

**UCS 1512
MICROPROCESSORS LAB
END SEMESTER
PRACTICAL EXAM**

Batch: 9

Name: S Vishakan

Date: 18-11-2020

Reg. No: 18 5001 196

AIM:

To write assembly language programs to perform the following:

1. To write an ALP using 8086 to count the number of zeroes and ones in an 8-bit number.
2. To write an ALP using 8051 to find the largest number in a list.

PROGRAM – 1: 8086 ALP – COUNT NUMBER OF 0s AND 1s IN AN 8-BIT NUMBER:

ALGORITHM:

1. Begin.
2. Initialize the data segment.
3. Initialize a variable, say *“num”*, to store the 8-bit number.
4. Initialize variables, say *“zeroes”* and *“ones”* to store the number of zeroes and ones in the 8-bit number as output.
5. Close the data segment.
6. Initialize the code segment.
7. Load the 8-bit number *“num”* to the AL register.
8. Clear the contents of AH, BL, CH, DL registers.
9. Use BL register to keep track of number of zeroes, DL to keep track of number of ones.
10. Load the value 08H to CL register to keep count of number of digits.
11. While CL \neq 0:
 - a. Shift left the value in AL register by 1.
 - b. If CF \neq 1:
 - i. $BL \leftarrow BL + 1$.
 - c. Else:
 - i. $DL \leftarrow DL + 1$.
 - d. $CL \leftarrow CL - 1$.
12. Transfer the value in BL to *“zeroes”*, and the value in DL to *“ones”*.
13. Close the code segment.
14. End.

PROGRAM	COMMENTS
ASSUME CS: CODE, DS: DATA	
DATA SEGMENT	INITIALIZE DATA SEGMENT.
NUM DB 07H	<i>NUM</i> = 07 (8-BIT NUMBER)
ZEROES DB ?	VARIABLE TO STORE NUMBER OF ZEROES.
ONES DB ?	VARIABLE TO STORE NUMBER OF ONES.
DATA ENDS	
CODE SEGMENT	
ORG 0100H	
START:	
MOV AX, DATA	
MOV DS, AX	DS POINTS TO BASE ADDRESS OF DATA SEGMENT.
MOV AH, 00H	CLEAR AH.
MOV CH, 00H	CLEAR CH.
MOV AL, NUM	$AL \leftarrow NUM$.
MOV BL, 00H	TO COUNT THE NUMBER OF ZEROES.
MOV CL, 08H	COUNT OF NUMBER OF DIGITS.
MOV DL, 00H	TO COUNT THE NUMBER OF ONES.
LOOPER:	
SHL AL, 1	SHIFT LEFT BY 1, IF CARRY = 1, THEN THE LSB DIGIT WAS 1.
JC ONE	GO TO <i>ONE</i> .
INC BL	ELSE, INCREMENT ZEROES COUNT, I.E BL.
JMP SKIP	GO TO <i>SKIP</i> .
ONE:	
INC DL	$DL = DL + 1$.
SKIP:	
DEC CL	$CL = CL + 1$.
JNZ LOOPER	IF $CL \neq 0$, GO BACK TO <i>LOOPER</i> .
MOV ZEROES, BL	$ZEROES \leftarrow BL$.
MOV ONES, DL	$ONES \leftarrow DL$.
MOV AH, 4CH	TERMINATE THE PROGRAM WITH DOS INTERRUPT.
INT 21H	
CODE ENDS	
END START	

UNASSEMBLED CODE:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Microsoft Object Linker V2.01 (Large)
(C) Copyright 1982, 1983 by Microsoft Inc.

Warning: No STACK segment

There was 1 error detected.

Q:\>DEBUG COUNT01.EXE
-U
076B:0100 B86A07      MOV     AX,076A
076B:0103 8ED8        MOV     DS,AX
076B:0105 B400        MOV     AH,00
076B:0107 B500        MOV     CH,00
076B:0109 A00000      MOV     AL,[0000]
076B:010C B300        MOV     BL,00
076B:010E B108        MOV     CL,08
076B:0110 B200        MOV     DL,00
076B:0112 D0E0        SHL     AL,1
076B:0114 7204        JB      011A
076B:0116 FEC3        INC     BL
076B:0118 EB02        JMP     011C
076B:011A FEC2        INC     DL
076B:011C FEC9        DEC     CL
076B:011E 75F2        JNZ     0112
-
```

SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
076B:011A FEC2      INC      DL
076B:011C FEC9      DEC      CL
076B:011E 75F2      JNZ      0112
-D 076A:0000
076A:0000 07 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
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 07 05 03 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
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 .....
-
```

PROGRAM – 2: 8051 ALP TO FIND THE LARGEST NUMBER IN A GIVEN LIST:

ALGORITHM:

1. Begin.
2. Initialize the list in the internal memory with some values starting from a base address in the internal RAM, say 20H.
3. Store the number of elements in a register, say R1.
4. Store the base address of the list in a register, say R0.
5. Clear the accumulator register.
6. While $R1 \neq 0$:
 - a. Transfer to B the value pointed by the address stored at R0.
 - b. Compare the values of A and B.
 - c. If $A < B$:
 - i. $A \leftarrow B$.
 - d. Else:
 - i. $R0 \leftarrow R0 + 1$.
 - e. $R1 \leftarrow R1 - 1$.
7. Transfer the value stored in register A to a location in the internal RAM, say 30H.
8. End.

PROGRAM	COMMENTS
START:	
MOV R0, #00H	CLEAR THE VALUE OF REGISTER R0.
MOV 30H, #00H	CLEAR THE VALUE AT ADDRESS 30H.
MOV 20H, #01H	LOAD THE LIST DATA INTO MEMORY.
MOV 21H, #02H	
MOV 22H, #05H	
MOV 23H, #04H	
MOV 24H, #03H	
MOV R1, #05H	LOAD THE LENGTH OF THE LIST INTO R1.
MOV R0, #20H	LOAD THE BASE ADDRESS INTO R0.
CLR A	CLEAR ACCUMULATOR REGISTER.
BACK:	
MOV B, @R0	MOVE THE VALUE POINTED BY THE ADDRESS AT R0 TO B.
CJNE A, B, NEXT	IF A \neq B, JUMP TO NEXT.
NEXT:	
JNC SKIP	IF A < B, THEN CARRY FLAG = 1. ELSE CARRY FLAG = 0.
MOV A, B	A \leftarrow B. A IS THE CURRENT LARGEST VALUE.
SKIP:	
INC R0	GOTO NEXT ELEMENT BY INCREMENTING R0 BY 1.
DJNZ R1, BACK	IF R1 \neq 0, GO TO "BACK".
MOV 30H, A	MOVE LARGEST ELEMENT, I.E VALUE IN A, TO 30H.
HALT:	
SJMP HALT	HALT THE PROGRAM WITH AN INFINITE LOOP.

SAMPLE I/O SNAPSHOT:

EdSim51DI - Version 2.1.21

System Clock (MHz) 12.0 1 Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	0x00	B	0x04
0x00	0x00	0x00	0x00	R6	0x00	ACC	0x05
RxD	TxD	TMOD	0x00	R5	0x00	PSW	0x00
1	1	TCON	0x00	R4	0x00	IP	0x00
SCON	0x00	PC	0x0027	R3	0x00	IE	0x00
				R2	0x00	PCON	0x00
pins	bits	TH1	TL1	R1	0x00	DPH	0x00
0xFF	0xFF	P3	0x00	R0	0x25	DPL	0x00
0xFF	0xFF	P2				SP	0x07
0xFF	0xFF	P1					
0xFF	0xFF	P0					

8051

Data Memory

addr	0x00	0x00	value
0	25	00	00
1	00	00	00
2	00	00	00
3	00	00	00
4	00	00	00
5	00	00	00
6	00	00	00
7	00	00	00
8	00	00	00
9	00	00	00
A	00	00	00
B	00	00	00
C	00	00	00
D	00	00	00
E	00	00	00
F	00	00	00

Modify RAM

Remove All Breakpo...

RST Step Pause New Load Save Copy Paste

Time: 117us - Instructions: 65

```

;8051 ALP TO FIND THE LARGES

START:
0000| MOV R0, #00H
0002| MOV 30H, #00H

;LOAD DATA INTO MEMORY
0005| MOV 20H, #01H
0008| MOV 21H, #02H
000B| MOV 22H, #05H
000E| MOV 23H, #03H
0011| MOV 24H, #04H

;LOAD THE LENGTH
0014| MOV R1, #05H

;FIND THE LARGEST NUMBER
0016| MOV R0, #20H ;LOAD THE BA
0018| CLR A ;CLEAR AC

```

P0.7 1 Display-select Decoder CS|DAC WR
P0.6 1 Keypad Column 2
P0.5 1 Keypad Column 1
P0.4 1 Keypad Column 0
P0.3 1 Keypad Row 3
P0.2 1 Keypad Row 2
P0.1 1 Keypad Row 1
P0.0 1 Keypad Row 0
P1.7 1 LED 7|Seg. dp|DAC DB7|LCD DB7
P1.6 1 LED 6|Seg. g|DAC DB6|LCD DB6
P1.5 1 LED 5|Seg. f|DAC DB5|LCD DB5
P1.4 1 LED 4|Seg. e|DAC DB4|LCD DB4
P1.3 1 LED 3|... d|..DB3|..DB3|.. RS
P1.2 1 LED 2|... c|..DB2|..DB2|LCD E
P1.1 1 LED 1|Seg. b|DAC DB1|LCD DB1
P1.0 1 LED 0|Seg. a|DAC DB0|LCD DB0
P2.7 1 SW 7|ADC DB7
P2.6 1 SW 6|ADC DB6
P2.5 1 SW 5|ADC DB5
P2.4 1 SW 4|ADC DB4
P2.3 1 SW 3|ADC DB3
P2.2 1 SW 2|ADC DB2
P2.1 1 SW 1|ADC DB1
P2.0 1 SW 0|ADC DB0
P3.7 1 ADC RD|Comparator Output
P3.6 1 ADC WR
P3.5 1 Motor Sensor
P3.4 1 Display-select Input 1
P3.3 1 AND Gate Output|Display-se..t 0
P3.2 1 ADC INTR
P3.1 1 Motor Control Bit 1|Ext. UART Rx
P3.0 1 Motor Control Bit 0|Ext. UART Tx

DI LD

AND Gate Disabl...

Key Bounce Disabl...

Standard

U No Parity 8-bit UART @ 4800 Baud

Rx Rx Reset

Tx Tx Send

0.0 V output

Scope DAC

BF 0 AC 0x00 IR 0x00 DR 0x00

1111111

ADC

MAX MIN

Motor Enabled

RESULT:

The assembly level programs were written to perform the above specified tasks (counting the number of 0s and 1s in an 8-bit number and finding the largest number in a list using 8086 and 8051 respectively) and their outputs were verified.