BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electrical and Electronic Engineering

Course No. : EEE 416

Course Title: Microprocessor and Interfacing Laboratory

Logical Instructions and Jump Commands in Assembly Language

Name: Mir Sayeed Mohammad

ID: 1606003

Level: 4

Term: 1

Section: A

Submission Deadline: 14 - 3 -2021

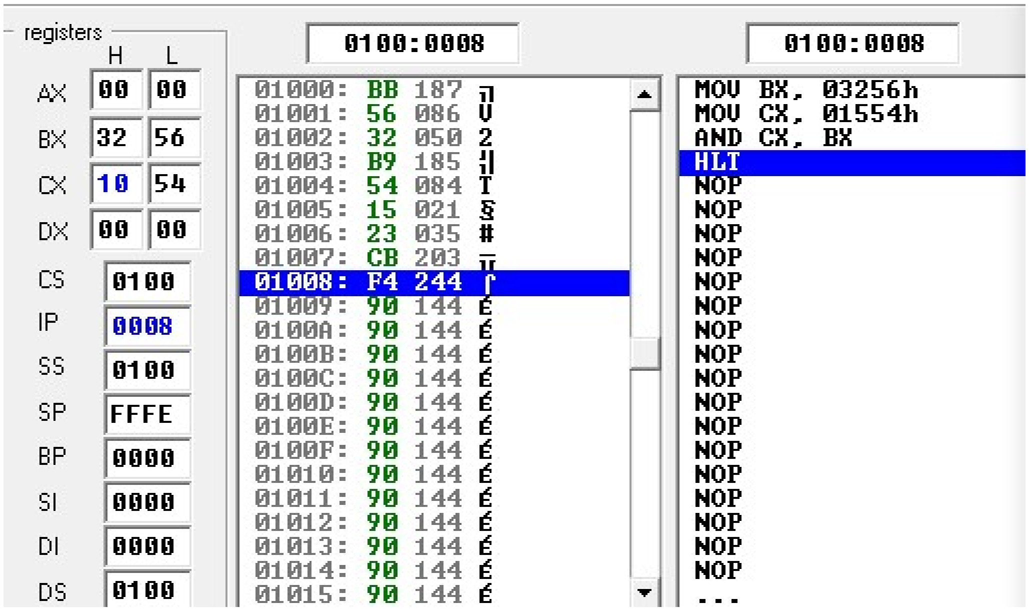
Exercise Part 1 (a)

Assembly Code:

```
CODE SEGMENT

   ASSUME CS:CODE, DS:CODE

   MOV BX, 3256H
   MOV CX, 1554H
   AND CX, BX

   HLT

CODE ENDS
    END
```

Analysis:

In the CX register, 1554H value is loaded. In the following instruction, the contents of CX and BX register are logically multiplied and the result put inside the CX register, giving the value of 1054H in CX register.



```
registers
      H   L
AX   00  00
BX   32  56
CX   10  54
DX   00  00
CS   0100
IP   0008
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
```

```
0100:0008

01000: BB 187 ╗
01001: 56 086 V
01002: 32 050 2
01003: B9 185 ╣
01004: 54 084 T
01005: 15 021 §
01006: 23 035 #
01007: CB 203 ╦
01008: F4 244 ⌠
01009: 90 144 É
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
```

```
0100:0008

MOV BX, 03256h
MOV CX, 01554h
AND CX, BX
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

Fig: Execution of AND operation

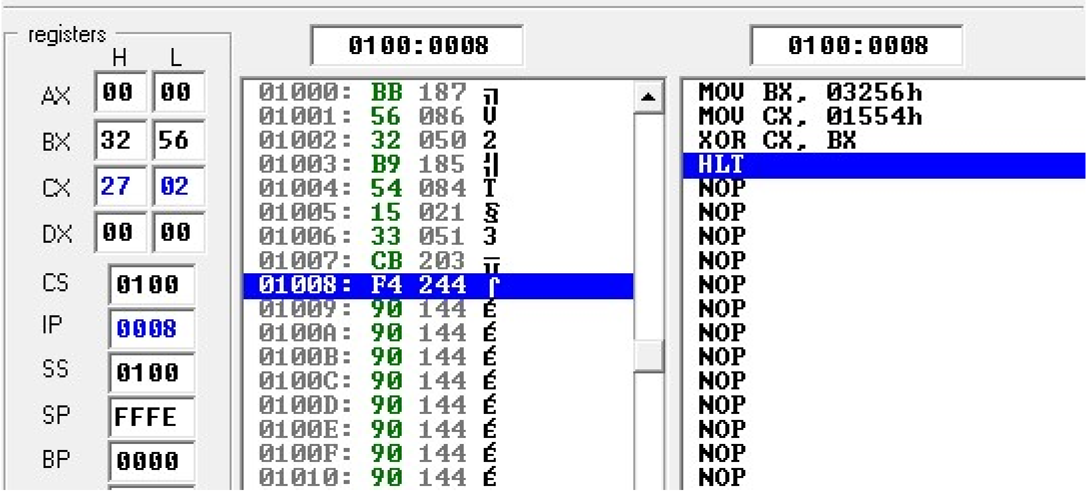
Exercise Part 1 (b)

Assembly Code:

```
CODE SEGMENT

   ASSUME CS:CODE, DS:CODE

   MOV BX, 3256H
   MOV CX, 1554H
   XOR CX, BX

   HLT

CODE ENDS
    END
```

Analysis:

XOR operation is performed on the contents of CX and BX register, and the result = 2702H is stored in the CX register



```
     H   L
AX   00  00
BX   32  56
CX   27  02
DX   00  00
CS   0100
IP   0008
SS   0100
SP   FFFE
BP   0000
```

```
0100:0008

01000: BB 187 ╗
01001: 56 086 V
01002: 32 050 2
01003: B9 185 ╣
01004: 54 084 T
01005: 15 021 §
01006: 33 051 3
01007: CB 203 ╦
01008: F4 244 ⌠
01009: 90 144 É
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
```

```
0100:0008

MOV BX, 03256h
MOV CX, 01554h
XOR CX, BX
HLT
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
```

Fig: Execution of XOR operation

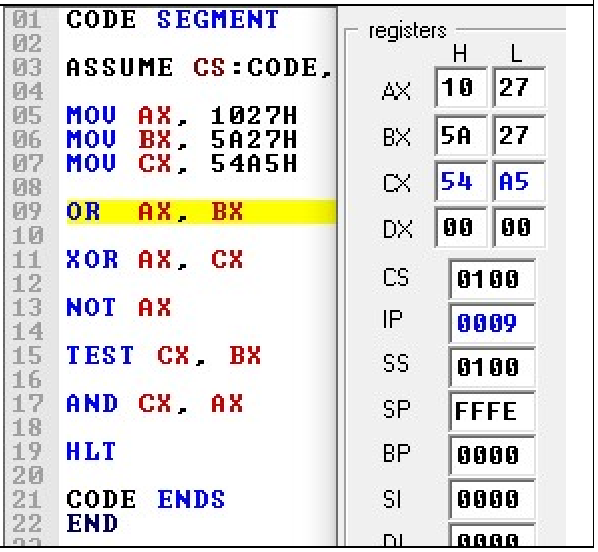
Exercise Part 1 (c)

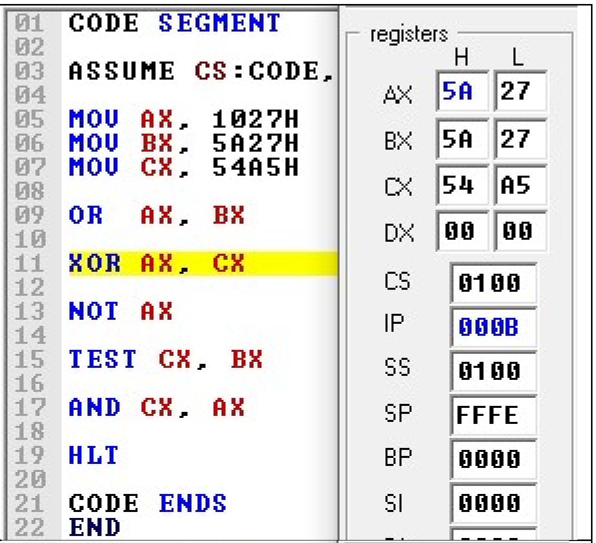
Assembly Code:

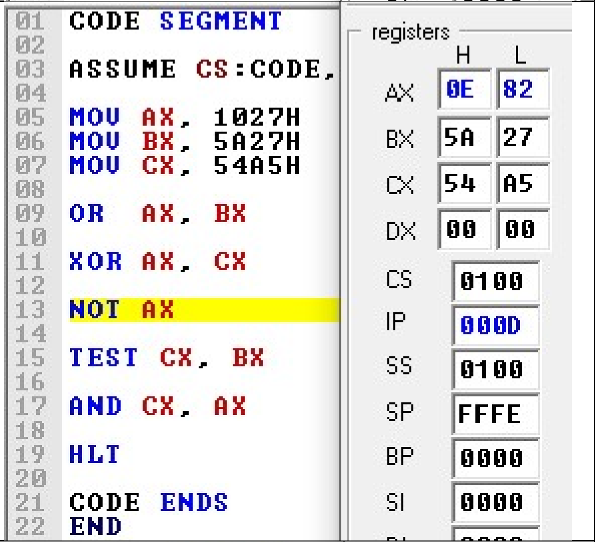
```
CODE SEGMENT

   ASSUME CS:CODE, DS:CODE

   MOV AX, 1027H
   MOV BX, 5A27H
   MOV CX, 54A5H

   OR  AX, BX     ; AX = AX | BX

   XOR AX, CX     ; AX = AX ^ CX

   NOT AX         ; AX = ~AX

   TEST CX, BX    ; check flags, result not stored

   AND CX, AX     ; CX = CX & AX

   HLT

CODE ENDS
    END
```

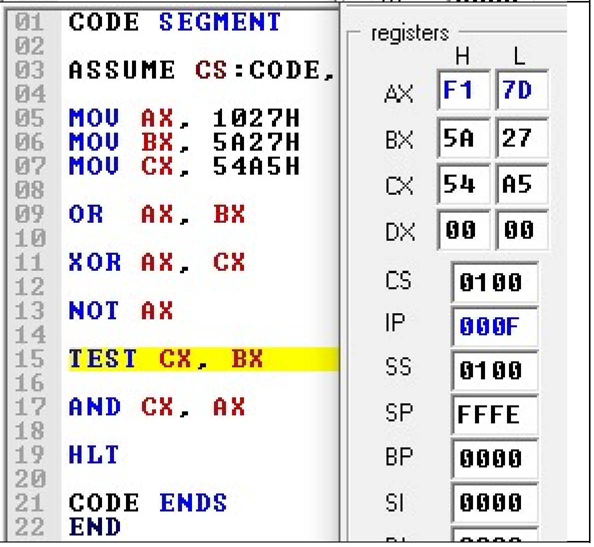
Analysis:

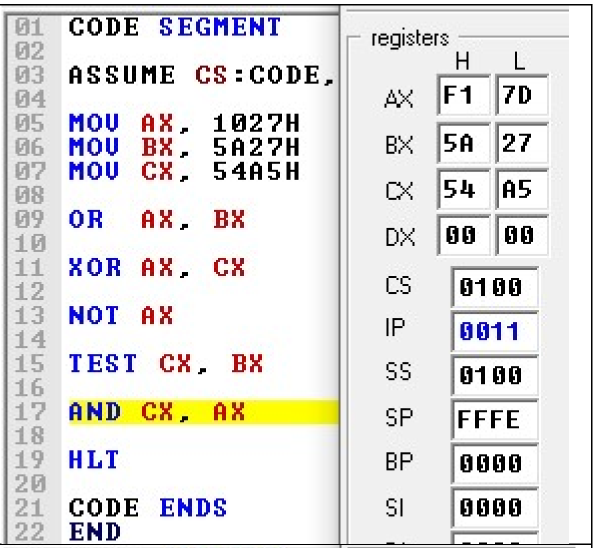
Code segment executed	Registers in Hex	Registers in Bin	Output

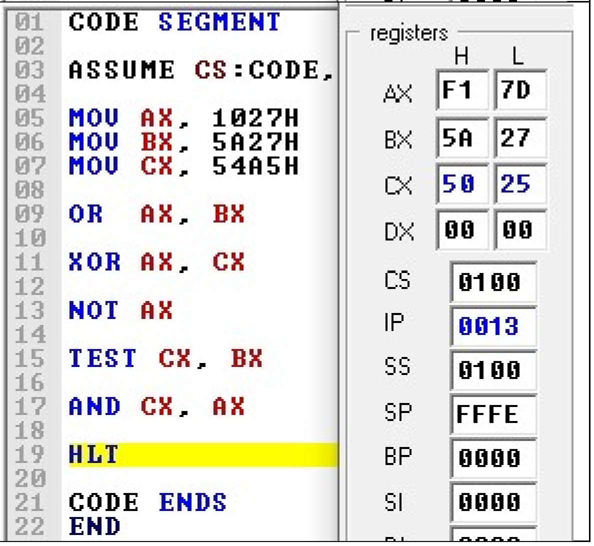
MOV AX, 1027H
MOV BX, 5A27H
MOV CX, 54A5H
	AX = 1027
	BX = 5A27
	CX = 54A5
		0001 0000 0010 0111
		0101 1010 0010 0111
		0101 0100 1010 0101
			
			CODE SEGMENT
			
			ASSUME CS:CODE,
			
			MOV AX, 1027H
			MOV BX, 5A27H
			MOV CX, 54A5H
			
			OR AX, BX
			
			XOR AX, CX
			
			NOT AX
			
			TEST CX, BX
			
			AND CX, AX
			
			HLT
			
			CODE ENDS
			END
			registers
			H L
			AX 10 27
			BX 5A 27
			CX 54 A5
			DX 00 00
			CS 0100
			IP 0009
			SS 0100
			SP FFFE
			BP 0000
			SI 0000
			DI 0000

OR AX, BX	AX = 5A27
	BX = 5A27
	CX = 54A5
		0101 1010 0010 0111
		0101 1010 0010 0111
		0101 0100 1010 0101
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 1027H
			MOV BX, 5A27H
			MOV CX, 54A5H
			OR AX, BX
			XOR AX, CX
			NOT AX
			TEST CX, BX
			AND CX, AX
			HLT
			CODE ENDS
			END
			registers
			H L
			AX 5A 27
			BX 5A 27
			CX 54 A5
			DX 00 00
			CS 0100
			IP 000B
			SS 0100
			SP FFFE
			BP 0000
			SI 0000

XOR AX, CX	AX = E82
	BX = 5A27
	CX = 54A5
		0000 1110 1000 0010
		0101 1010 0010 0111
		0101 0100 1010 0101
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 1027H
			MOV BX, 5A27H
			MOV CX, 54A5H
			OR AX, BX
			XOR AX, CX
			NOT AX
			TEST CX, BX
			AND CX, AX
			HLT
			CODE ENDS
			END
			registers
			H L
			AX 0E 82
			BX 5A 27
			CX 54 A5
			DX 00 00
			CS 0100
			IP 000D
			SS 0100
			SP FFFE
			BP 0000
			SI 0000

NOT AX	AX = F17D
	BX = 5A27
	CX = 54A5
		1111 0001 0111 1101
		0101 1010 0010 0111
		0101 0100 1010 0101
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 1027H
			MOV BX, 5A27H
			MOV CX, 54A5H
			OR AX, BX
			XOR AX, CX
			NOT AX
			TEST CX, BX
			AND CX, AX
			HLT
			CODE ENDS
			END
			registers
			H L
			AX F1 7D
			BX 5A 27
			CX 54 A5
			DX 00 00
			CS 0100
			IP 000F
			SS 0100
			SP FFFE
			BP 0000
			SI 0000

TEST CX, BX	AX = F17D
	BX = 5A27
	CX = 54A5
		1111 0001 0111 1101
		0101 1010 0010 0111
		0101 0100 1010 0101
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 1027H
			MOV BX, 5A27H
			MOV CX, 54A5H
			OR AX, BX
			XOR AX, CX
			NOT AX
			TEST CX, BX
			AND CX, AX
			HLT
			CODE ENDS
			END
			registers
			H L
			AX F1 7D
			BX 5A 27
			CX 54 A5
			DX 00 00
			CS 0100
			IP 0011
			SS 0100
			SP FFFE
			BP 0000
			SI 0000

AND CX, AX	AX = F17D
	BX = 5A27
	CX = 5025
		1111 0001 0111 1101
		0101 1010 0010 0111
		0101 0000 0010 0101
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 1027H
			MOV BX, 5A27H
			MOV CX, 54A5H
			OR AX, BX
			XOR AX, CX
			NOT AX
			TEST CX, BX
			AND CX, AX
			HLT
			CODE ENDS
			END
			registers
			H L
			AX F1 7D
			BX 5A 27
			CX 50 25
			DX 00 00
			CS 0100
			IP 0013
			SS 0100
			SP FFFE
			BP 0000
			SI 0000

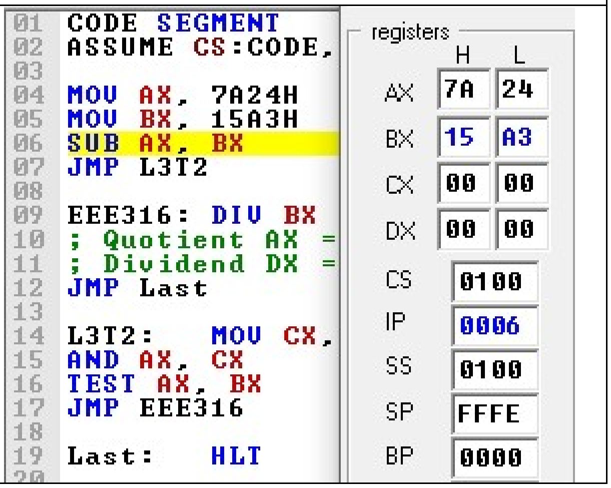
Exercise Part 2 (a)

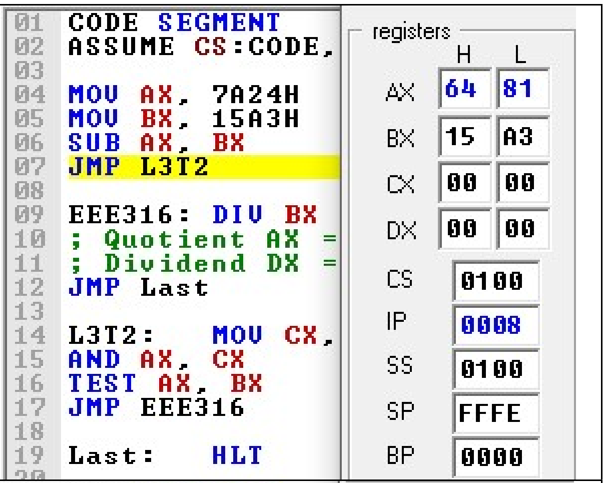
Assembly Code:

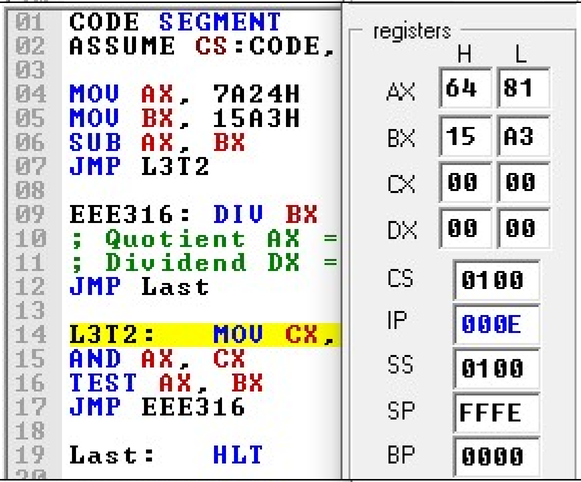
```
CODE SEGMENT
   ASSUME CS:CODE, DS:CODE

      MOV AX, 7A24H
      MOV BX, 15A3H

      SUB AX, BX      ; AX = 6481H after subtraction
      JMP L3T2

EEE316:
      DIV BX    ; Divide AX = 4481H by BX = 15A3H
                ; Quotient AX = 0003H
                ; Dividend DX = 0398H
      JMP Last

L3T2:   MOV CX, 45B1H

      AND AX, CX      ; AX = 4481H after AND
      TEST AX, BX
      JMP EEE316

Last:   HLT

CODE    ENDS
      END
```

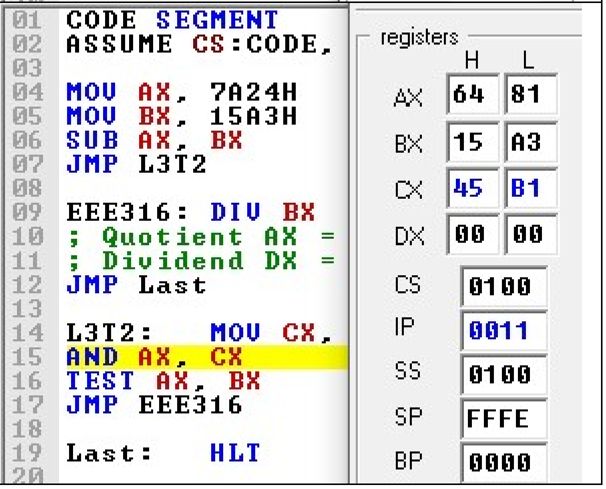
Analysis:

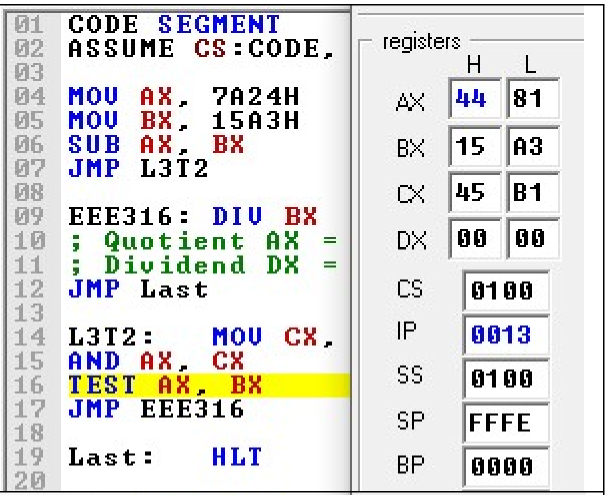
Code segment executed	Registers in Hex	Registers in Bin	Output

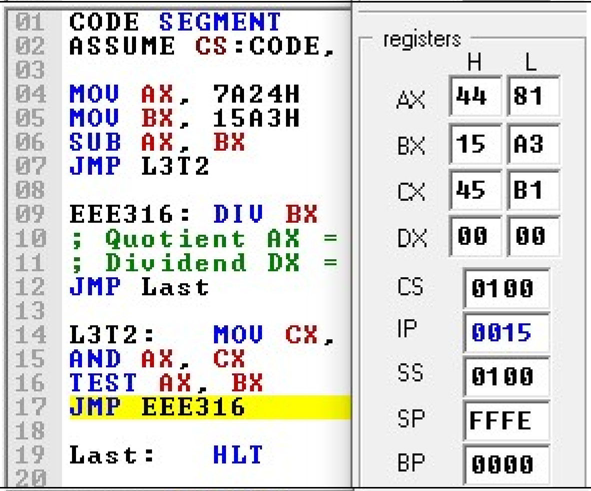
MOV AX, 7A24H
MOV BX, 15A3H
	AX = 7A24
	BX = 15A3
	CX = 0000
	DX = 0000
		0111 1010 0010 0100
		0001 0101 1010 0011
		0000 0000 0000 0000
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 15A3H
			SUB AX, BX
			JMP L3T2
			
			EEE316: DIV BX
			; Quotient AX =
			; Dividend DX =
			JMP Last
			
			L3T2: MOV CX,
			AND AX, CX
			TEST AX, BX
			JMP EEE316
			
			Last: HLT
			registers
			H L
			AX 7A 24
			BX 15 A3
			CX 00 00
			DX 00 00
			CS 0100
			IP 0006
			SS 0100
			SP FFFE
			BP 0000

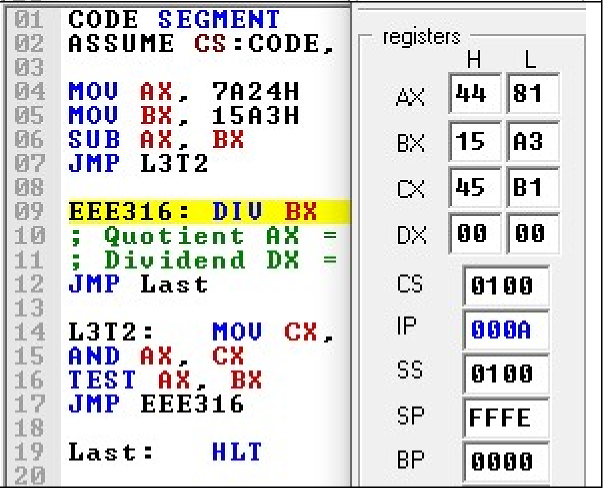
SUB AX, BX	AX = 6481
	BX = 15A3
	CX = 0000
	DX = 0000
		0110 0100 1000 0001
		0001 0101 1010 0011
		0000 0000 0000 0000
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 15A3H
			SUB AX, BX
			JMP L3T2
			
			EEE316: DIV BX
			; Quotient AX =
			; Dividend DX =
			JMP Last
			
			L3T2: MOV CX,
			AND AX, CX
			TEST AX, BX
			JMP EEE316
			
			Last: HLT
			registers
			H L
			AX 64 81
			BX 15 A3
			CX 00 00
			DX 00 00
			CS 0100
			IP 0008
			SS 0100
			SP FFFE
			BP 0000

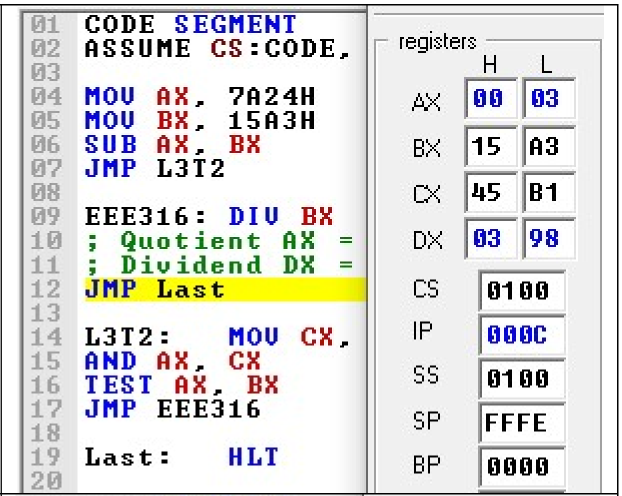
JMP L3T2	AX = 6481
	BX = 15A3
	CX = 0000
	DX = 0000
		0110 0100 1000 0001
		0001 0101 1010 0011
		0000 0000 0000 0000
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 15A3H
			SUB AX, BX
			JMP L3T2
			
			EEE316: DIV BX
			; Quotient AX =
			; Dividend DX =
			JMP Last
			
			L3T2: MOV CX,
			AND AX, CX
			TEST AX, BX
			JMP EEE316
			
			Last: HLT
			registers
			H L
			AX 64 81
			BX 15 A3
			CX 00 00
			DX 00 00
			CS 0100
			IP 000E
			SS 0100
			SP FFFE
			BP 0000

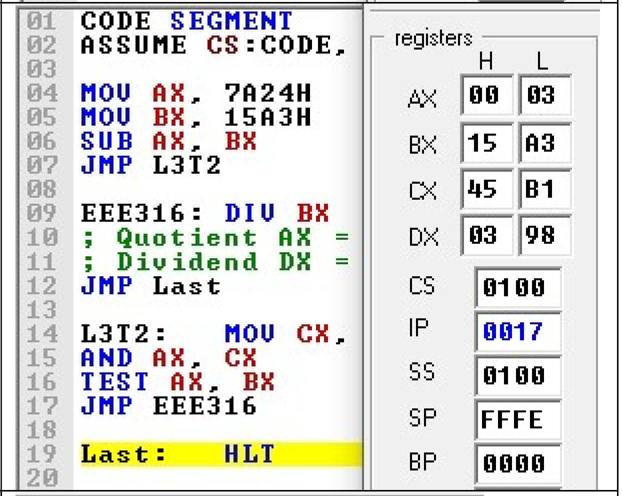
L3T2: MOV CX, 45B1H	AX = 6481
	BX = 15A3
	CX = 45B1
	DX = 0000
		0110 0100 1000 0001
		0001 0101 1010 0011
		0100 0101 1011 0001
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 15A3H
			SUB AX, BX
			JMP L3T2
			
			EEE316: DIV BX
			; Quotient AX =
			; Dividend DX =
			JMP Last
			
			L3T2: MOV CX,
			AND AX, CX
			TEST AX, BX
			JMP EEE316
			
			Last: HLT
			registers
			H L
			AX 64 81
			BX 15 A3
			CX 45 B1
			DX 00 00
			CS 0100
			IP 0011
			SS 0100
			SP FFFE
			BP 0000

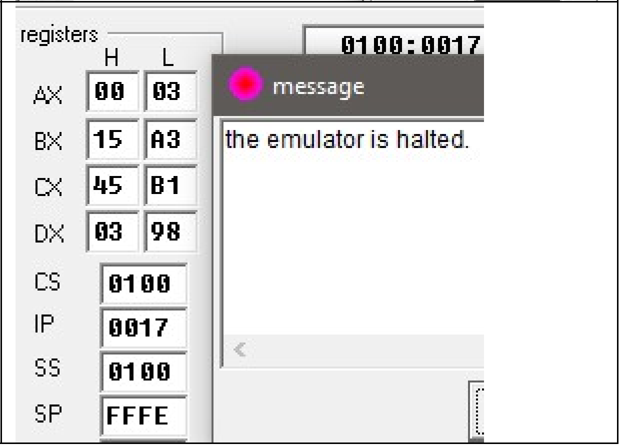
AND AX, CX	AX = 4481
	BX = 15A3
	CX = 45B1
	DX = 0000
		0100 0100 1000 0001
		0001 0101 1010 0011
		0100 0101 1011 0001
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 15A3H
			SUB AX, BX
			JMP L3T2
			
			EEE316: DIV BX
			; Quotient AX =
			; Dividend DX =
			JMP Last
			
			L3T2: MOV CX,
			AND AX, CX
			TEST AX, BX
			JMP EEE316
			
			Last: HLT
			
			registers
			H L
			AX 44 81
			BX 15 A3
			CX 45 B1
			DX 00 00
			CS 0100
			IP 0013
			SS 0100
			SP FFFE
			BP 0000

TEST AX, BX	AX = 4481
	BX = 15A3
	CX = 45B1
	DX = 0000
		0100 0100 1000 0001
		0001 0101 1010 0011
		0100 0101 1011 0001
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 15A3H
			SUB AX, BX
			JMP L3T2
			
			EEE316: DIV BX
			; Quotient AX =
			; Dividend DX =
			JMP Last
			
			L3T2: MOV CX,
			AND AX, CX
			TEST AX, BX
			JMP EEE316
			
			Last: HLT
			
			registers
			H L
			AX 44 81
			BX 15 A3
			CX 45 B1
			DX 00 00
			CS 0100
			IP 0015
			SS 0100
			SP FFFE
			BP 0000

JMP EEE316	AX = 4481
	BX = 15A3
	CX = 45B1
	DX = 0000
		0100 0100 1000 0001
		0001 0101 1010 0011
		0100 0101 1011 0001
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 15A3H
			SUB AX, BX
			JMP L3T2
			
			EEE316: DIV BX
			; Quotient AX =
			; Dividend DX =
			JMP Last
			
			L3T2: MOV CX,
			AND AX, CX
			TEST AX, BX
			JMP EEE316
			
			Last: HLT
			
			registers
			H L
			AX 44 81
			BX 15 A3
			CX 45 B1
			DX 00 00
			CS 0100
			IP 000A
			SS 0100
			SP FFFE
			BP 0000

EEE316:
DIV BX
	AX = 0003
	BX = 15A3
	CX = 45B1
	DX = 0398
		0000 0000 0000 0011
		0001 0101 1010 0011
		0100 0101 1011 0001
		0000 0011 1001 1000
			

JMP Last	AX = 0003
	BX = 15A3
	CX = 45B1
	DX = 0398
		0000 0000 0000 0011
		0001 0101 1010 0011
		0100 0101 1011 0001
		0000 0011 1001 1000
			

Last: HLT	AX = 0003
	BX = 15A3
	CX = 45B1
	DX = 0398
		0000 0000 0000 0011
		0001 0101 1010 0011
		0100 0101 1011 0001
		0000 0011 1001 1000
			

```
CODE SEGMENT
ASSUME CS:CODE,

MOV AX, 7A24H
MOV BX, 15A3H
SUB AX, BX
JMP L3T2

EEE316: DIV BX
; Quotient AX =
; Dividend DX =
JMP Last

L3T2:   MOV CX,
AND AX, CX
TEST AX, BX
JMP EEE316

Last:   HLT
```

```
registers
     H   L
AX   00  03
BX   15  A3
CX   45  B1
DX   03  98
CS   0100
IP   000C
SS   0100
SP   FFFE
BP   0000
```

```
CODE SEGMENT
ASSUME CS:CODE,

MOV AX, 7A24H
MOV BX, 15A3H
SUB AX, BX
JMP L3T2

EEE316: DIV BX
; Quotient AX =
; Dividend DX =
JMP Last

L3T2:   MOV CX,
AND AX, CX
TEST AX, BX
JMP EEE316

Last:   HLT
```

```
registers
     H   L
AX   00  03
BX   15  A3
CX   45  B1
DX   03  98
CS   0100
IP   0017
SS   0100
SP   FFFE
BP   0000
```

```
registers
     H   L
AX   00  03
BX   15  A3
CX   45  B1
DX   03  98
CS   0100
IP   0017
SS   0100
SP   FFFE

0100:0017
message
the emulator is halted.
```

The last label is required because the way the program is structured, L3T2 is jumped to first, and then L3T2 directs the pointer to EEE316 which is above itself in the instruction list. After EEE316, the program counter will automatically point to the location that is below EEE316, incidentally reaching L3T2 and looping here forever. This is why the pointer is required to jump to Last where the instruction register receives halt command and stops execution.

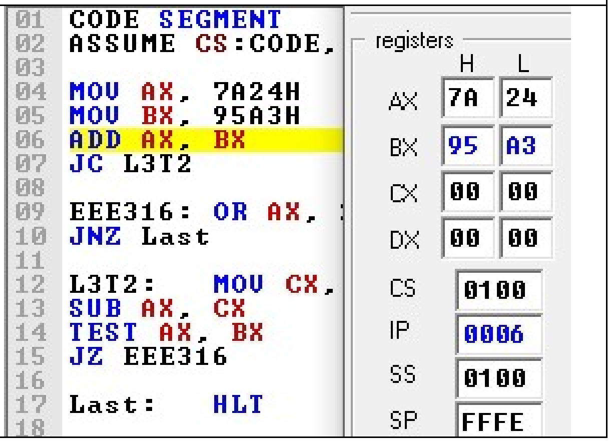
Exercise Part 2 (b)

Assembly Code:

```
CODE SEGMENT
   ASSUME CS:CODE, DS:CODE

      MOV AX, 7A24H
      MOV BX, 95A3H
      ADD AX, BX
      JC L3T2        ; Jump to L3T2 if CF=1

EEE316: OR AX, 23H
      JNZ Last       ; Jump to Last if previous operation not zero

L3T2:   MOV CX, 0FC7H
      SUB AX, CX
      TEST AX, BX
      JZ EEE316      ; Jump to EEE316 if previous operation zero

Last:   HLT

CODE    ENDS
      END
```

Analysis:

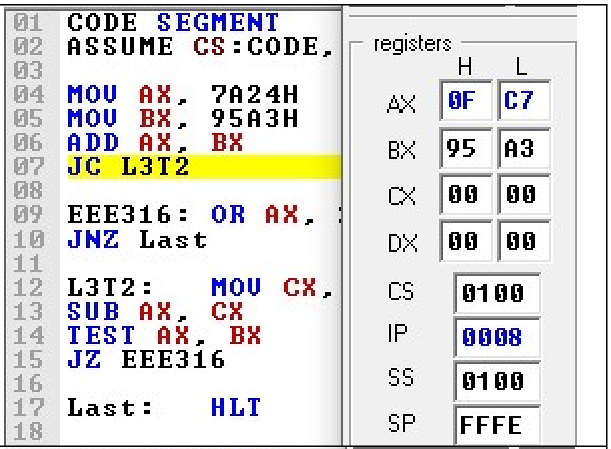
Code segment executed	Registers in Hex	Registers in Bin	Output

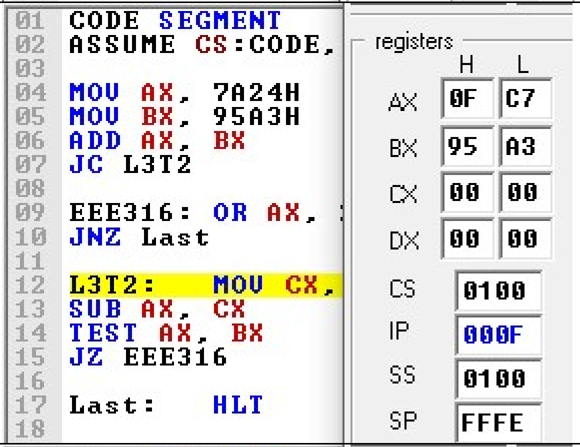
MOV AX, 7A24H
MOV BX, 95A3H
	AX = 7A24
	BX = 95A3
	CX = 0000
		0111 1010 0010 0100
		1001 0101 1010 0011
		0000 0000 0000 0000
			

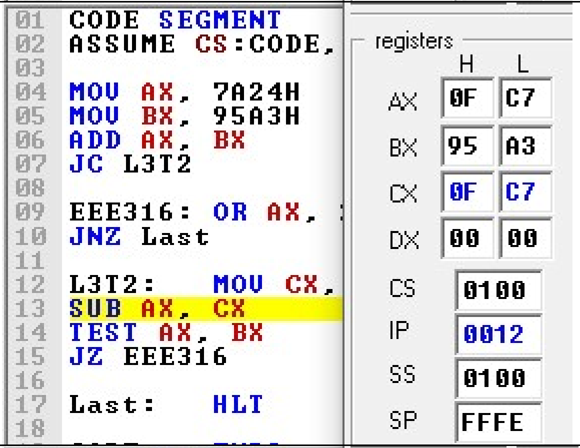
```
CODE SEGMENT
ASSUME CS:CODE,

MOV AX, 7A24H
MOV BX, 95A3H
ADD AX, BX
JC L3T2

EEE316: OR AX, :
JNZ Last

L3T2:    MOV CX,
SUB AX, CX
TEST AX, BX
JZ EEE316

Last:    HLT
```

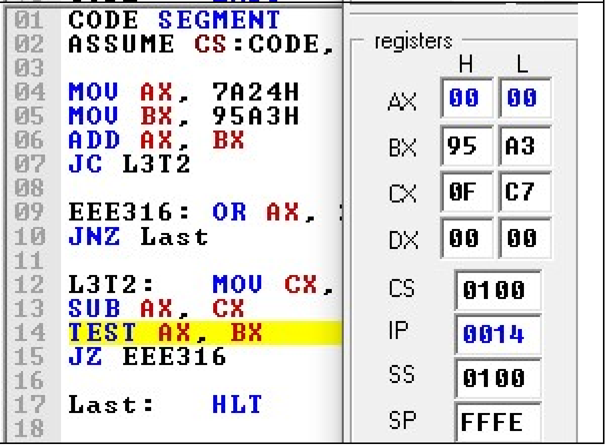
registers

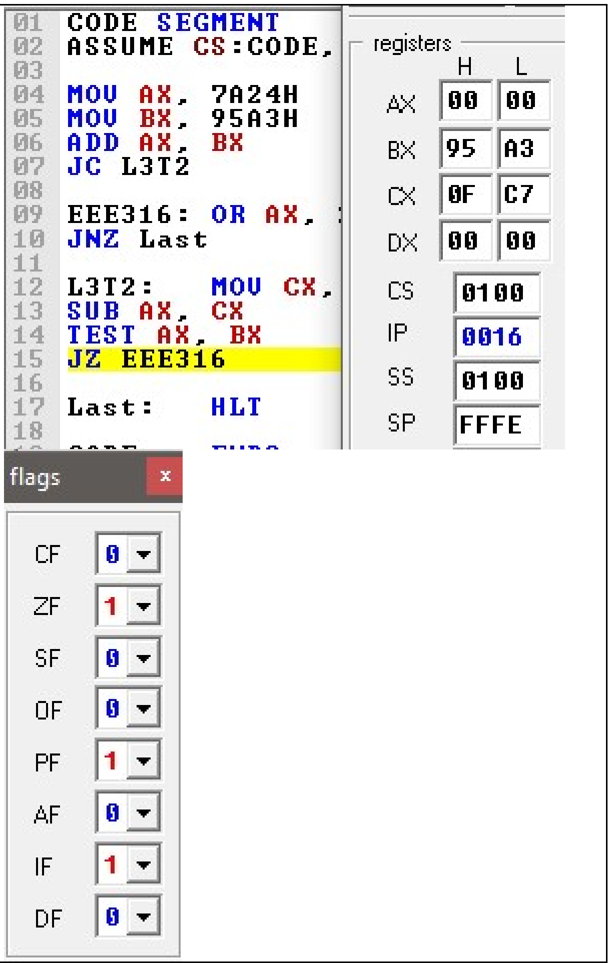
	H	L
AX	7A	24
BX	95	A3
CX	00	00
DX	00	00
CS	0100	
IP	0006	
SS	0100	
SP	FFFE	

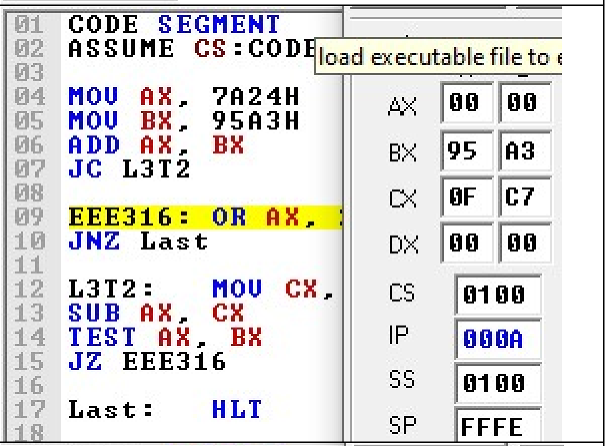
ADD AX, BX	AX = 0FC7
	BX = 95A3
	CX = 0000
		0000 1111 1100 0111
		1001 0101 1010 0011
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 7A24H
			MOV BX, 95A3H
			ADD AX, BX
			JC L3T2
			EEE316: OR AX,
			JNZ Last
			L3T2: MOV CX,
			SUB AX, CX
			TEST AX, BX
			JZ EEE316
			Last: HLT
			registers H L
			AX 0F C7
			BX 95 A3
			CX 00 00
			DX 00 00
			CS 0100
			IP 0008
			SS 0100
			SP FFFE

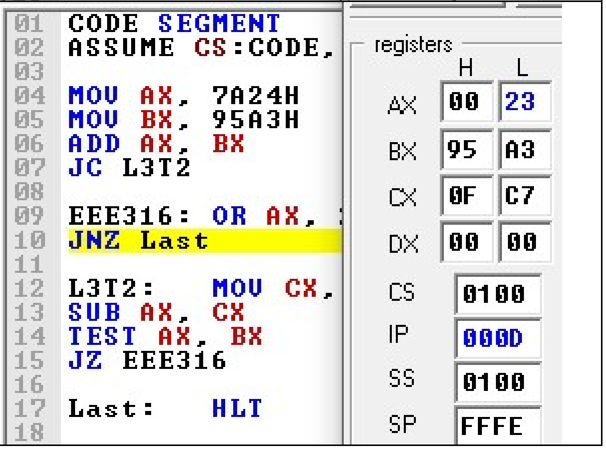
JC L3T2	AX = 0FC7
	BX = 95A3
	CX = 0000
		0000 1111 1100 0111
		1001 0101 1010 0011
		0000 0000 0000 0000
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 7A24H
			MOV BX, 95A3H
			ADD AX, BX
			JC L3T2
			EEE316: OR AX,
			JNZ Last
			L3T2: MOV CX,
			SUB AX, CX
			TEST AX, BX
			JZ EEE316
			Last: HLT
			registers H L
			AX 0F C7
			BX 95 A3
			CX 00 00
			DX 00 00
			CS 0100
			IP 000F
			SS 0100
			SP FFFE

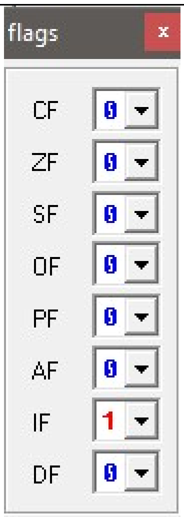
L3T2: MOV CX, 0FC7H	AX = 0FC7
	BX = 95A3
	CX = 0FC7
		0000 1111 1100 0111
		1001 0101 1010 0011
		0000 1111 1100 0111
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 7A24H
			MOV BX, 95A3H
			ADD AX, BX
			JC L3T2
			EEE316: OR AX,
			JNZ Last
			L3T2: MOV CX,
			SUB AX, CX
			TEST AX, BX
			JZ EEE316
			Last: HLT
			registers H L
			AX 0F C7
			BX 95 A3
			CX 0F C7
			DX 00 00
			CS 0100
			IP 0012
			SS 0100
			SP FFFE

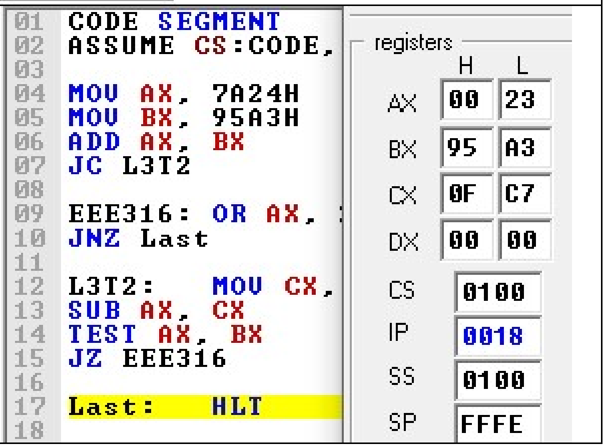
SUB AX, CX	AX = 0000
	BX = 95A3
	CX = 0FC7
		0000 0000 0000 0000
		1001 0101 1010 0011
		0000 1111 1100 0111
			
			CODE SEGMENT
			ASSUME CS:CODE,
			MOV AX, 7A24H
			MOV BX, 95A3H
			ADD AX, BX
			JC L3T2
			EEE316: OR AX,
			JNZ Last
			L3T2: MOV CX,
			SUB AX, CX
			TEST AX, BX
			JZ EEE316
			Last: HLT
			registers H L
			AX 00 00
			BX 95 A3
			CX 0F C7
			DX 00 00
			CS 0100
			IP 0014
			SS 0100
			SP FFFE

TEST AX, BX	AX = 0000
	BX = 95A3
	CX = 0FC7
		0000 0000 0000 0000
		1001 0101 1010 0011
		0000 1111 1100 0111
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 95A3H
			ADD AX, BX
			JC L3T2
			
			EEE316: OR AX, :
			JNZ Last
			
			L3T2: MOV CX,
			SUB AX, CX
			TEST AX, BX
			JZ EEE316
			
			Last: HLT
			
			registers H L
			AX 00 00
			BX 95 A3
			CX 0F C7
			DX 00 00
			CS 0100
			IP 0016
			SS 0100
			SP FFFE
			
			flags
			CF 0
			ZF 1
			SF 0
			OF 0
			PF 1
			AF 0
			IF 1
			DF 0

JZ EEE316	AX = 0000
	BX = 95A3
	CX = 0FC7
		0000 0000 0000 0000
		1001 0101 1010 0011
		0000 1111 1100 0111
			
			CODE SEGMENT
			ASSUME CS:CODE
			
			MOV AX, 7A24H
			MOV BX, 95A3H
			ADD AX, BX
			JC L3T2
			
			EEE316: OR AX, :
			JNZ Last
			
			L3T2: MOV CX,
			SUB AX, CX
			TEST AX, BX
			JZ EEE316
			
			Last: HLT
			
			load executable file to e
			AX 00 00
			BX 95 A3
			CX 0F C7
			DX 00 00
			CS 0100
			IP 000A
			SS 0100
			SP FFFE

EEE316: OR AX, 23H	AX = 0023
	BX = 95A3
	CX = 0FC7
		0000 0000 0010 0011
		1001 0101 1010 0011
		0000 1111 1100 0111
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 95A3H
			ADD AX, BX
			JC L3T2
			
			EEE316: OR AX, :
			JNZ Last
			
			L3T2: MOV CX,
			SUB AX, CX
			TEST AX, BX
			JZ EEE316
			
			Last: HLT
			
			registers H L
			AX 00 23
			BX 95 A3
			CX 0F C7
			DX 00 00
			CS 0100
			IP 000D
			SS 0100
			SP FFFE

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

JNZ Last	AX = 0023
	BX = 95A3
	CX = 0FC7
		0000 0000 0010 0011
		1001 0101 1010 0011
		0000 1111 1100 0111
			
			CODE SEGMENT
			ASSUME CS:CODE,
			
			MOV AX, 7A24H
			MOV BX, 95A3H
			ADD AX, BX
			JC L3T2
			
			EEE316: OR AX,
			JNZ Last
			
			L3T2: MOV CX,
			SUB AX, CX
			TEST AX, BX
			JZ EEE316
			
			Last: HLT
			
			registers
			H L
			AX 00 23
			BX 95 A3
			CX 0F C7
			DX 00 00
			CS 0100
			IP 0018
			SS 0100
			SP FFFE

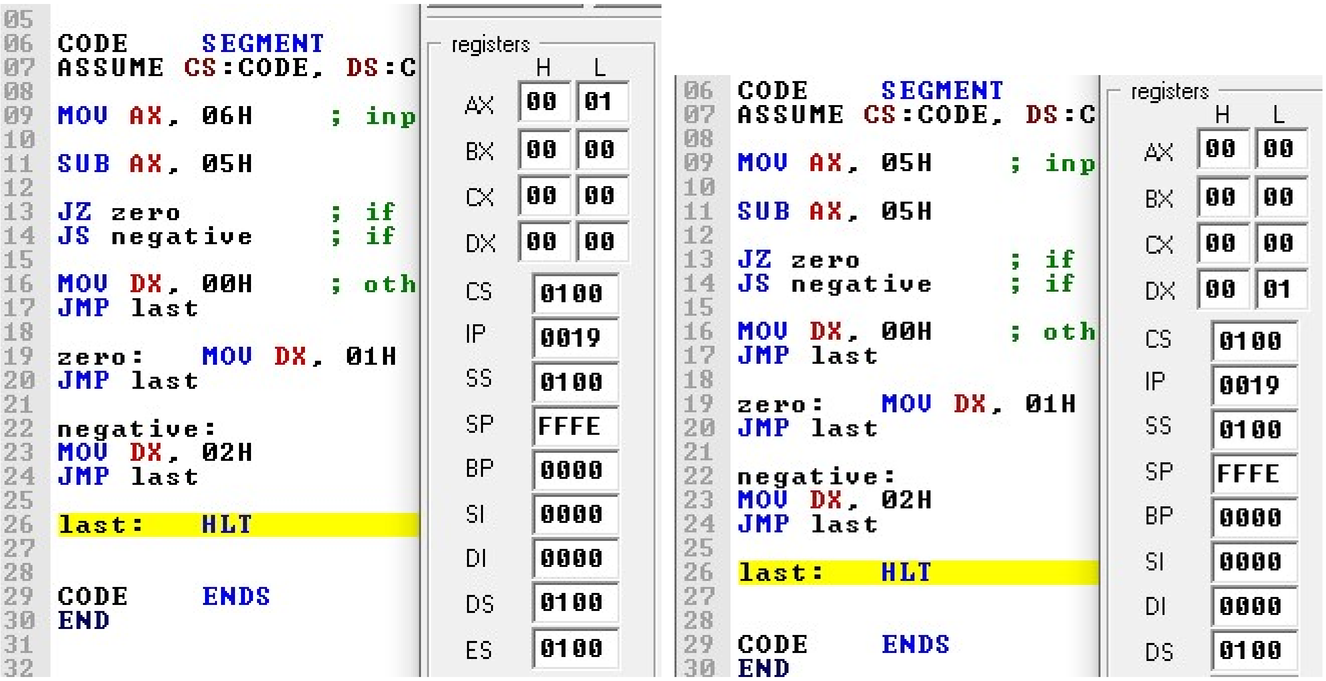
Homework 1

Write an assembly code that will determine whether a number is greater than 5, equal to or less, and put 0, 1 or 2 for the conditions in DX

Assembly Code:

```
CODE    SEGMENT
   ASSUME CS:CODE, DS:CODE

        MOV AX, 06H    ; input number

        SUB AX, 05H

        JZ zero        ; if result was zero
        JS negative    ; if result was negative

        MOV DX, 00H    ; otherwise
        JMP last

zero:   MOV DX, 01H
        JMP last

negative:
        MOV DX, 02H
        JMP last

last:   HLT

CODE    ENDS
        END
```

Output:



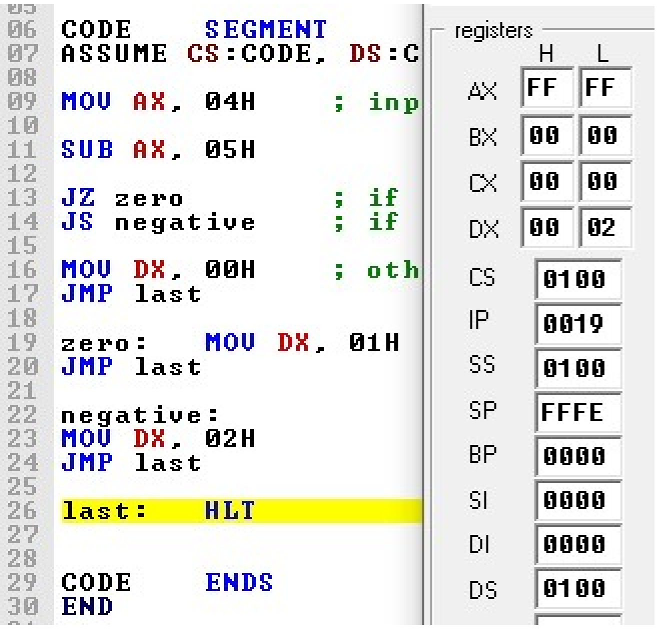
```
CODE    SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 06H     ; inp

SUB AX, 05H

JZ zero         ; if
JS negative     ; if

MOV DX, 00H     ; oth
JMP last

zero:   MOV DX, 01H
JMP last

negative:
MOV DX, 02H
JMP last

last:   HLT


CODE    ENDS
END
```

```
registers
      H    L
AX   00   01
BX   00   00
CX   00   00
DX   00   00
CS   0100
IP   0019
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
ES   0100
```

```
CODE    SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 05H     ; inp

SUB AX, 05H

JZ zero         ; if
JS negative     ; if

MOV DX, 00H     ; oth
JMP last

zero:   MOV DX, 01H
JMP last

negative:
MOV DX, 02H
JMP last

last:   HLT


CODE    ENDS
END
```

```
registers
      H    L
AX   00   00
BX   00   00
CX   00   00
DX   00   01
CS   0100
IP   0019
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
```

Fig: (a) AX = 6, DX = 0 after execution (b) AX = 5, DX = 1 after execution



```
CODE    SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 04H     ; inp

SUB AX, 05H

JZ zero         ; if
JS negative     ; if

MOV DX, 00H     ; oth
JMP last

zero:   MOV DX, 01H
JMP last

negative:
MOV DX, 02H
JMP last

last:   HLT


CODE    ENDS
END
```

```
registers
      H    L
AX   FF   FF
BX   00   00
CX   00   00
DX   00   02
CS   0100
IP   0019
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
DS   0100
```

Fig: (c) AX = 4, DX = 2 after execution

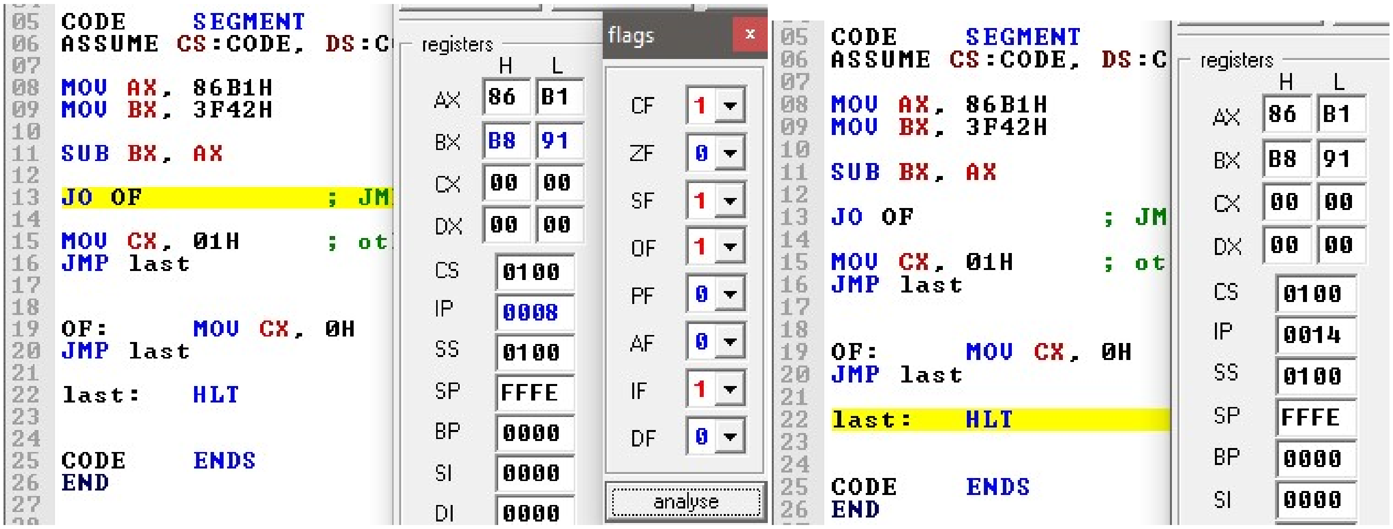
Homework 2

Subtract 86B1H from 3F42H and store 0 in CX if overflow occurs and 1 if no overflow occurs

Assembly Code:

```
CODE    SEGMENT
   ASSUME CS:CODE, DS:CODE

      MOV AX, 86B1H
      MOV BX, 3F42H
      SUB BX, AX

      JO OF          ; JMP if overflow

      MOV CX, 01H     ; otherwise
      JMP last

OF:    MOV CX, 0H
      JMP last

last:   HLT
CODE    ENDS
      END
```

Output:



```
CODE     SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 86B1H
MOV BX, 3F42H

SUB BX, AX

JO OF                ; JM

MOV CX, 01H      ; ot
JMP last


OF:      MOV CX, 0H
JMP last

last:    HLT


CODE     ENDS
END
```

```
registers
      H    L
AX   86   B1
BX   B8   91
CX   00   00
DX   00   00
CS   0100
IP   0008
SS   0100
SP   FFFE
BP   0000
SI   0000
DI   0000
```

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

```
CODE     SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 86B1H
MOV BX, 3F42H

SUB BX, AX

JO OF                ; JM

MOV CX, 01H      ; ot
JMP last


OF:      MOV CX, 0H
JMP last

last:    HLT


CODE     ENDS
END
```

```
registers
      H    L
AX   86   B1
BX   B8   91
CX   00   00
DX   00   00
CS   0100
IP   0014
SS   0100
SP   FFFE
BP   0000
SI   0000
```

Fig: (a) Subtraction of AX from BX, overflow flag 1 (b) CX assumes value of 0

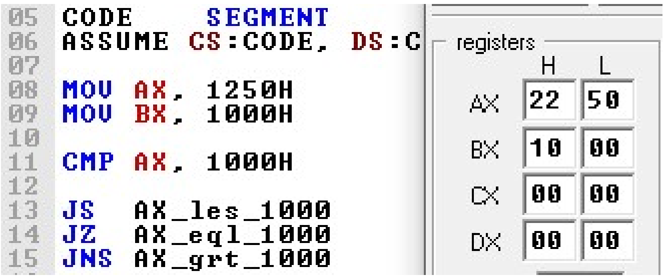
Homework 3

Take 2 arbitrary numbers x and y. If x>1000H perform x+y. If y<1000H perform x-y. If x>1000H and y<100H perform x = x’.

Assembly Code:

```
CODE    SEGMENT
   ASSUME CS:CODE, DS:CODE

      MOV AX, 0250H
      MOV BX, 0050H

      CMP AX, 1000H      ; Compares 1st val with 1000H

      JS  AX_les_1000     ; AX<1000
      JZ  AX_eql_1000     ; AX=1000
      JNS AX_grt_1000     ; AX>1000

AX_grt_1000:

      CMP BX, 100H       ; Compares 2nd val with 100H

      JS  AX_grt_1000_BX_les_100  ; AX > 1000 & BX < 100
      JZ  AX_grt_1000_BX_eql_100  ; AX > 1000 & BX = 100
      JNS AX_grt_1000_BX_grt_100  ; AX > 1000 & BX > 100

AX_les_1000:
AX_eql_1000:

      CMP BX, 1000H      ; Compares 2nd val with 1000H

      JS  BX_les_1000     ; BX<1000
      JNS last           ; AX<=1000 & BX >=1000

; 1st CASE
AX_grt_1000_BX_eql_100:
AX_grt_1000_BX_grt_100:

      ADD AX, BX
      JMP last

; 2nd CASE
BX_les_1000:
        SUB AX, BX
        JMP last

; 3rd CASE
AX_grt_1000_BX_les_100:

        NOT AX
        JMP last

last:   HLT

CODE    ENDS
        END
```

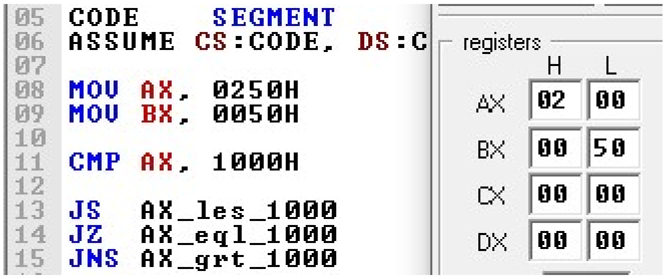
Output:



```
CODE    SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 1250H
MOV BX, 1000H

CMP AX, 1000H

JS  AX_les_1000
JZ  AX_eql_1000
JNS AX_grt_1000
```

```
registers
     H   L
AX   22  50
BX   10  00
CX   00  00
DX   00  00
```

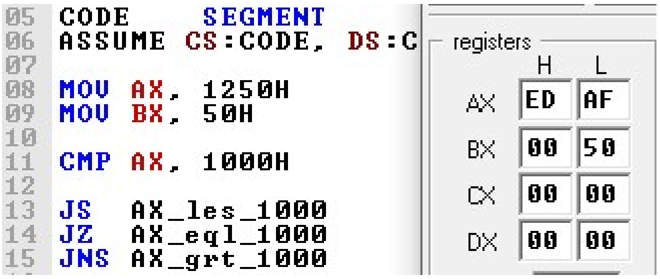
Fig: AX > 1000H, BX !<100H (1st condition applicable, x+y operation)



```
CODE    SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 0250H
MOV BX, 0050H

CMP AX, 1000H

JS  AX_les_1000
JZ  AX_eql_1000
JNS AX_grt_1000
```

```
registers
     H   L
AX   02  00
BX   00  50
CX   00  00
DX   00  00
```

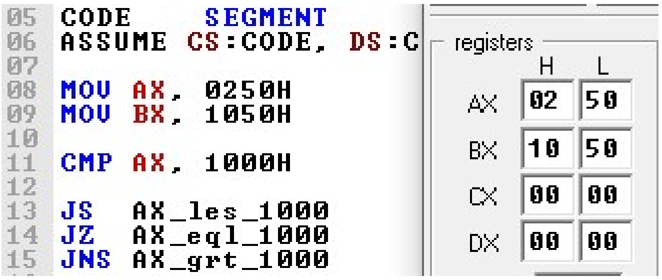
Fig: AX < 1000H, BX < 1000H (2nd Condition applicable, x-y operation)



```
CODE    SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 1250H
MOV BX, 50H

CMP AX, 1000H

JS  AX_les_1000
JZ  AX_eql_1000
JNS AX_grt_1000
```

```
registers
     H   L
AX   ED  AF
BX   00  50
CX   00  00
DX   00  00
```

Fig: AX > 1000H, BX < 100H (3rd condition applicable, x = x’ operation)



```
CODE     SEGMENT
ASSUME CS:CODE, DS:C

MOV AX, 0250H
MOV BX, 1050H

CMP AX, 1000H

JS   AX_les_1000
JZ   AX_eql_1000
JNS  AX_grt_1000
```

registers

	H	L
AX	02	50
BX	10	50
CX	00	00
DX	00	00

Fig: AX < 1000H, BX > 1000H (No condition applicable, no operation)

Homework 4

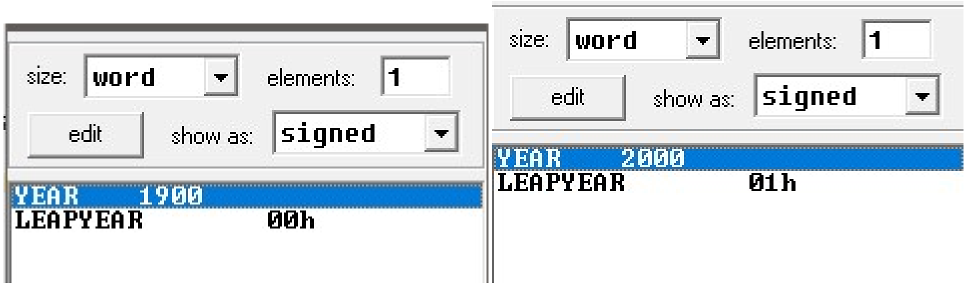
Write an assembly code that checks if a year is a leap year. Code template is shown below. If ‘YEAR’ is a leap year, put 1 in ‘LEAPYEAR’. Else put 0 in ‘LEAPYEAR’. You may observe value of LEAPYEAR by pressing the “var” button, beside the “flag” button.

Assembly Code:

```
CODE    SEGMENT
   ASSUME CS:CODE, DS:CODE

         MOV AX, CS
         MOV DS, AX

         MOV AX, YEAR
         MOV DX, 0

         ; *************SOLUTION CODE HERE*************

         ; DIVIDE BY 400

         MOV CX, AX      ; creates a copy of the main year
         MOV BX, 0400D   ; to divide by 400 first

         DIV BX
         CMP DX, 0H      ; compare dividend with 0

         JZ  positive    ; leap year if divided by 400

         ; DIVIDE BY 100

         MOV AX, CX      ; restore value of AX
         MOV DX, 0       ; making sure dividend 0
         MOV BX, 0100D   ; to divide by 100

         DIV BX
         CMP DX, 0H      ; compare dividend with 0

         JZ  negative    ; not leap year if divided by 100

         ; DIVIDE BY 4

         MOV AX, CX      ; restore value of AX
         MOV DX, 0       ; making sure dividend 0
         MOV BX, 04D     ; to divide by 4
```

```
        DIV BX
        CMP DX, 0H       ; compare dividend with 0

        JZ  positive    ;leap year if divided by 4
        JNZ negative    ; not leap year

positive:

        MOV LEAPYEAR, 01H
        JMP terminate

negative:

        MOV LEAPYEAR, 0H
        JMP terminate

terminate:

        ; ************** END OF SOLUTION *************

        HLT

        YEAR DW 2021D   ; DW = Data Word (16 bits)
        LEAPYEAR DB ?   ; DB = Data Byte (8 bits)

CODE    ENDS
        END
```

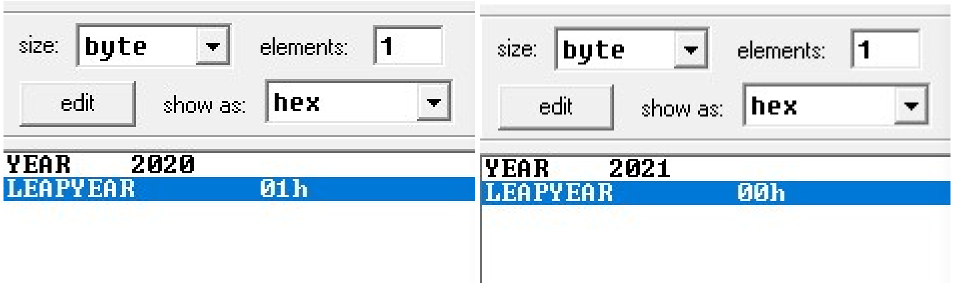
Output:



```
size: word      elements: 1
edit            show as: signed

YEAR     1900
LEAPYEAR          00h
```

```
size: word      elements: 1
edit            show as: signed

YEAR     2000
LEAPYEAR          01h
```

Fig: (a) 1900 not leap year (b) 2000 leap year



```
size: byte      elements: 1
edit            show as: hex

YEAR     2020
LEAPYEAR          01h
```

```
size: byte      elements: 1
edit            show as: hex

YEAR     2021
LEAPYEAR          00h
```

Fig: (c) 2020 leap year (d) 2021 not leap year