UCS 1512: MICROPROCESSORS LAB PRACTICAL EXAMINATION

SRINITHYEE S K **185001166**

CSE C **18.11.20**

QUESTIONS:

7.a. Write an ALP using 8086 to find the largest among a list.
7.b. Write an ALP using 8051 to convert a number from HEX to BCD.

SOLUTIONS:

7a.

AIM:

To find the largest number among a list.

ALGORITHM:

1. Start.
2. Using the assume keyword, give names to denote the code and data segment.
3. Declare the data segment
4. Initialize the data segment with a list that stores several numbers, an increment counter variable and the result, which is the largest element.
5. Close the data segment.
6. Declare the code segment
7. Set the preferred offset value(usually 100h)
8. Load the data segment content into the AX register
9. Load the contents of AX register into the DS register
10. Set the pointer to the first element.
11. Store the length of the list in the cx register
12. Make the value of bl register to 0 to store the largest number
13. Move the element in SI to the AL register
14. Compare the value in al to the value in the address present in the BL register
15. If there is carry, copy the value of si to bl (ie, the new largest number)
16. If there is no carry:

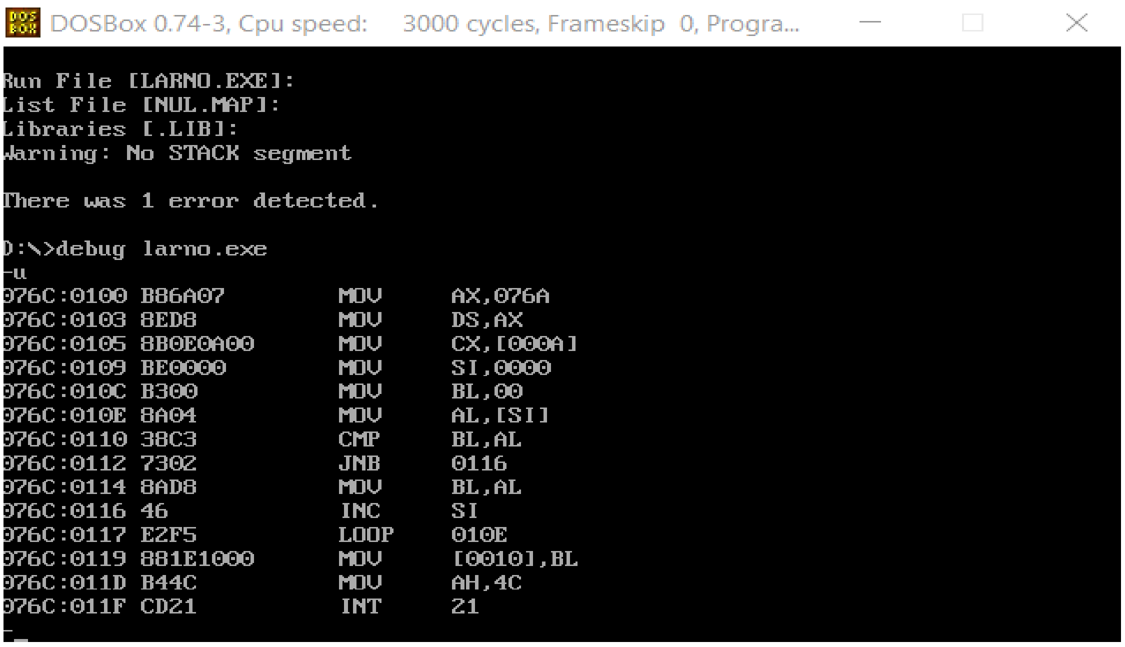
        i. Increment SI and move to the next element
        ii. Loop through step 14
17. Load the value in BL to the res variable
18. Introduce an interrupt for safe exit
19. Close the code segment
20. End.

CODE:

PROGRAM	COMMENTS
assume ds:data,cs:code	Name for code and data segment
data segment	Declare data segment
list db 11h,33h,22h,64h,56h,73h,31h,61h,43h,70h	Declare the list
count dw 000Ah	
org 0010h	
res db ?	Variable to store the largest number
data ends	Close data segment
code segment	Declare code segment
org 0100h	Set an offset value
start:	
mov ax,data	Copy the base address of data segment to AX
mov ds,ax	Copy the address in AX to DS
mov cx,count	Counter for length of list
mov si,offset list	Pointer to parse the list
mov bl,00h	Index of largest element
loop1:	
mov al,[si]	Move starting elemnt of si to AL
cmp bl,al	Compare the value in AL to the value presentin the address stored in BL
jnc here	Jump on not Carry to label “here”
mov bl,al	Copy the element to al
here:	
inc si	Move to next element in list

loop loop1	Loop to label loop1
mov res,bl	Move the value in BL(largest number) to res
mov ah,4ch	DOS interrupt for termination
int 21h	Interrupt the process with return code and exit
code ends	
end start	

UNASSEMBLED CODE:

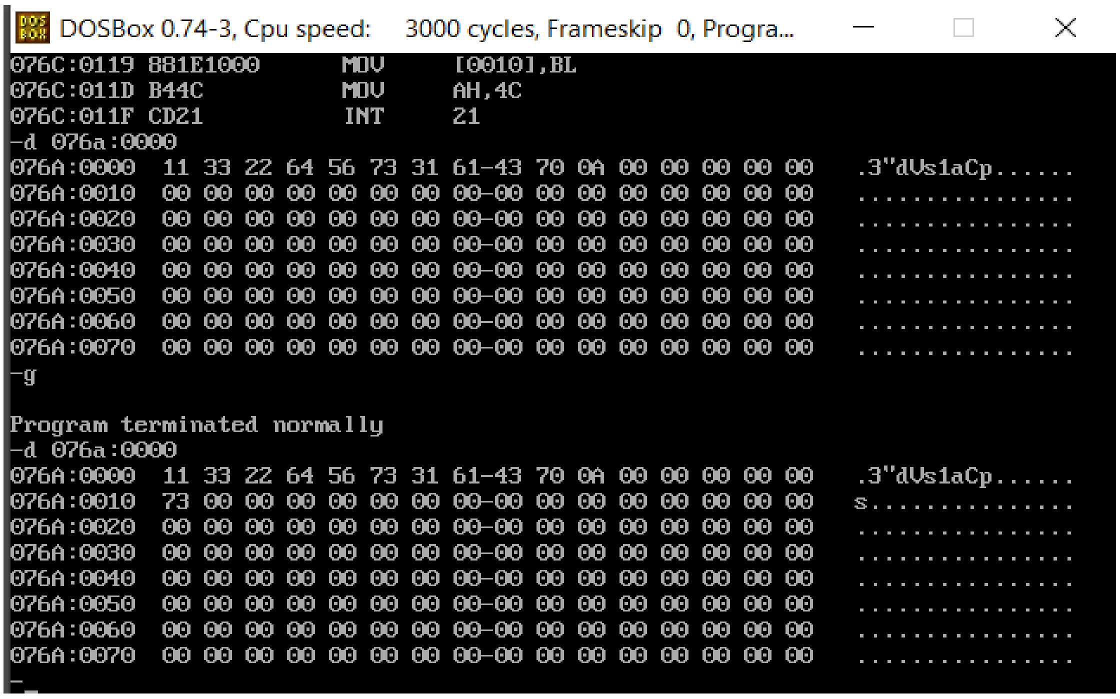

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
Run File [LARNO.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
Warning: No STACK segment

There was 1 error detected.

D:\>debug larno.exe
-u
076C:0100 B86A07        MOV     AX,076A
076C:0103 8ED8          MOV     DS,AX
076C:0105 8B0E0A00      MOV     CX,[000A]
076C:0109 BE0000        MOV     SI,0000
076C:010C B300          MOV     BL,00
076C:010E 8A04          MOV     AL,[SI]
076C:0110 38C3          CMP     BL,AL
076C:0112 7302          JNB     0116
076C:0114 8AD8          MOV     BL,AL
076C:0116 46            INC     SI
076C:0117 E2F5          LOOP    010E
076C:0119 881E1000      MOV     [0010],BL
076C:011D B44C          MOV     AH,4C
076C:011F CD21          INT     21
-_
```

SNAPSHOTS:


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
076C:0119 881E1000      MOV     [0010],BL
076C:011D B44C          MOV     AH,4C
076C:011F CD21          INT     21
-d 076a:0000
076A:0000  11 33 22 64 56 73 31 61-43 70 0A 00 00 00 00 00   .3"dVs1aCp......
076A:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-g

Program terminated normally
-d 076a:0000
076A:0000  11 33 22 64 56 73 31 61-43 70 0A 00 00 00 00 00   .3"dVs1aCp......
076A:0010  73 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   s...............
076A:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-_
```

RESULT:

An ALP was written in 8086 tofind the largest number from a list of numbers. It was tested and the results were verified.

7b.

AIM:

To convert a number from HEX to BCD.

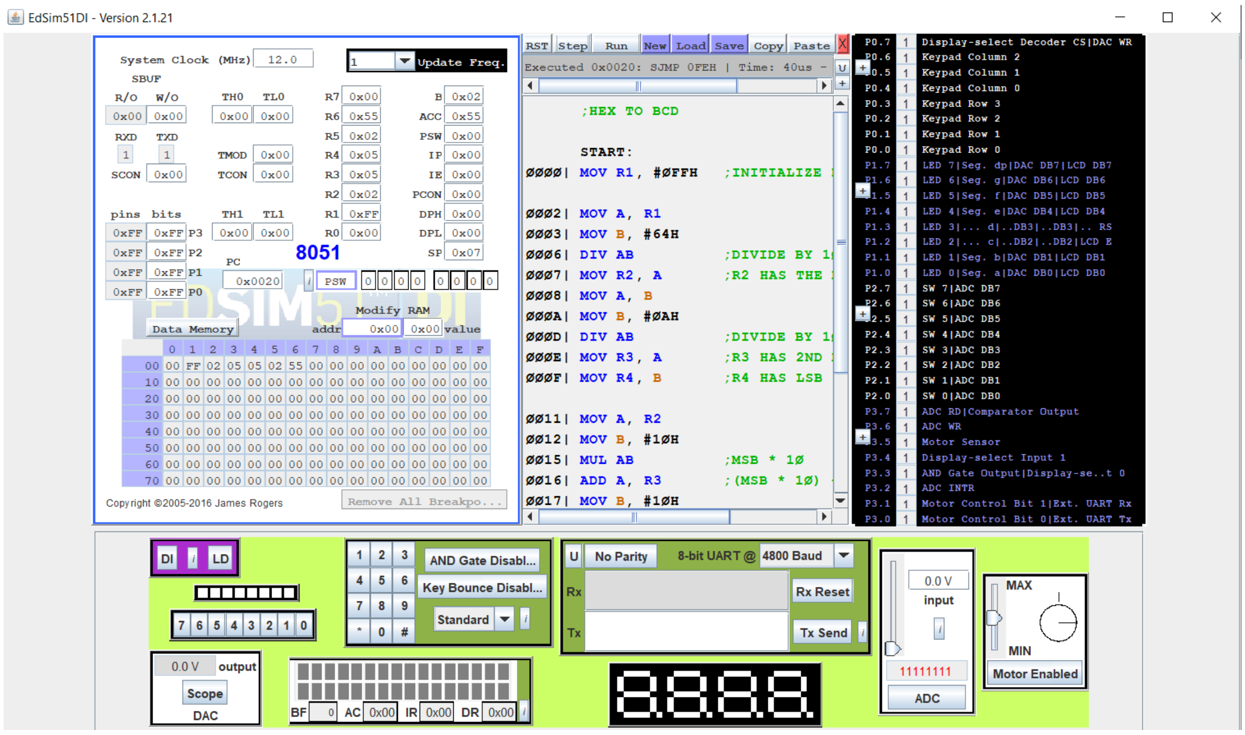
ALGORITHM:

1. Start.
2. Store the value of HEX form in register R1
3. Move the value of R1 to A
4. Load 64h into register B
5. Divide by 100
6. Move the most significant bytes to R2
7. Divide the most significant bits by 10
8. Move the 2nd Most Significant bit to R3
9. Move the least significant bit to r4
10. Multiply the Most Significant Bit with 10 (ie, the value stores in register R2 *10)
11. Add the resul of the above operation(ie, MSB*10) with 2nd MSB stored in R3
12. Multiply A abd B to get MSB*10 + 2nd MSB *10
13. The higher byte of the result is stored in register R5 and the lower byte is stored in register R6
14. The resultant is then added with the Lowest Significant bit, stored previously in the register R4
15. The final result is then moved to A
16. Halt the program
17. End.

CODE:

PROGRAM	COMMENTS
START:	
MOV R1, #0FFH	INITIALIZE HEX VALUE
MOV A, R1	
MOV B, #64H	
DIV AB	DIVIDE BY 100
MOV R2, A	R2 HAS THE MSB
MOV A, B	
MOV B, #0AH	
DIV AB	DIVIDE BY 10
MOV R3, A	R3 HAS 2ND MSB
MOV R4, B	R4 HAS LSB
MOV A, R2	
MOV B, #10H	
MUL AB	MSB * 10
ADD A, R3	(MSB * 10) + 2ND MSB
MOV B, #10H	
MUL AB	(MSB * 100 + 2ND MSB * 10)
MOV R5, B	HIGHER BYTE
MOV R6, A	LOWER BYTE
ADD A, R4	+ LSB
MOV R6, A	MOV RESULT TO A
HALT:	
SJMP HALT	

SNAPSHOT:



```
Executed 0x0020: SJMP 0FEH | Time: 40us -
```

```
          ;HEX TO BCD

          START:
ØØØØ| MOV R1, #ØFFH     ;INITIALIZE

ØØØ2| MOV A, R1
ØØØ3| MOV B, #64H
ØØØ6| DIV AB            ;DIVIDE BY 1
ØØØ7| MOV R2, A         ;R2 HAS THE
ØØØ8| MOV A, B
ØØØA| MOV B, #ØAH
ØØØD| DIV AB            ;DIVIDE BY 1
ØØØE| MOV R3, A         ;R3 HAS 2ND
ØØØF| MOV R4, B         ;R4 HAS LSB

ØØ11| MOV A, R2
ØØ12| MOV B, #1ØH
ØØ15| MUL AB            ;MSB * 1Ø
ØØ16| ADD A, R3         ;(MSB * 1Ø)
ØØ17| MOV B, #1ØH
```

```
System Clock (MHz) 12.0        1  Update Freq.
R7 0x00   B    0x02
R6 0x55   ACC  0x55
R5 0x02   PSW  0x00
R4 0x05   IP   0x00
R3 0x05   IE   0x00
R2 0x02   PCON 0x00
R1 0xFF   DPH  0x00
R0 0x00   DPL  0x00
          SP   0x07
PC 0x0020
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	FF	02	05	05	02	55	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2016 James Rogers

RESULT:

AN ALP was written in 8051 to convert a number from Hexadecimal to BCD. It was tested and the output was verified.