ELE 338 – Preliminary Work 3

Anıl Karaca – 21728405

Q1)

Source Code:

```
org 100h
;ELE338 - Preliminary Work 3 - Question 1
;Anil Karaca - 21728405
MOV AX, 00000h ;Expected result:1 2 3 4
CALL findZero
MOV AX, 0D000h ;Expected result:2 3 4
CALL findZero
MOV AX, 00F0Ah ;Expected result:1 3
CALL findZero
MOV AX, 010F7h ;Expected result:2
CALL findZero
MOV AX, 0DDDDh ;Expected result:
CALL findZero
MOV AX, 02007h ;Expected result:2 3
CALL findZero

ret

findZero PROC

MOV SI, 1d ;Initialize the location counter SI
MOV BX, AX ;Assign AX to BX so we can freely work on BX

checkNibble:
CMP SI, 5d ;Check if we are done with the loop
JZ terminate

SHR BX, 12 ;Leave the left most byte alone by shifting to the right
CMP BX, 0d ;Check if that byte is zero
JZ display ;If it is zero display it's location
continue:
INC SI ;Increment the location counter
ROL AX, 4 ;Rotate the AX to the left
MOV BX, AX ;Assign AX to BX so we can freely work on BX
JMP checkNibble ;Jump to checkNibble
```

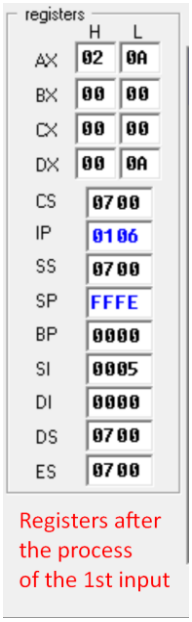
```
display:
;Print the location of the zero we found
MOV DX, 30h
ADD DX, SI

MOV CX, AX ;Backup AX

MOV AH, 2d
INT 21h

;Print the space character
MOV DX, 20h
MOV AH, 2d
INT 21h

MOV AX, CX ;Restore AX back
JMP continue

terminate:
;Feed new line
MOV DX, 0Dh
MOV AH, 2d
INT 21h

MOV DX, 0Ah
MOV AH, 2d
INT 21h

RET
findZero ENDP
```

Screenshots:

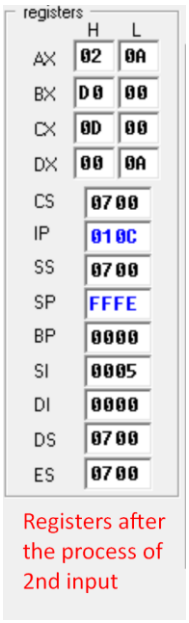
1st input-> 0000h, 2nd input-> D000h, 3rd input-> 0F0Ah, 4th input-> 10F7h, 5th input-> DDDDh,

6th input-> 2007h

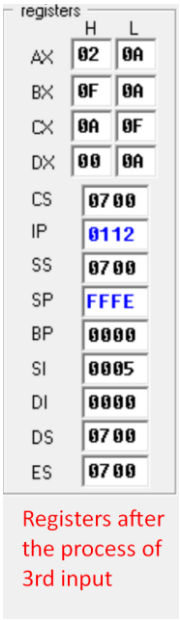
Registers:



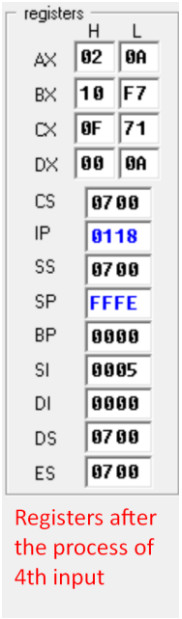
Registers after the process of the 1st input



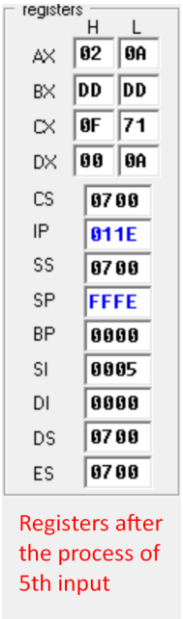
Registers after the process of 2nd input



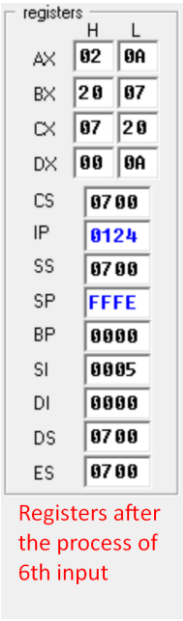
Registers after the process of 3rd input



Registers after the process of 4th input

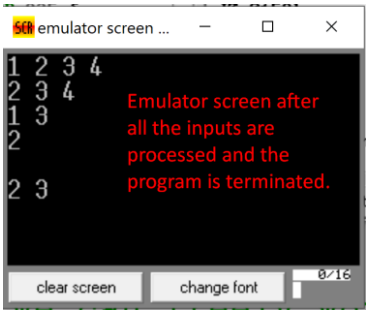


Registers after the process of 5th input



Registers after the process of 6th input

Emulator Screen:



Comments:

My algorithm lefts the most significant nibble alone by shifting the number to the right by using **SHR**, and then it checks whether the obtained number is equal to zero or not. If it is equal to zero, the code prints the value of the location counter **SI**. This process goes on until all the nibbles are checked. After each loop the number is rotated to the left by making use of **ROL** command, this way my code is able to check every nibble step by step.

"**checkNibble**" part of the code is the main loop of the algorithm. This rotations and shifts I've mentioned above take place in this loop.

"**display**" part of the code prints the locations of the zero nibbles on the emulator screen.

"**terminate**" part of the code just feeds a new line, and then terminates the procedure.

My code doesn't print anything if the number doesn't contain any zero nibbles, it just feeds a new line. I could do something else to specify this situation but I wanted to keep the code simple.

Q2)

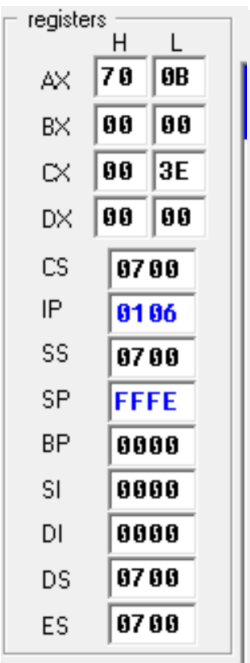
Source Code:

```
org 100h

;ELE338 - Preliminary Work 3 - Question 2

;Anil Karaca - 21728405

MOV AX,0B007h ;Expected result:AX=700Bh

CALL swap

MOV AX,0AABBh ;Expected result:AX=BABAh

CALL swap

MOV AX,0EEDDh ;Expected result:AX=DEDEh

CALL swap

MOV AX,07700h ;Expected result:AX=0707h

CALL swap

MOV AX,07901h ;Expected result:AX=1907h

CALL swap

MOV AX,00FB6h ;Expected result:AX=6FB0h

CALL swap


ret


swap PROC


ROL AH,4 ;Swap the nibbles in AH

ROR AX,4 ;Rotate the number to the right by a nibble

ROL AL,4 ;Swap the nibbles in AL


RET

swap ENDP
```

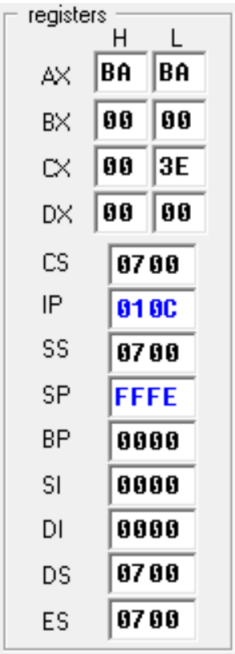
Screenshots:

1st input-> B007h, 2nd input-> AABBh, 3rd input-> EEDDh, 4th input-> 7700h, 5th input-> 7901h, 6th input-> 0FB6h

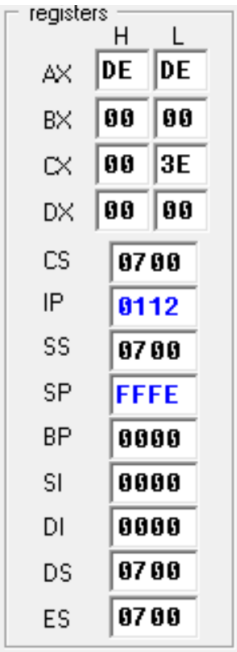
Registers:



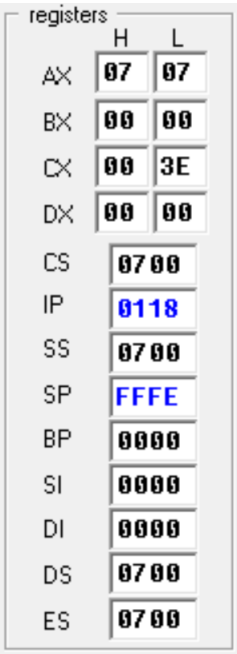
Registers after the 1st input is processed



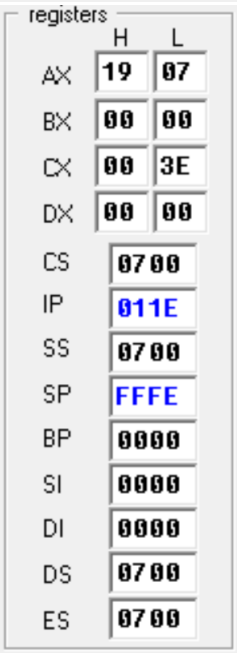
Registers after the 2nd input is processed



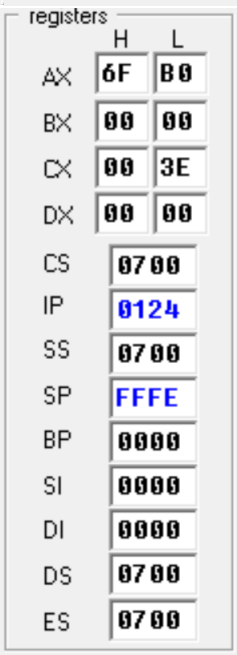
Registers after the 3rd input is processed



Registers after the 4th input is processed



Registers after the 5th input is processed



Registers after the 6th input is processed

Comments: I believe my algorithm for this question is pretty straight forward.

At the first, we swap AH's nibbles by making use of ROL. And then, we rotate the number stored in AX register by a nibble. And finally, we swap the AL's nibbles by making use of ROL once more.