Microcontroller Lab Report

8086 programming Part 2c

Aaditya Prakash Kattekola

Roll: 194201

ECE Section B

August 28, 2021

1 Question 1

1.1 Aim

Write an assembly language program for 8086 processor find the largest number in a array. Length of the array is N. Where N represent the last 2 digits of your roll number. (Since array of length 01 is not ideal, I used array of length 5)

1.2 Program

1.2.1 Code

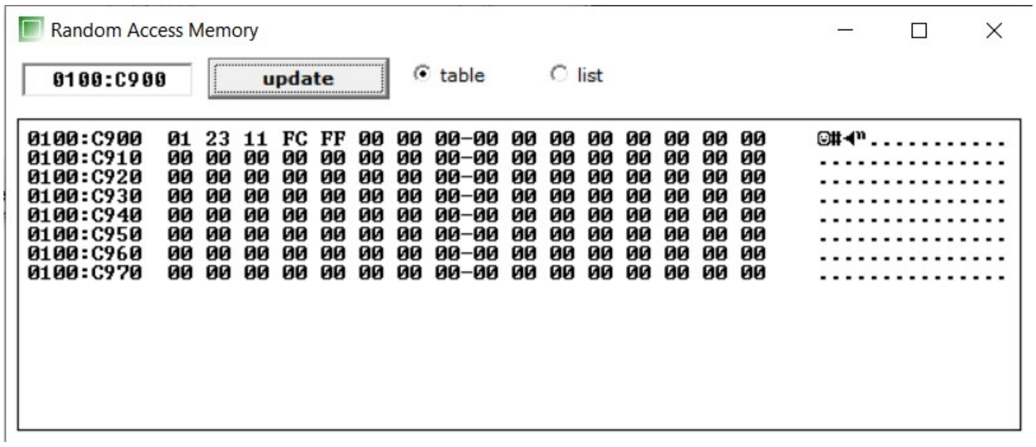
```
;ROLL => 194201
;LARGEST ARRAY ELEMENT
MOV CL, 05H ;ARRAY SIZE, LAST 2 DIGITS ARE 01
MOV SI, 0C900H
MOV DX, 00000H

ITER: MOV AL, [SI]
CMP DL, AL
JNC UPDATE
MOV DL, AL
UPDATE: INC SI
CLC
DEC CL
JNZ ITER
HLT ;HALT
```

1.2.2 Emulator

Address (CS:0100, IP:0000)	Machine code	Instruction
01000	B1, 05	MOV CL, 05H
01002	BE, 00, C9	MOV 0C900H
01005	BA, 00, 00	MOV DX, 00000H
01008	8A, 04	MOV AL, [SI]
0100A	3A, D0	CMP DL, AL
0100C	73, 02	JNB 010H
0100E	8A, D0	MOV DL, AL
01010	46	INC SI
01011	F8	CLC
01012	FE, C9	DEC CL
01014	75, F2	JNE 0108H
01016	F4	HLT

1.3 Result

1.3.1 Input



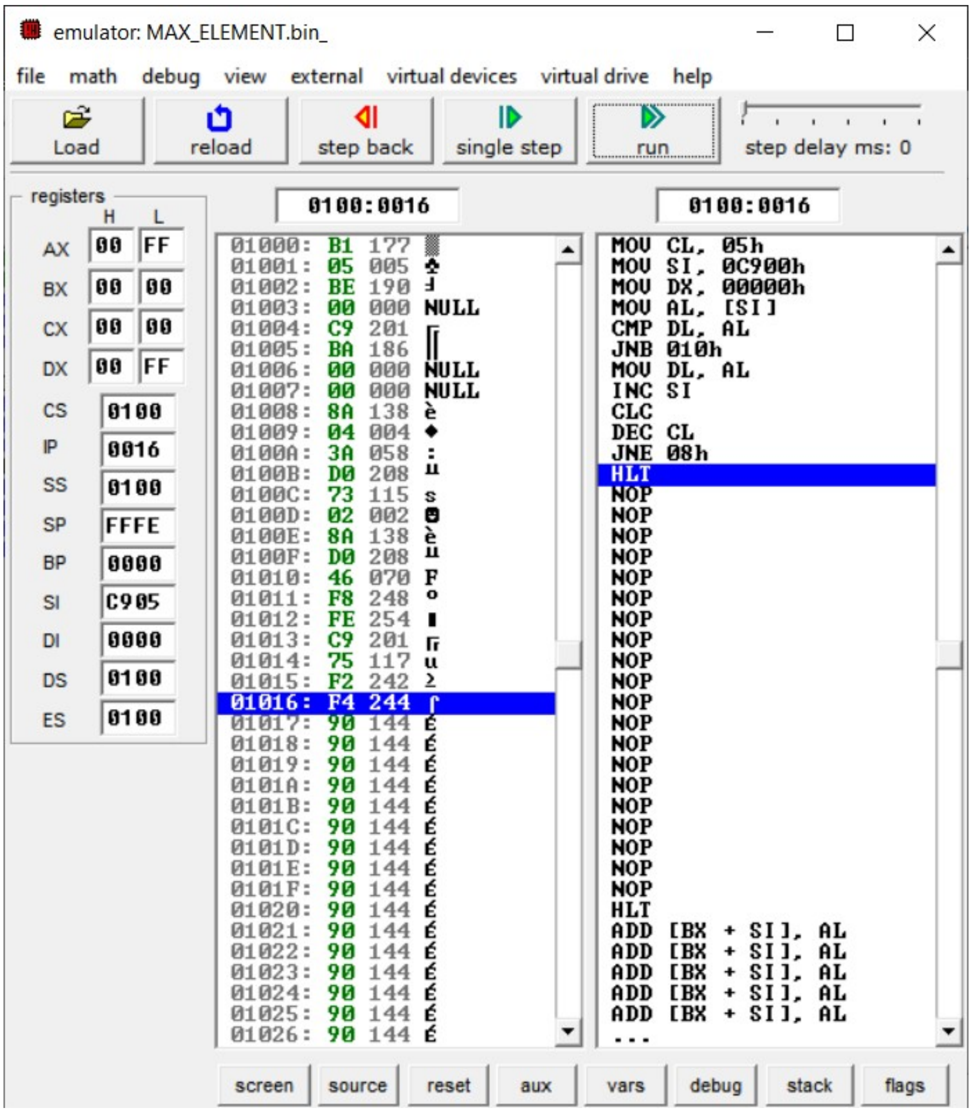
```
0100:C900  01 23 11 FC FF 00 00 00-00 00 00 00 00 00 00 00    ☺#◄ⁿ...........
0100:C910  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ...............
0100:C920  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ...............
0100:C930  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ...............
0100:C940  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ...............
0100:C950  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ...............
0100:C960  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ...............
0100:C970  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ...............
```

Figure 1: RAM Input for Q1

1.3.2 Expectation

$Final\ Ans:\ FF\ in\ DX$

1.3.3 Emulator


emulator: MAX_ELEMENT.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   FF
BX    00   00
CX    00   00
DX    00   FF
CS    0100
IP    0016
SS    0100
SP    FFFE
BP    0000
SI    C905
DI    0000
DS    0100
ES    0100
```

```
0100:0016

01000: B1 177 ▒
01001: 05 005 ♣
01002: BE 190 ╛
01003: 00 000 NULL
01004: C9 201 ╔
01005: BA 186 ║
01006: 00 000 NULL
01007: 00 000 NULL
01008: 8A 138 è
01009: 04 004 ♦
0100A: 3A 058 :
0100B: D0 208 ╨
0100C: 73 115 s
0100D: 02 002 ☻
0100E: 8A 138 è
0100F: D0 208 ╨
01010: 46 070 F
01011: F8 248 °
01012: FE 254 ■
01013: C9 201 ╔
01014: 75 117 u
01015: F2 242 ≥
01016: F4 244 ⌠
01017: 90 144 É
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
```

```
0100:0016

MOV CL, 05h
MOV SI, 0C900h
MOV DX, 00000h
MOV AL, [SI]
CMP DL, AL
JNB 010h
MOV DL, AL
INC SI
CLC
DEC CL
JNE 08h
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
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
...
```

Figure 2: STACK Output for Q1

2 Question 2

2.1 Aim

Write an assembly language program for 8086 processor to calculate LCM of two 8 bit numbers.

2.2 Program

2.2.1 Code

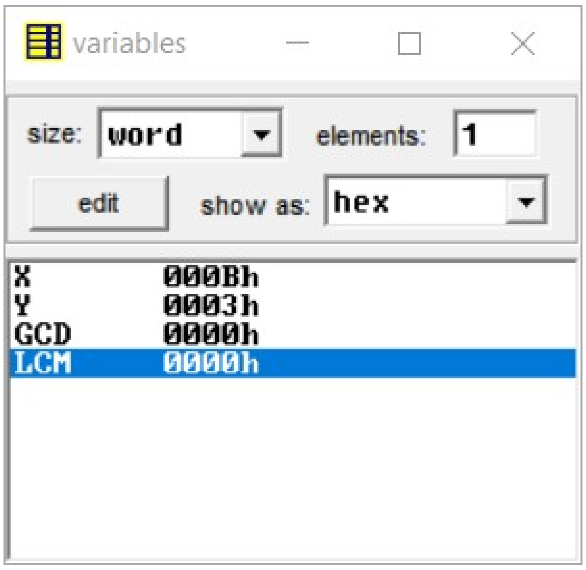
```
;ROLL => 194201
;LCM
MOV SI, 0C900H
X DW 000BH ;11
Y DW 0003H ;3
GCD DW ?
LCM DW ?

MOV AX, X
MOV BX, Y

COMP: CMP AX, BX
JE FINAL
JB SWAP

DIVIDE: MOV DX, 00000H
DIV BX
CMP DX, 00000H
JE FINAL
MOV AX, DX
JMP COMP

SWAP: XCHG AX, BX
JMP DIVIDE


FINAL: MOV GCD, BX
MOV AX, X
MOV BX, Y
MUL BX
DIV GCD
MOV LCM, AX
MOV DX, LCM

HLT ;HALT
```

2.2.2 Emulator

Address (CS:0100, IP:0000)	Machine code	Instruction
01000	BE, 00, C9	MOV SI, 0C900H
01003	0B, 00	OR AX, [BX + SI]
01005	03, 00	ADD AX, [BX + SI]
01007	01, 00	ADD [BX + SI], AX
01009	21, 00	AND [BX + SI], AX
0100B	A1, 03, 00	MOV AX, [00003H]
0100E	8B, 1E, 05, 00	MOV BX, [00005H]
01012	3B, C3	CMP AX, BX
01014	74, 13	JZ 029H
01016	72, 0E	JB 026H
01018	BA, 00, 00	MOV DX, 00000H
0101B	F7, F3	DIV BX
0101D	83, FA, 00	CMP DX, 00H
01020	74, 07	JZ 029H
01022	8B, C2	MOV AX, DX
01024	EB, EC	JMP 012H
01026	96	XCHG AX, BX
01027	EB, EF	JMP 018H
01029	80, 1E, 07, 00	MOV [00007H], BX
0102D	A1, 03, 00	MOV AX, [00003H]
01030	8B, 1E, 05, 00	MOV BX, [00005H]
01034	F7, E3	MUL BX
01036	F7, 36, 07, 00	DIV W.[00007H]
0103A	A3, 09, 00	MOV [00009], AX
0103D	8B, 16, 09, 00	MOV DX, [00009H]
01041	F4	HLT

2.3 Result

2.3.1 Input



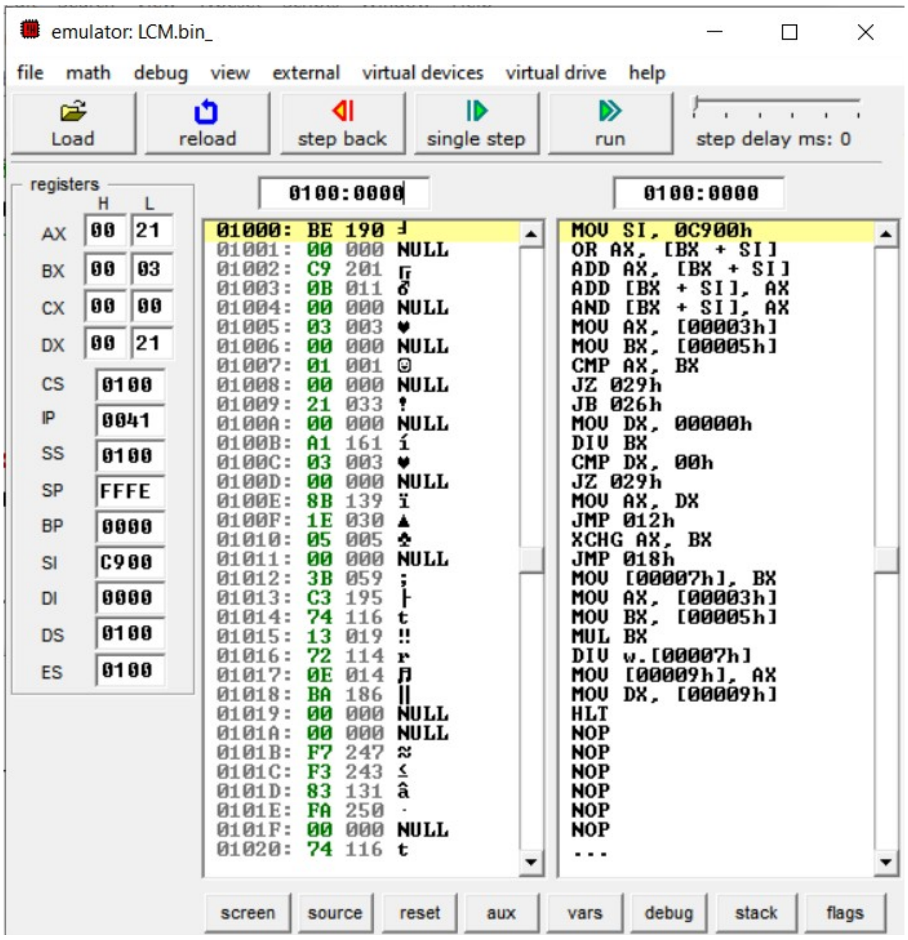
```
X       000Bh
Y       0003h
GCD     0000h
LCM     0000h
```

Figure 3: VARIABLE Input for Q2

2.3.2 Expectation

$0021H$ (33 *in DEC*) *in DX*

2.3.3 Emulator

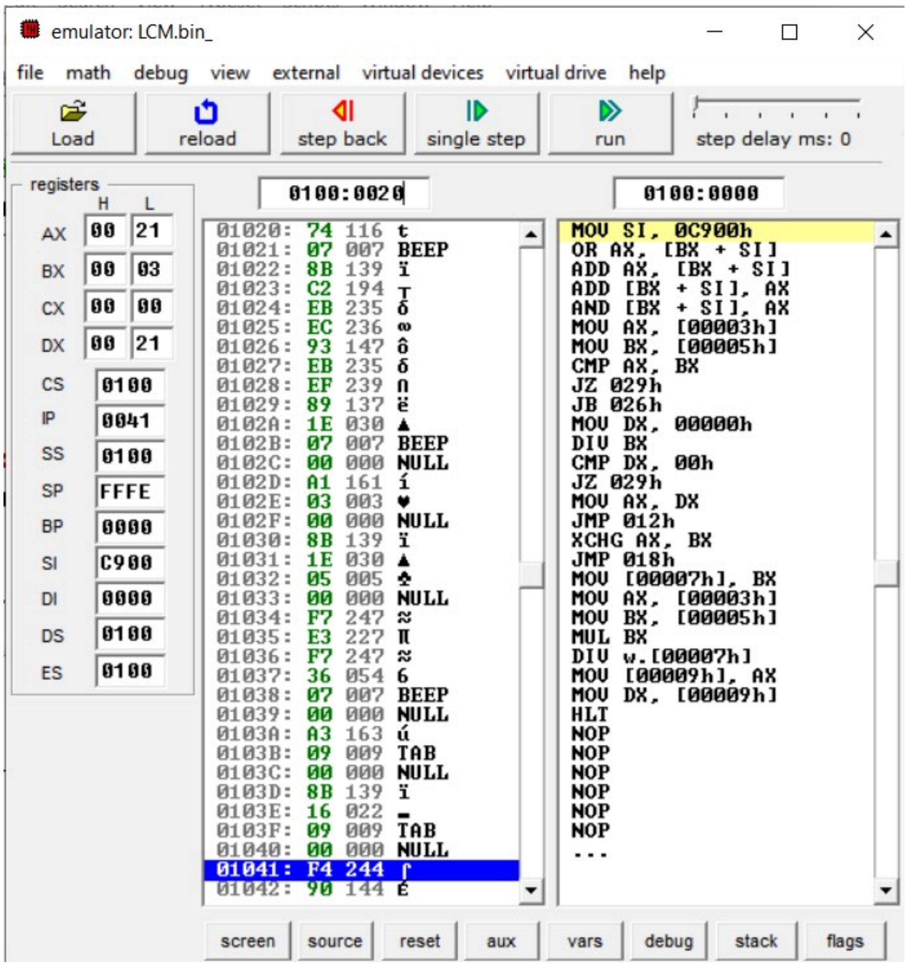


```
registers
      H    L
AX   00   21
BX   00   03
CX   00   00
DX   00   21
CS   0100
IP   0041
SS   0100
SP   FFFE
BP   0000
SI   C900
DI   0000
DS   0100
ES   0100
```

```
0100:0000

01000: BE 190 ╛
01001: 00 000 NULL
01002: C9 201 ╔
01003: 0B 011 ♂
01004: 00 000 NULL
01005: 03 003 ♥
01006: 00 000 NULL
01007: 01 001 ☺
01008: 00 000 NULL
01009: 21 033 !
0100A: 00 000 NULL
0100B: A1 161 í
0100C: 03 003 ♥
0100D: 00 000 NULL
0100E: 8B 139 ï
0100F: 1E 030 ▲
01010: 05 005 ♣
01011: 00 000 NULL
01012: 3B 059 ;
01013: C3 195 ├
01014: 74 116 t
01015: 13 019 ‼
01016: 72 114 r
01017: 0E 014 ♫
01018: BA 186 ║
01019: 00 000 NULL
0101A: 00 000 NULL
0101B: F7 247 ≈
0101C: F3 243 ≤
0101D: 83 131 â
0101E: FA 250 ·
0101F: 00 000 NULL
01020: 74 116 t
```

```
0100:0000

MOV SI, 0C900h
OR AX, [BX + SI]
ADD AX, [BX + SI]
ADD [BX + SI], AX
AND [BX + SI], AX
MOV AX, [00003h]
MOV BX, [00005h]
CMP AX, BX
JZ 029h
JB 026h
MOV DX, 00000h
DIV BX
CMP DX, 00h
JZ 029h
MOV AX, DX
JMP 012h
XCHG AX, BX
JMP 018h
MOV [00007h], BX
MOV AX, [00003h]
MOV BX, [00005h]
MUL BX
DIV w.[00007h]
MOV [00009h], AX
MOV DX, [00009h]
HLT
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Figure 4: STACK Output for Q2.1

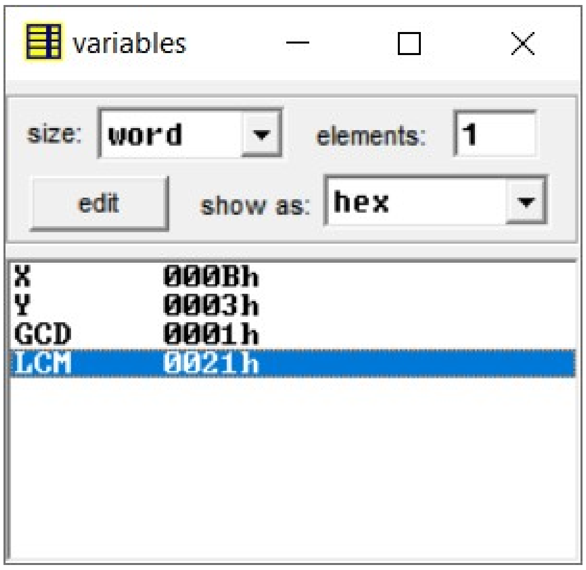


```
registers
      H    L
AX   00   21
BX   00   03
CX   00   00
DX   00   21
CS   0100
IP   0041
SS   0100
SP   FFFE
BP   0000
SI   C900
DI   0000
DS   0100
ES   0100
```

```
0100:0020

01020: 74 116 t
01021: 07 007 BEEP
01022: 8B 139 ï
01023: C2 194 ┬
01024: EB 235 δ
01025: EC 236 ∞
01026: 93 147 ô
01027: EB 235 δ
01028: EF 239 ∩
01029: 89 137 ë
0102A: 1E 030 ▲
0102B: 07 007 BEEP
0102C: 00 000 NULL
0102D: A1 161 í
0102E: 03 003 ♥
0102F: 00 000 NULL
01030: 8B 139 ï
01031: 1E 030 ▲
01032: 05 005 ♣
01033: 00 000 NULL
01034: F7 247 ≈
01035: E3 227 π
01036: F7 247 ≈
01037: 36 054 6
01038: 07 007 BEEP
01039: 00 000 NULL
0103A: A3 163 ú
0103B: 09 009 TAB
0103C: 00 000 NULL
0103D: 8B 139 ï
0103E: 16 022 ▬
0103F: 09 009 TAB
01040: 00 000 NULL
01041: F4 244 ⌠
01042: 90 144 É
```

```
0100:0000

MOV SI, 0C900h
OR AX, [BX + SI]
ADD AX, [BX + SI]
ADD [BX + SI], AX
AND [BX + SI], AX
MOV AX, [00003h]
MOV BX, [00005h]
CMP AX, BX
JZ 029h
JB 026h
MOV DX, 00000h
DIV BX
CMP DX, 00h
JZ 029h
MOV AX, DX
JMP 012h
XCHG AX, BX
JMP 018h
MOV [00007h], BX
MOV AX, [00003h]
MOV BX, [00005h]
MUL BX
DIV w.[00007h]
MOV [00009h], AX
MOV DX, [00009h]
HLT
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Figure 5: STACK Output for Q2.2



```
size: word        elements: 1
show as: hex

X      000Bh
Y      0003h
GCD    0001h
LCM    0021h
```

Figure 6: VARIABLE Output for Q2