Microcontroller Lab Report

8086 programming Part 2a

Aaditya Prakash Kattekola

Roll: 194201

ECE Section B

August 28, 2021

1 Question 1

1.1 Aim

Write an efficient assembly language program (minimum code length) for 8086 MP for a system has four inputs and four outputs. The four output bits represents the gray code equivalent of input binary number.

1.2 Program

1.2.1 Code

```
;ROLL => 194201
;GRAY CODE CONVERTER
MOV AX, 1011B ;NUMBER TO BE CONVERTED IS MOVED TO AX
MOV BX, AX

SHR AX, 01 ;SHIFT AX 1 BIT RIGHT
XOR AX, BX ; AX <- AX ^ BX

HLT
```

1.2.2 Emulator

Address (CS:0100, IP:0000)	Machine code	Instruction
01000	B8,0B,00	MOV AX, 0000BH
01003	8B, D8	MOV BX, AX
01005	D1, E8	SHR AX, 1
01007	33, C3	XOR AX, BX
01009	F4	HLT

1.3 Result

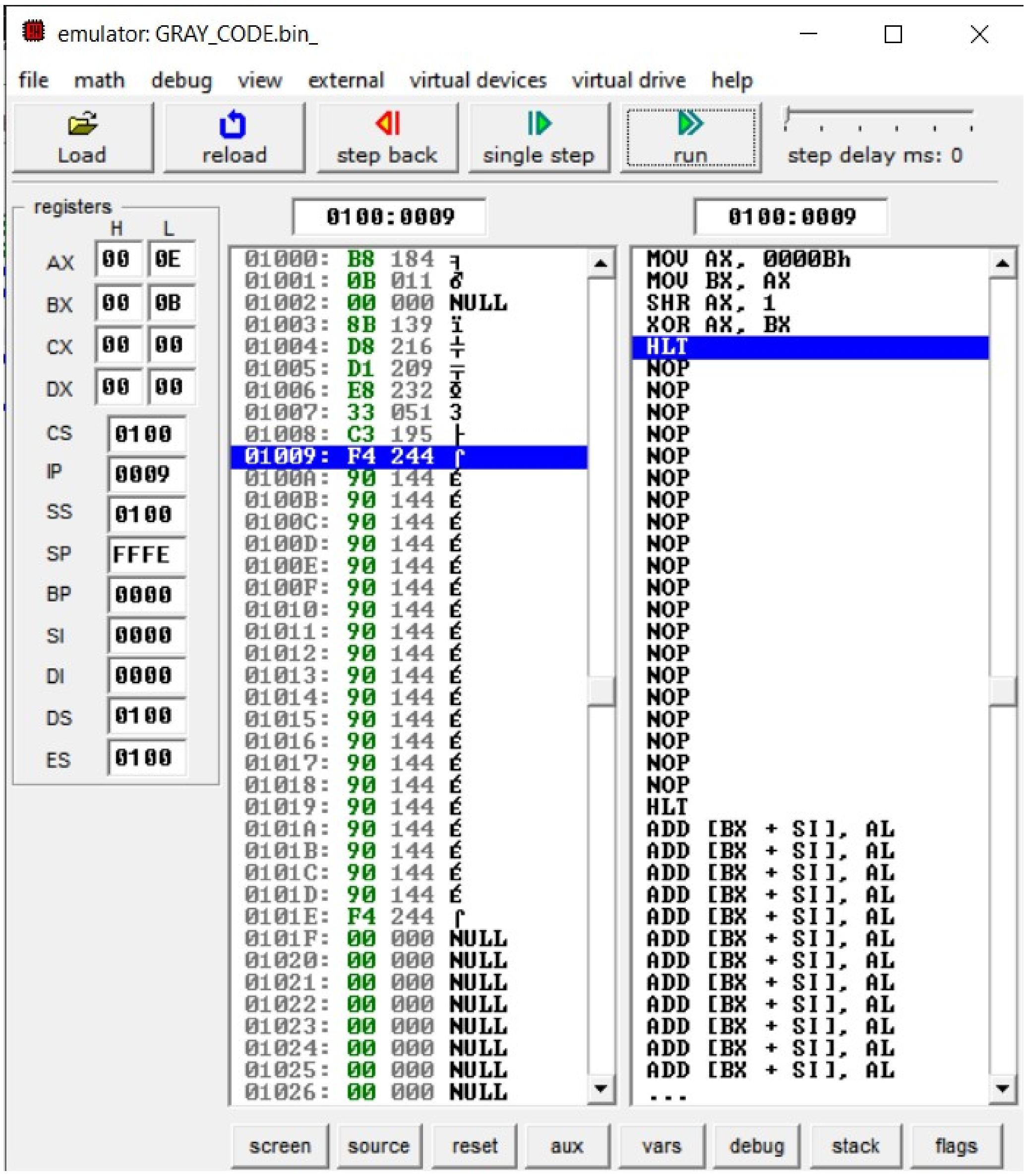
1.3.1 Input

AX: 1011B (B in HEX)

1.3.2 Expectation

$AX \leftarrow AX \oplus BX;$
$1011B + 0101B \rightarrow 1110B\ or\ 000EH\ in\ AX$

1.3.3 Emulator



emulator: GRAY_CODE.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  0E
BX   00  0B
CX   00  00
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: 0B 011 ♂
01002: 00 000 NULL
01003: 8B 139 ï
01004: D8 216 ╪
01005: D1 209 ╤
01006: E8 232 Φ
01007: 33 051 3
01008: C3 195 ├
01009: F4 244 ⌠
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
01016: 90 144 É
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: F4 244 ⌠
0101F: 00 000 NULL
01020: 00 000 NULL
01021: 00 000 NULL
01022: 00 000 NULL
01023: 00 000 NULL
01024: 00 000 NULL
01025: 00 000 NULL
01026: 00 000 NULL
```

```
0100:0009

MOV AX, 0000Bh
MOV BX, AX
SHR AX, 1
XOR AX, BX
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
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
...
```

Figure 1: STACK Output for Q1

2 Question 2

2.1 Aim

Write a program for 8086 processor to generate the Fibonacci series (Each number in the Fibonacci series is the sum of the previous two numbers.)

2.2 Program

2.2.1 Code

```
;ROLL => 194201
;FINONACCI SEQUENCE
MOV AL, 00H ;FIRST TERM
MOV SI, 0C900H ;ADDRESS WHERE TERMS WILL BE STORED

MOV [SI], AL ;STORE AT POINTER LOCATION
INC SI
INC AL
MOV [SI], AL;

MOV CL, 05H ;NUMBER OF STEPS
SUB CL, 02H ;2 TERMS ARE ALREADY PRESENT

FIB: MOV AL, [SI-1] ;MAKE AX THE PREV VALUE
ADD AL, [SI] ;ADD CURRENT VALUE TO PREV VALUE
INC SI
MOV [SI], AL
DEC CL
JNZ FIB
HLT
```

2.2.2 Emulator

Address (CS:0100, IP:0000)	Machine code	Instruction
01000	B0, 00	MOV AL, 00H
01002	BE, 00, C9	MOV SI, 0C900H
01005	88, 04	MOV [SI], AL
01007	46	INC SI
01008	FE, C0	INC AL
0100A	88, 04	MOV [SI], AL
0100C	B1, 05	MOV CL, 05H
0100E	80, E9, 02	SUB CL, 02H
01011	8A, 44, FF	MOV AL, [SI] - 01H
01014	02, 04	ADD AL, [SI]
01016	46	INC SI
01017	88, 04	MOV [SI], AL
01019	FE, C9	DEC CL
0101B	FE, C9	JNE 011H
0101D	F4	HLT

2.3 Result

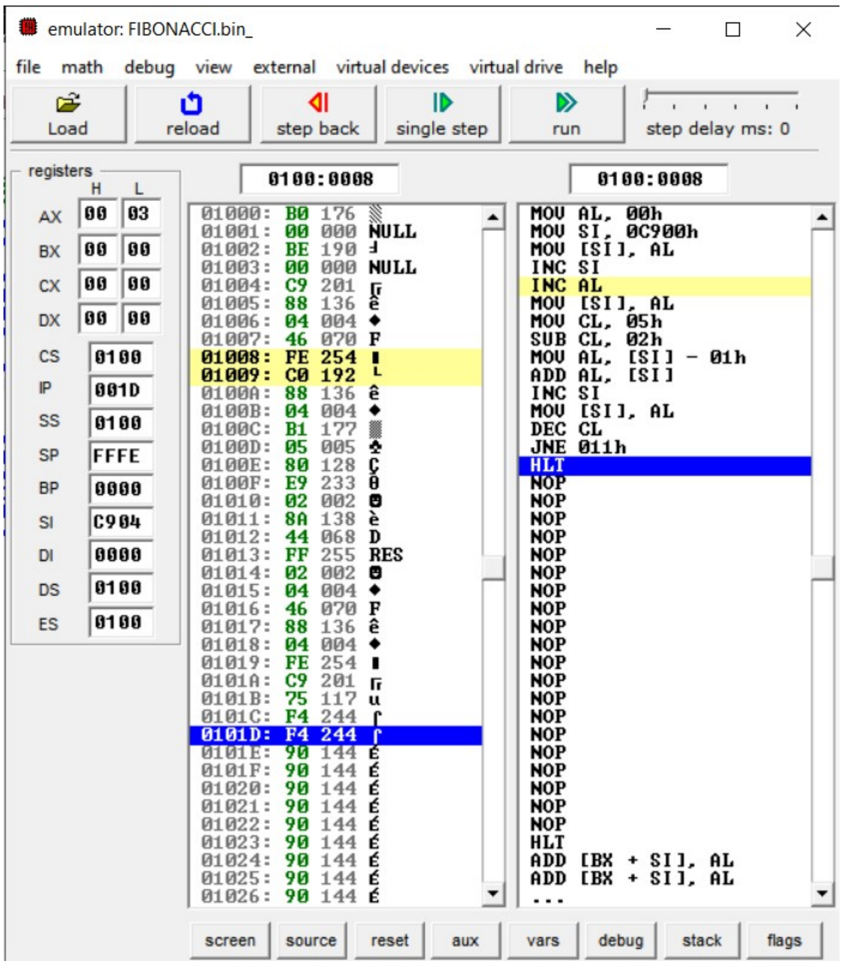
2.3.1 Input

CL: 05H ;LENGTH OF FIBONACCI SERIES

2.3.2 Expectation

$SI \leftarrow C900H$
$[C900]$: 00 $[C901]$: 01
$[SI+1] \leftarrow [SI] + [SI-1]$
0100 : $C900$ 00 01 01 02 03

2.3.3 Emulator



registers

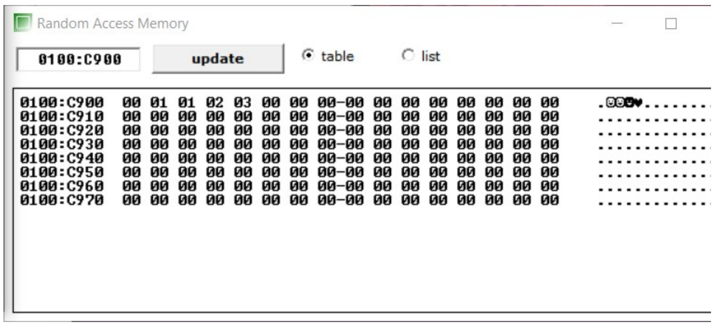
	H	L
AX	00	03
BX	00	00
CX	00	00
DX	00	00

CS	0100
IP	001D
SS	0100
SP	FFFE
BP	0000
SI	C904
DI	0000
DS	0100
ES	0100

```
0100:0008
01000: B0 176 ▒
01001: 00 000 NULL
01002: BE 190 ╛
01003: 00 000 NULL
01004: C9 201 ╔
01005: 88 136 ê
01006: 04 004 ♦
01007: 46 070 F
01008: FE 254 ■
01009: C0 192 └
0100A: 88 136 ê
0100B: 04 004 ♦
0100C: B1 177 ▒
0100D: 05 005 ♣
0100E: 80 128 Ç
0100F: E9 233 Θ
01010: 02 002 ☻
01011: 8A 138 è
01012: 44 068 D
01013: FF 255 RES
01014: 02 002 ☻
01015: 04 004 ♦
01016: 46 070 F
01017: 88 136 ê
01018: 04 004 ♦
01019: FE 254 ■
0101A: C9 201 ╔
0101B: 75 117 u
0101C: F4 244 ⌠
0101D: F4 244 ⌠
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
```

```
0100:0008
MOV AL, 00h
MOV SI, 0C900h
MOV [SI], AL
INC SI
INC AL
MOV [SI], AL
MOV CL, 05h
SUB CL, 02h
MOV AL, [SI] - 01h
ADD AL, [SI]
INC SI
MOV [SI], AL
DEC CL
JNE 011h
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
...
```

Figure 2: STACK Output for Q2



0100:C900   update   (● table ○ list)

```
0100:C900  00 01 01 02 03 00 00 00-00 00 00 00 00 00 00 00   .☺☺☻♥.........
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 3: RAM Output for Q2