ELE 338 – Preliminary Work 1

Anıl Karaca – 21728405

Q1)

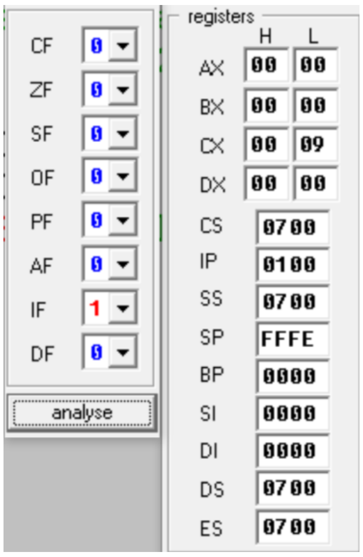
<u>Answer:</u> The 2 numbers I've chosen are *(FFFF)h and (0001)h*. It could be done using any 2 numbers that set both CF and ZF to 1.

<u>Source Code:</u>

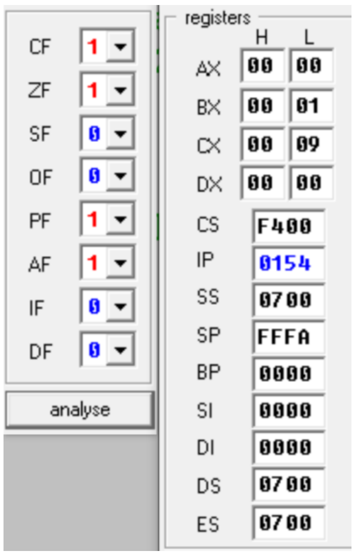
```
org 100h


;ELE338 - Preliminary Work 1 - Question 1

;Anil Karaca - 21728405


    MOV AX, 0FFFFh ;Assigning (FFFF)h to AX register

    MOV BX, 00001h ;Assigning (0001)h to BX register

    ADD AX, BX ;Making addition AX=AX+BX


ret
```

<u>Screenshots:</u>

Before and after the operation:



```
CF 0    registers
ZF 0         H  L
SF 0    AX  00 00
OF 0    BX  00 00
PF 0    CX  00 09
AF 0    DX  00 00
IF 1    CS  0700
DF 0    IP  0100
        SS  0700
        SP  FFFE
        BP  0000
        SI  0000
        DI  0000
        DS  0700
        ES  0700
```



```
CF 1    registers
ZF 1         H  L
SF 0    AX  00 00
OF 0    BX  00 01
PF 1    CX  00 09
AF 1    DX  00 00
IF 0    CS  F400
DF 0    IP  0154
        SS  0700
        SP  FFFA
        BP  0000
        SI  0000
        DI  0000
        DS  0700
        ES  0700
```

<u>Comments:</u> Here we basically just add (FFFF)h and (0001)h and store the result in AX.

This addition makes both CF and ZF equal to 1.

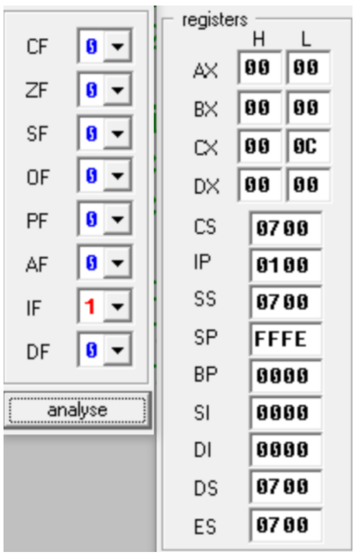
Q2)

Source Code:

```
org 100h


;ELE338 - Preliminary Work 1 - Question 2

;Anil Karaca - 21728405


MOV AX, 0d ;Assigning 1 to AX register which will indicate the
position

;MOV BX 10111111b ;Assigning the first example number to BX
register

MOV BX, 11111011b ;Assigning the second example number to BX
register


Back: ROR BX, 1 ;Rotating BX to the right by one to check the LSB

INC AX ;Increase the position AX

JC Back ;Jump if the bit(CF) is 1


ret
```

Screenshots:

Before and after the operation:



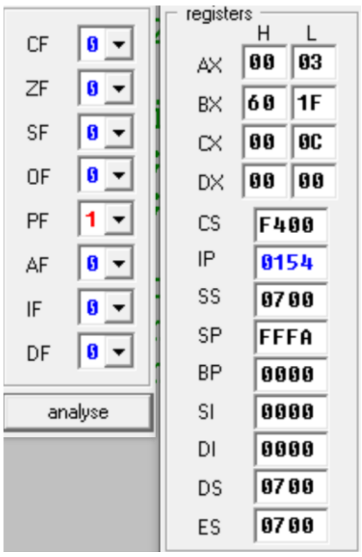
CF	0
ZF	0
SF	0
OF	0
PF	0
AF	0
IF	1
DF	0

analyse

registers

	H	L
AX	00	00
BX	00	00
CX	00	0C
DX	00	00

CS	0700
IP	0100
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700



CF	0
ZF	0
SF	0
OF	0
PF	1
AF	0
IF	0
DF	0

analyse

registers

	H	L
AX	00	03
BX	60	1F
CX	00	0C
DX	00	00

CS	F400
IP	0154
SS	0700
SP	FFFA
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

Comments: Here we basically rotate the binary number we store in BX to the right until we get a "0" in CF, while increasing the AX value which indicates the position.

At the end of the execution the value we got in AX is the result.

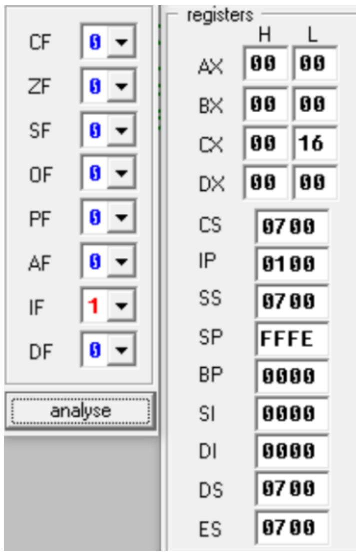
Q3)

Source Code:

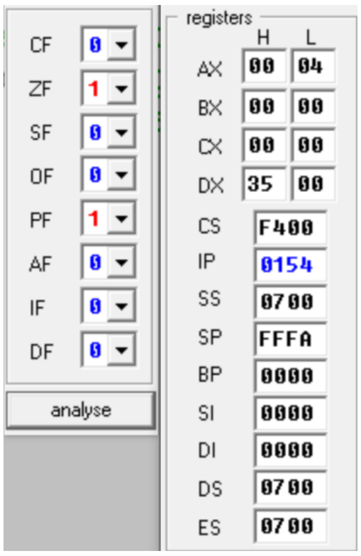
```
org 100h
;ELE338 - Preliminary Work 1 - Question 3
;Anil Karaca - 21728405

MOV CX, 8d ;Assigning 8 to CX, since we are dealing with 8-bit binary
numbers
;MOV DX, 00101001b; Assigning the first example to DX register
MOV DX, 01101010b; Assigning the second example to DX register
MOV AX, 0d ;Initializing a counter to AX register

JMP Rotate ;Jump to "Rotate" to initialize the process
FoundZero: INC AX ;Increase AX by 1
DEC CX ;Decrease the counter

Rotate: ROR DX, 1 ;Rotate DX to the right
JCXZ Done; Terminate the program if CX is 0
JNC FoundZero; Jump to "FoundZero" if CF is 0
LOOP Rotate; Decrease CX and jump to "Rotate" if CF 1

Done: ret
```

Screenshots:

Before and after the operation:



```
CF  0        registers
ZF  0             H   L
SF  0        AX   00  00
OF  0        BX   00  00
PF  0        CX   00  16
AF  0        DX   00  00
IF  1        CS   0700
DF  0        IP   0100
             SS   0700
analyse      SP   FFFE
             BP   0000
             SI   0000
             DI   0000
             DS   0700
             ES   0700
```



```
CF  0        registers
ZF  1             H   L
SF  0        AX   00  04
OF  0        BX   00  00
PF  1        CX   00  00
AF  0        DX   35  00
IF  0        CS   F400
DF  0        IP   0154
             SS   0700
analyse      SP   FFFA
             BP   0000
             SI   0000
             DI   0000
             DS   0700
             ES   0700
```

Comments: Here we basically rotate the binary number we store in DX to the right until the loop counter CX is equal to 0, while checking whether the bit we got in CF is 0 or not. If it is equal to 0, we increase the "0" bit counter AX by 1.

At the end of the execution the value we got in AX is the result.