

MODEL PRACTICAL EXAM

Batch: 9

Name: S Vishakan

Date: 10-11-2020

Reg. No: 18 5001 196

AIM:

To write assembly language programs to perform the following:

1. To write an ALP using 8051 to sort a list of numbers in descending order.
2. To write an ALP using 8086 to count odd and even numbers in a list.

PROGRAM – 1: 8051 ALP – DESCENDING ORDER SORT OF A 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 no. of elements (count of the list) in a register, say R7.
4. Copy the count value to another register, say R6. R7 & R6 now denote the outer and inner loop count, respectively.
5. Store the base address of the list (20H) in a register, say R0.
6. While R7 \neq 0:
 - a. While R6 \neq 0:
 - i. $A \leftarrow [R0]$.
 - ii. $B \leftarrow [R0 + 1]$.
 - iii. If $A < B$:
 1. $[R0 + 1] \leftarrow A$.
 2. $[R0] \leftarrow B$.
 - iv. Decrement R6.
 - b. Decrement R7.
7. Find the descending order sorted list in the internal RAM, starting from the base address (20H) till (20H + no. of elements in the list)
8. End.

PROGRAM	COMMENTS
ORG 0	
MOV 20H, #01H	PUT 5 NUMBERS IN THE INTERNAL MEMORY.
MOV 21H, #02H	STARTING FROM 20H.
MOV 22H, #03H	ASSUME 5 NUMBERS IN THE SERIES.
MOV 23H, #04H	
MOV 24H, #05H	
START:	OUTER LOOP.
MOV R7, #04H	OUTER LOOP COUNT.
BACK2:	INNER LOOP.
MOV R6, #04H	INNER LOOP COUNT.
MOV R0, #20H	POINT TO BASE ADDRESS.
BACK1:	
MOV A, @R0	A HAS THE FIRST NUMBER.
INC R0	GO TO THE NEXT LOCATION.
MOV B, @R0	B HAS THE SECOND NUMBER.
CJNE A, B, NEXT	IF A \neq B, THEN GO TO NEXT NUMBER.
NEXT:	
JNC SKIP	IF A > B, GO TO SKIP. ELSE SWAP A AND B.
MOV @R0, A	CURRENT LOCATION, POINTED BY R0 HAS A NOW.
DEC R0	GO TO PREVIOUS LOCATION.
MOV @R0, B	PREVIOUS LOCATION, POINTED BY R0 HAS B NOW.
INC R0	GO TO THE NEXT LOCATION.
SKIP:	
DJNZ R6, BACK1	DECREMENT R6. IF R6 \neq 0, GO TO BACK1.
DJNZ R7, BACK2	DECREMENT R7. IF R7 \neq 0, GO TO BACK2.
HALT:	
SJMP HALT	HALT THE PROGRAM USING AN INFINITE LOOP.

SAMPLE I/O SNAPSHOT:

EdSim51DI - Version 2.1.21 & Dynamic Interface x

System Clock (MHz) 12.0 1 Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	B
0x00	0x00	0x00	0x00	0x00	0x01

RXD TXD

R7	ACC	PSW	IP	IE	PCON	DPH	DPL	SP
0x00	0x02	0x01	0x00	0x00	0x00	0x00	0x00	0x07

SCON 0x00 TCON 0x00

pins bits

TH1	TL1	PC	PSW
0x00	0x00	0x0027	0 0 0 0 0 0 0 1

Modify RAM

addr	0x00	0x00	value
0	24	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

Copyright ©2005-2016 James Rogers Remove All Breakpo...

RST Step Run New Load Save Copy Paste

Time: 275us - Instructions: 173

```
BACK1:
0015| MOV A, @R0 ;A HAS 1
0016| INC R0
0017| MOV B, @R0 ;B HAS 2
0019| CJNE A, B, NEXT ;IF A !=

NEXT:
001C| JNC SKIP ;IF A>B,
001E| MOV @R0, A ;2ND POSI
001F| DEC R0
0020| MOV @R0, B ;1ST POSI
0022| INC R0 ;R0 POINT

SKIP:
0023| DJNZ R6, BACK1 ;DECREMEN
0025| DJNZ R7, BACK2 ;DECREMEN

HALT:
0027| SJMP HALT
```

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

1 2 3 AND Gate Disabl...
4 5 6 Key Bounce Disabl...
7 8 9 Standard
* 0 #

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

8.8.8.8

0.0 V input
1111111
ADC

MAX
MIN
Motor Enabled

PROGRAM – 2: 8086 ALP – COUNT ODD AND EVEN NUMBERS IN A LIST:

ALGORITHM:

1. Begin.
2. Initialize the data segment.
3. Initialize an array(list) *arr* with some odd and even values.
4. Initialize a variable *arr_size* with the array length.
5. Initialize variables to store the counts of odd numbers & even numbers in the list, i.e. *oddcnt* & *evencnt*.
6. Close the data segment.
7. Start the code segment.
8. Move the starting address of data segment to DS using AX register.
9. Set $CL \leftarrow arr_size$.
10. Set $SI \leftarrow$ base address of *arr*.
11. Clear DX register.
12. While $CL \neq 0$:
 - a. $AL \leftarrow [SI]$.
 - b. $AL \leftarrow AL \& 01H$.
 - c. If $AL = 0$:
 - i. $DH = DH + 1$. (Stores even numbers count)
 - d. Else:
 - i. $DL = DL + 1$. (Stores odd number count)
 - e. $SI \leftarrow SI + 1$.
 - f. $CL \leftarrow CL - 1$.
13. $oddcnt \leftarrow DL$.
14. $evencnt \leftarrow DH$.
15. Terminate the program with DOS interrupt 4CH.
16. Close the code segment.
17. End.

PROGRAM	COMMENTS
ASSUME CS: CODE, DS: DATA	
DATA SEGMENT	INITIALIZE DATA SEGMENT.
ARR DB 02H, 08H, 0BH, 05H, 0BH, 0DH	ARRAY WITH 6 VALUES.
ORG 0010H	
ARR_SIZE DB 06H	<i>ARR_SIZE</i> = LENGTH OF ARRAY <i>ARR</i> .
ODDCNT DB 00H	VARIABLE TO STORE ODD NUMBER COUNT.
EVENCNT DB 00H	VARIABLE TO STORE EVEN NUMBER COUNT.
DATA ENDS	
CODE SEGMENT	
ORG 0100H	
START:	
MOV AX, DATA	
MOV DS, AX	DS POINTS TO BASE ADDRESS OF DATA SEGMENT.
MOV CL, ARR_SIZE	$CL \leftarrow ARR_SIZE$.
MOV SI, OFFSET ARR	SI HAS THE BASE ADDRESS OF LIST <i>ARR</i> .
MOV DX, 0000H	CLEAR DX. DX WILL STORE THE COUNTS.
LOOP1:	
MOV AL, [SI]	$AL \leftarrow [SI]$. (AL GETS THE VALUE AT LOCATION POINTED BY SI)
AND AL, 01H	($AL \& 0000\ 0001$) TO GET LAST BIT OF AL.
CMP AL, 00H	IF $AL = 0 \Rightarrow$ EVEN NUMBER.
JNZ ODD	OTHERWISE, JUMP TO LABEL <i>ODD</i> .
INC DH	$DH = DH + 1$, FOR EVEN COUNT.
JMP SKIP	JUMP TO LABEL <i>SKIP</i> .
ODD:	
INC DL	$DL = DL + 1$, FOR ODD COUNT.
SKIP:	
INC SI	$SI = SI + 1$, TO POINT TO THE NEXT NUMBER IN LIST
DEC CL	DECREMENT CL.
CMP CL, 00H	CHECK IF $CL = 0$.
JNZ LOOP1	IF $CL \neq 0$, GO BACK TO <i>LOOP1</i> .
HERE:	
MOV ODDCNT, DL	<i>ODDCNT</i> \leftarrow DL.
MOV EVENCNT, DH	<i>EVENCNT</i> \leftarrow DH.
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 COUNT.EXE
-u
076C:0100 B86A07      MOV     AX,076A
076C:0103 8ED8        MOV     DS,AX
076C:0105 8A0E1000     MOV     CL,[0010]
076C:0109 BE0000      MOV     SI,0000
076C:010C BA0000      MOV     DX,0000
076C:010F 8A04        MOV     AL,[SI]
076C:0111 2401        AND     AL,01
076C:0113 3C00        CMP     AL,00
076C:0115 7504        JNZ     011B
076C:0117 FEC6        INC     DH
076C:0119 EB02        JMP     011D
076C:011B FEC2        INC     DL
076C:011D 46          INC     SI
076C:011E FEC9        DEC     CL
-
```

SAMPLE I/O SNAPSHOT:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
076C:011B FEC2      INC      DL
076C:011D 46        INC      SI
076C:011E FEC9      DEC      CL
-d 076A:0000
076A:0000 02 08 0B 05 0B 0D 00 00-00 00 00 00 00 00 00 00 .....
076A:0010 06 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 02 08 0B 05 0B 0D 00 00-00 00 00 00 00 00 00 00 .....
076A:0010 06 04 02 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 .....
-
```


RESULT:

The assembly level programs were written to perform the above specified tasks (descending order sort using 8051 & odd-even numbers count in a list using 8086 respectively), and their outputs were verified.