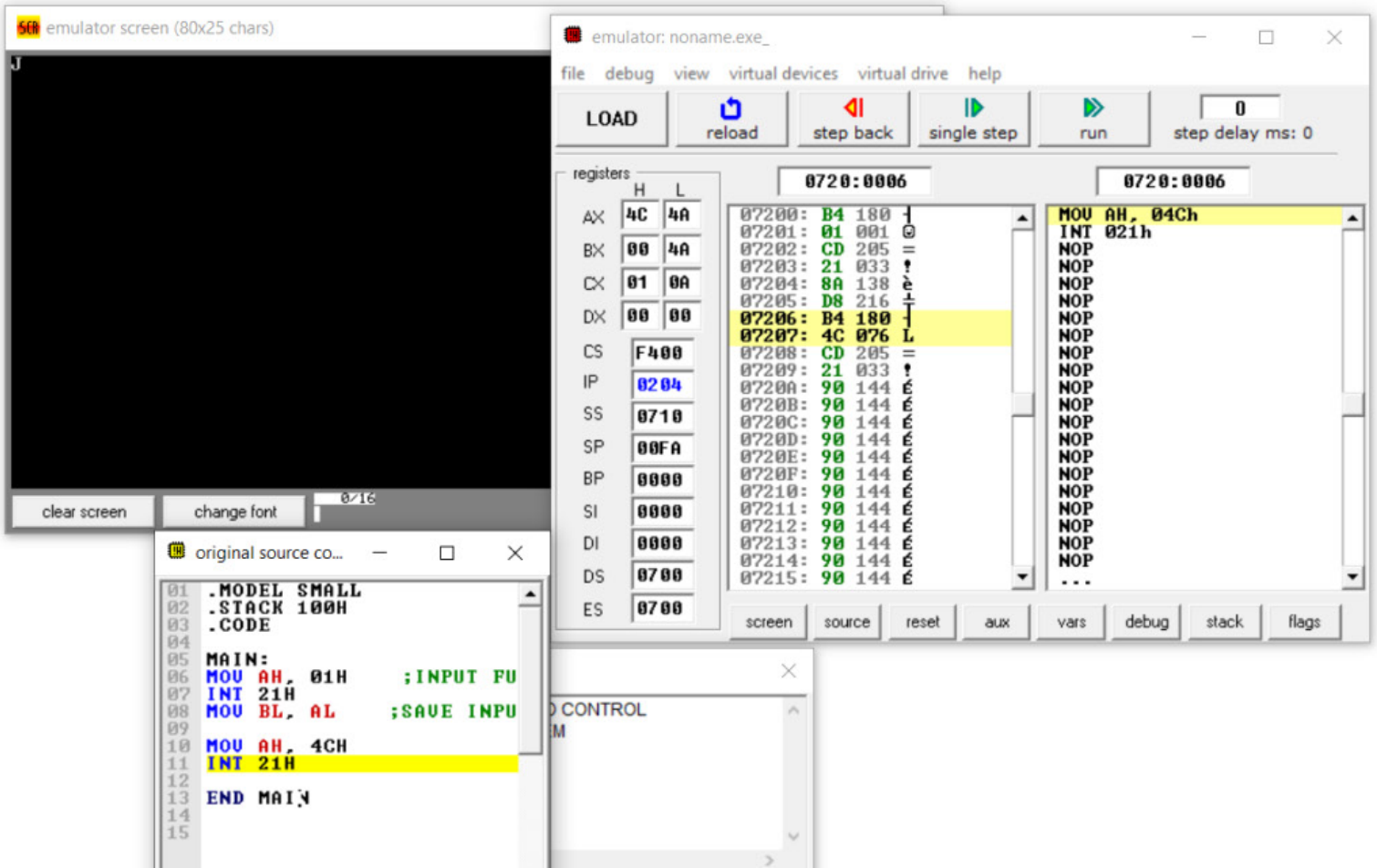
Exits program after input

```
.MODEL SMALL
.STACK 100H
.CODE

MAIN:
    MOV AH, 01H ;INPUT FUNCTION
    INT 21H
    MOV BL, AL  ;SAVE INPUT IN BL

    MOV AH, 4CH
    INT 21H

END MAIN
```



Technical Example of

B) Process

Loads 05h and 03h into registers

Adds values and stores result in AL

Demonstrates basic arithmetic operation

```
.MODEL SMALL
.STACK 100H
.CODE

MAIN:
    MOV AL, 05H ;FIRST NUMBER
    MOV BL, 03H ;SECOND NUMBER
    ADD AL, BL  ;AL = AL + BL = 8

    MOV AH, 4CH
    INT 21H

END MAIN
```

original source co...

```
.MODEL SMALL
.STACK 100H
.CODE

MAIN:
MOV AL, 05H     ;FIRST NU
MOV BL, 03H     ;SECOND N
ADD AL, BL    ;AL = AL +

MOV AH, 4CH
INT 21H

END MAIN
```

emulator: noname.exe_

file debug view virtual devices virtual drive help

LOAD | reload | step back | single step | run | 0 step delay ms: 0

registers

	H	L
AX	4C	08
BX	00	03
CX	01	0A
DX	00	00
CS	0720	
IP	001E	
SS	0710	
SP	0100	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0720:001E

```
07216: 90 144 É
07217: 90 144 É
07218: 90 144 É
07219: 90 144 É
0721A: 90 144 É
0721B: 90 144 É
0721C: 90 144 É
0721D: 90 144 É
0721E: F4 244 ⌠
0721F: 00 000 NULL
07220: 00 000 NULL
07221: 00 000 NULL
07222: 00 000 NULL
07223: 00 000 NULL
07224: 00 000 NULL
07225: 00 000 NULL
07226: 00 000 NULL
07227: 00 000 NULL
07228: 00 000 NULL
07229: 00 000 NULL
0722A: 00 000 NULL
0722B: 00 000 NULL
```

0720:001E

```
NOP
NOP
NOP
NOP
HLT
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
...
```

screen | source | reset | aux | vars | debug | stack | flags

Technical Example of

C) Output

Displays "HELLO!" message on screen

Uses data segment for string storage

Implements string output function

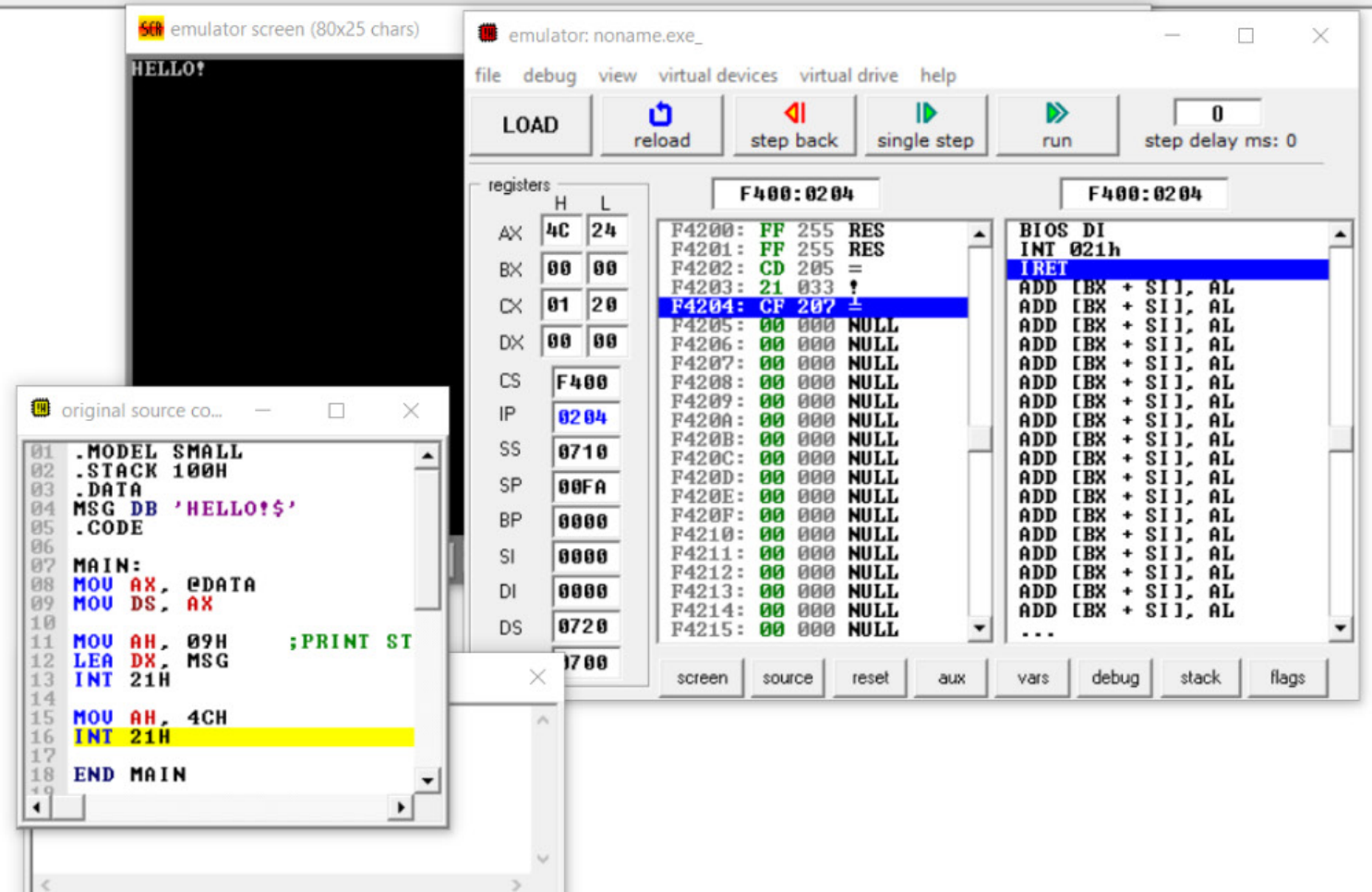
```
.MODEL SMALL
.STACK 100H
.DATA
    MSG DB 'HELLO!$'
.CODE

MAIN:
    MOV AX, @DATA
MOV DS, AX

MOV AH, 09H ;PRINT STRING
LEA DX, MSG
INT 21H

MOV AH, 4CH
INT 21H

END MAIN
```



Overview & Types of Register

Register Type	Registers	Bits	Code Example (EMU-8086)
General Purpose (Input Role)	AX, BX, CX, DX	16-bit	MOV AX, [var1] IN AL, 60h
Index — Source	SI	16-bit	MOV SI,OFFSET src LODSB ;AL<-[SI]
Segment — Data	DS	16-bit	MOV AX,@data MOV DS,AX
Accumulator (ALU Center)	AX / AL / AH	16/8-bit	ADD AX,BX MUL CX ;DX:AX=AX*CX
Count Register (Loop Control)	CX	16-bit	MOV CX,10 lbl: ADD AX,BX LOOP lbl
Instr. Pointer	IP	16-bit	JMP label CALL myProc RET
Segment — Extra	ES	16-bit	MOV AX,0B800h MOV ES,AX STOSW
Stack Pointer (Return Values)	SP	16-bit	PUSH param CALL func ADD SP,2 ;cleanup
DX — I/O Port & High Word	DX	16-bit	MUL BX ;DX:AX=res OUT DX,AL
SI + DI Together (Block Transfer)	SI, DI	16-bit	MOV CX,100 CLD REP MOVSB