



American International University- Bangladesh

Department of Computer Science

Lab Report Cover Sheet

Course Name	MICROPROCESSOR AND EMBEDDED SYSTEMS
Lab Report No.	05
Lecturer Name	MD. ALI NOOR
Semester	SUMMER 2021-22
Submission Date	23/06/2022
Section	0
Group No.	03

Student Name	Student ID	Contribution (out of 100%)
1. EMON, MEDEDI HASAN	19-41118-2	
2. ASIF AHAMMED	19-40020-1	
3. EASINUL ABEDIN SAYEM	19-40291-1	
4. MD REDOY SHEIKH	19-40101-1	
5. MD. NAZIM HASAN	19-40118-1	
6. MOHI UDDIN ANANDO	20-43585-1	

Title: Familiarization of assembly language programs

Objective:

1. To get familiarized with assembly language and work with advanced 8086 instructions.
2. Understand the process of branching and looping instructions in assembly language
3. Write, compile and execute flow control structures in assembly language programs using EMU 8086

Introduction:

The microprocessor 8086 serves as the foundation for the Intel X-86 family of processors. With this 16-bit processor under one's belt, one can investigate the 80386, 80406, and Pentium processors.

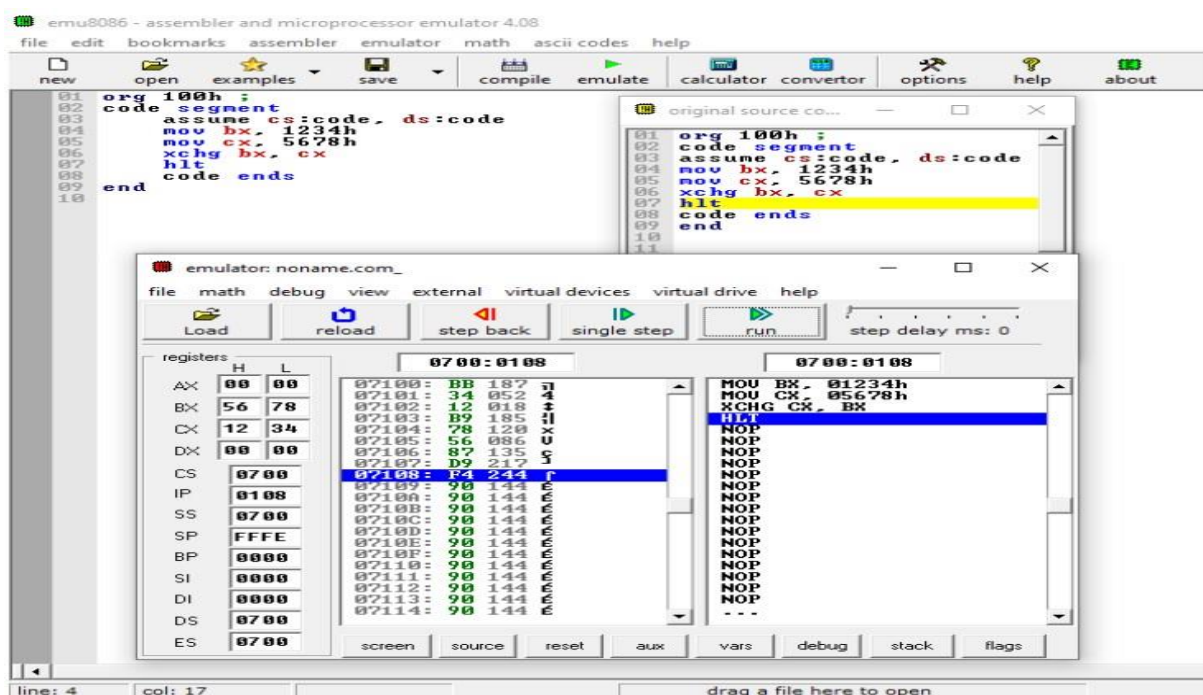
"MTS-86c" and "MDA 8086" are the micro-kits we're employing. There are numerous instructions in the 8086. In this experiment, the main objective is to learn how to write assembly programs using 8086 instructions and arrays.

Apparatus:

Serial No.	Components	Rating	Quantity
1	EMU8086	[ver.408 (32 bit WINOS compatible)]	1
2	PC having Intel Microprocessor		1

Lab Task(Code/Simulation):

Write a program to exchange the contents of two registers.



file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
01 code segment
02
03 assume cs:code, ds:code
04
05 mov ax, 1234h
06 mov bx, 5678h
07
08 mov cx, ax ; cx=1234
09 mov ax, bx ; ax=5678
10 mov bx, cx ; bx=1234
11
12 hlt
13 code ends
14
```

original source co...

```
01 code segment
02
03 assume cs:code,
04
05 mov ax, 1234h
06 mov bx, 5678h
07
08 mov cx, ax ; cx=
09 mov ax, bx ; ax=
10 mov bx, cx ; bx=
11
12 hlt
13 code ends
```

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		0100:000C		0100:000C	
	H	L			
AX	56	78	01000: B8	184	MOV AX, 01234h
BX	12	34	01001: 34	052	MOV BX, 05678h
CX	12	34	01002: 12	018	MOV CX, AX
DX	00	00	01003: BB	187	MOV AX, BX
CS	0100		01004: 78	120	MOV BX, CX
IP	000C		01005: 56	086	HLT
SS	0100		01006: 8B	139	NOP
SP	FFFE		01007: C8	200	NOP
BP	0000		01008: 8B	139	NOP
SI	0000		01009: C3	195	NOP
DI	0000		0100A: 8B	139	NOP
DS	0100		0100B: D9	217	NOP
ES	0100		0100C: F4	244	NOP
			0100D: 90	144	NOP
			0100E: 90	144	NOP
			0100F: 90	144	NOP
			01010: 90	144	NOP
			01011: 90	144	NOP
			01012: 90	144	...

screen source reset aux vars debug stack flags

Write a program for adding two numbers.

The screenshot displays the emu8086 software interface. The main window shows the assembly code for a program that adds two numbers. The code is as follows:

```

01 org 100h ;
02 code segment
03     assume cs:code, ds:code
04     mov al, 13h
05     mov dl, 01h
06     add al, dl
07     hlt
08 code ends
09 end
10

```

A secondary window titled 'original source co...' shows the same code, with the 'HLT' instruction highlighted in yellow. Below the code windows, the emulator window 'emulator: noname.com_' is open. It features a toolbar with buttons for Load, reload, step back, single step, run, and a step delay slider. The registers window on the left shows the current state of the 8086 registers:

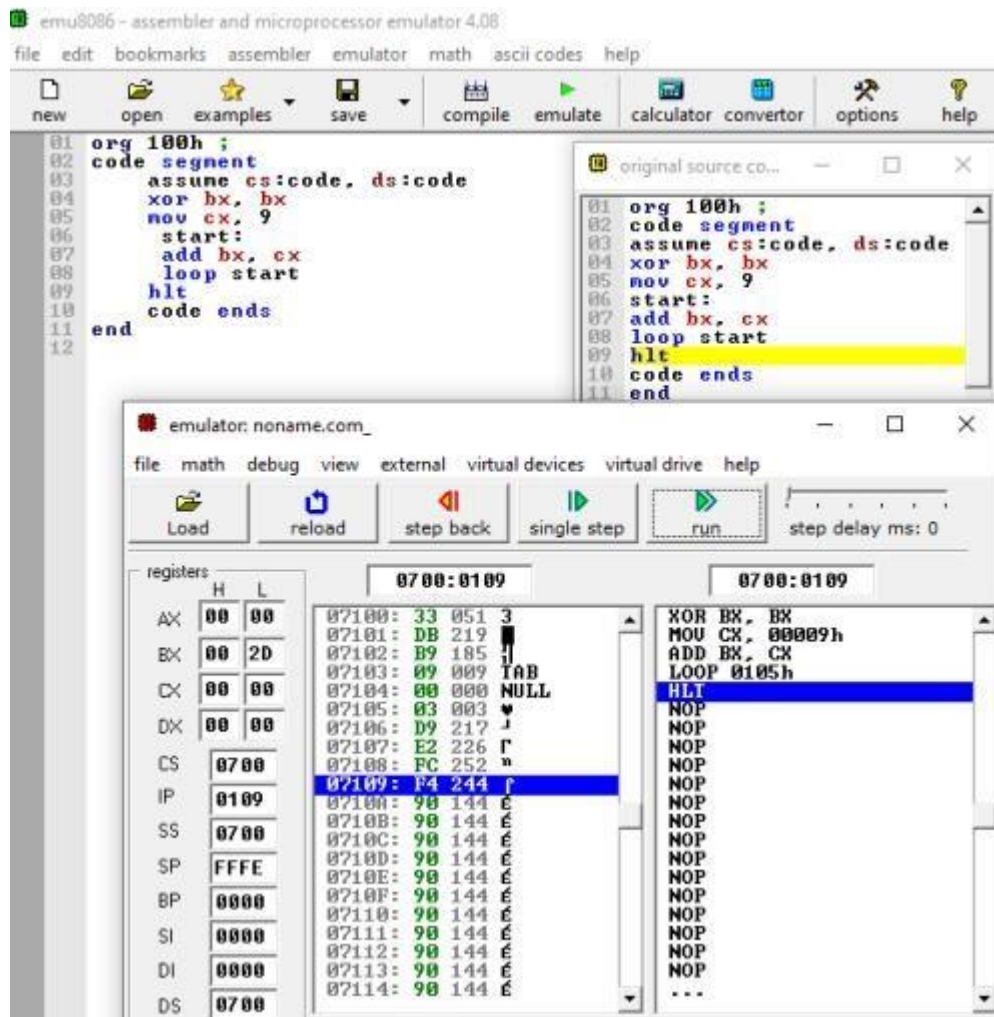
Register	H	L
AX	00	14
BX	00	00
CX	00	07
DX	00	01
CS	0700	
IP	0106	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The memory window shows the instruction stream starting at address 07100:

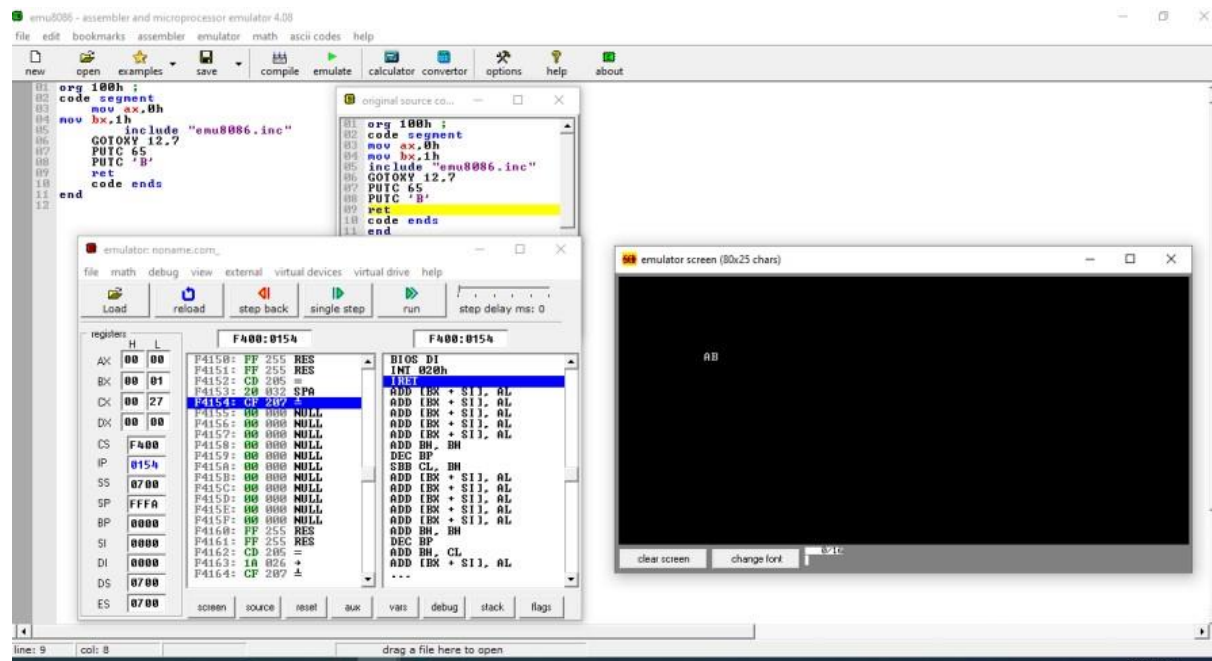
Address	Hex	Dec	Instruction
07100	B0	176	MOV AL, 013h
07101	13	019	MOV DL, 01h
07102	B2	178	ADD AL, DL
07103	01	001	HLT
07104	02	002	NOP
07105	C2	194	NOP
07106	F4	244	NOP
07107	90	144	NOP
07108	90	144	NOP
07109	90	144	NOP
0710A	90	144	NOP
0710B	90	144	NOP
0710C	90	144	NOP
0710D	90	144	NOP
0710E	90	144	NOP
0710F	90	144	NOP
07110	90	144	NOP
07111	90	144	NOP
07112	90	144	NOP
07113	90	144	NOP
07114	90	144	NOP

The bottom of the emulator window has tabs for screen, source, reset, aux, vars, debug, stack, and flags.

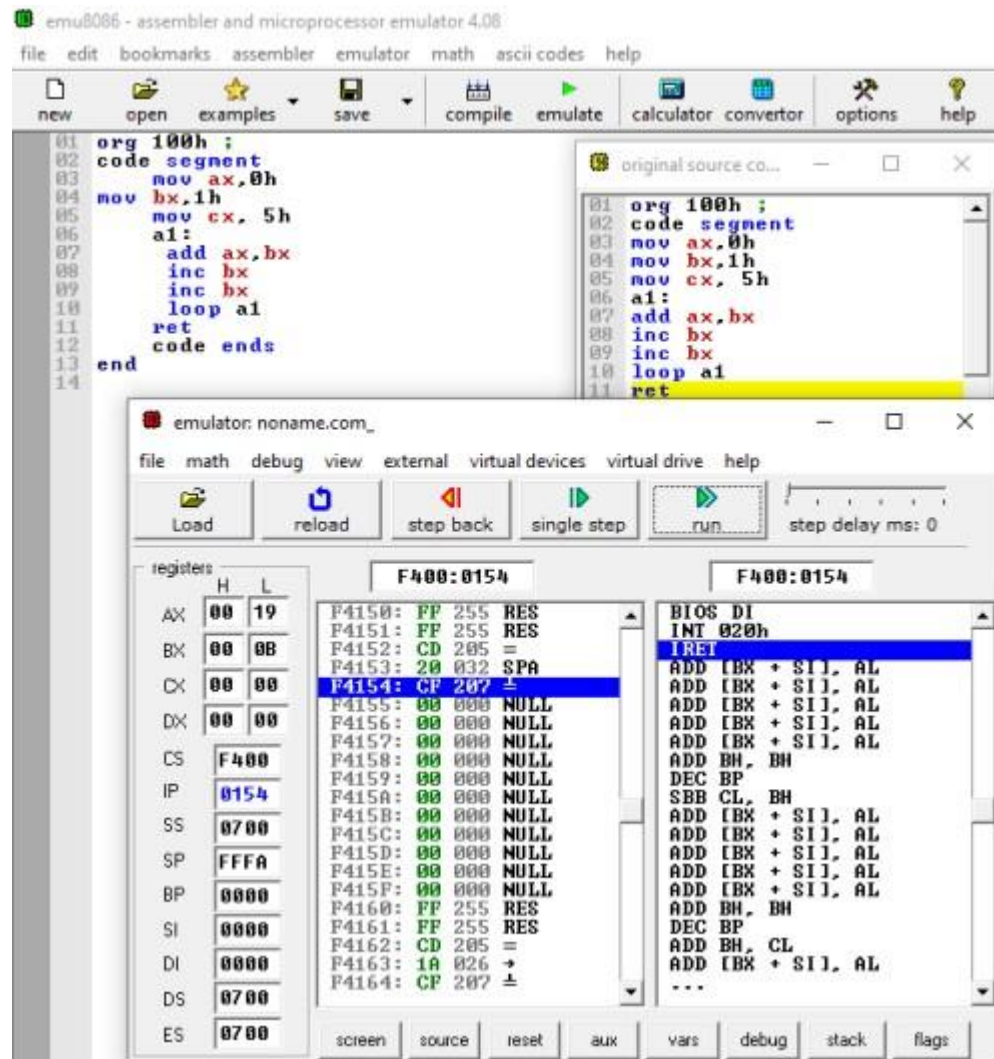
Write a program for subtraction between two numbers



Write a program which display two characters at column#12 and row#7 at emulator screen.



Write the assembly code for the following sequence $1+3+5+7+\dots+N$. Where $N = 5$ using a loop.



Write a code for finding the value of 6!

emu8086 - assembler and microprocessor emulator 4.08

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help

```
01 org 100h ;
02 code segment
03     mov ax, 1h
04     mov cx, 6h
05     li:
06     mul cx
07     loop li
08     ret
09 code ends
10 end
11
```

original source co...

```
01 org 100h ;
02 code segment
03     mov ax, 1h
04     mov cx, 6h
05     li:
06     mul cx
07     loop li
08     ret
09 code ends
10 end
11
```

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

	H	L
AX	02	D0
BX	00	00
CX	00	00
DX	00	00
CS	F400	
IP	0154	
SS	0700	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

F400:0154

F4150:	FF	255	RES
F4151:	FF	255	RES
F4152:	CD	205	=
F4153:	20	032	SPA
F4154:	CF	207	±
F4155:	00	000	NULL
F4156:	00	000	NULL
F4157:	00	000	NULL
F4158:	00	000	NULL
F4159:	00	000	NULL
F415A:	00	000	NULL
F415B:	00	000	NULL
F415C:	00	000	NULL
F415D:	00	000	NULL
F415E:	00	000	NULL
F415F:	00	000	NULL
F4160:	FF	255	RES
F4161:	FF	255	RES
F4162:	CD	205	=
F4163:	1A	026	+
F4164:	CF	207	±

BIOS DI INT 020h

```
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
DEC BP
SBB CL, BH
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

Homework:

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
01 org 100h
02 code segment
03 ;Random values in registers
04 mov ax, 0303h
05 mov bx, 2020h
06 mov cx, 0007h
07
08 add ax, bx ; AX= AX+BX
09
10 sub ax, cx ; AX= AX-CX = AX+BX-CX
11
12 mov dx, ax ; DX= AX+BX-CX
13
14 hlt
15 code ends
16
17 end
18
```

original source co...

```
01 org 100h
02 code segment
03 ;Random values in registers
04 mov ax, 0303h
05 mov bx, 2020h
06 mov cx, 0007h
07
08 add ax, bx ; A>
09
10 sub ax, cx ; A>
11
12 mov dx, ax ; D>
13
14 hlt
15 code ends
16
17 end
18
```

emulator: Lab 6 equation.com_

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	23	1C
BX	20	20
CX	00	07
DX	23	1C
CS	0700	
IP	010F	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:010F

07100:	B8	184	MOV AX, 00303h
07101:	03	003	MOV BX, 02020h
07102:	03	003	MOV CX, 00007h
07103:	B8	187	ADD AX, BX
07104:	20	032	SUB AX, CX
07105:	20	032	MOV DX, AX
07106:	B9	185	MOV AX, CX
07107:	07	007	NOI
07108:	00	000	NOI
07109:	03	003	NOI
0710A:	C3	195	NOI
0710B:	2B	043	NOI
0710C:	C1	193	NOI
0710D:	05	193	NOI
0710E:	D0	208	NOI
0710F:	F4	244	NOI
07110:	90	144	NOI
07111:	90	144	NOI
07112:	90	144	NOI
...			...

screen source reset aux vars debug stack flags

Discussion:

While writing each program, we learnt the reason for using each function and the associated values. We were able to understand each line of code, which allowed us to solve a specific problem or task using the knowledge we've gained from this experiment.