Experiment No. 7

Title: 8086 Assembly program to sort numbers using bubble sort algorithm

Batch: B1 **Roll No: 16010420133** **Experiment No.: 7**

Aim: Use 8086 Assembly program to sort numbers using bubble sort algorithm.

Resources needed: DosBox, 8086 Assembler-TASM

Theory:

Bubble sort is the simplest sorting algorithm available. To write this program in 8086 assembly language we store the data (assume 8 bit data) in the data segment and then write the program in the code segment.

Stack segment is not required for this program.

Procedure:

a) Download the given program.
b) Compile and link the program in DOXBOX using the given instructions.
c) Step through the program using F8 key.
d) Take screenshot of the output once the program is over and include it in the writeup.

Observations and Results:

1) Understand basics of assembly language programming for microprocessor 8086.

2) Observe the memory locations and different registers affected while execution of the program.

Algorithm:

1. Load data from offset 500 to register CL (for count).
2. Travel from starting memory location to last and compare two numbers if first number is greater than second number then swap them.
3. First pass fix the position for last number.
4. Decrease the count by 1.
5. Again travel from starting memory location to (last-1, by help of count) and compare two numbers if first number is greater than second number then swap them.
6. Second pass fix the position for last two numbers.

7. Repeated.

Program:

MEMORY ADDRESS	MNEMONICS	COMMENT
400	MOV SI, 500	SI<-500
403	MOV CL, [SI]	CL<-[SI]
405	DEC CL	CL<-CL-1
407	MOV SI, 500	SI<-500
40A	MOV CH, [SI]	CH<-[SI]
40C	DEC CH	CH<-CH-1
40E	INC SI	SI<-SI+1
40F	MOV AL, [SI]	AL<-[SI]
411	INC SI	SI<-SI+1
412	CMP AL, [SI]	AL-[SI]
414	JC 41C	JUMP TO 41C IF CY=1
416	XCHG AL, [SI]	SWAP AL AND [SI]
418	DEC SI	SI<-SI-1
419	XCHG AL, [SI]	SWAP AL AND [SI]
41B	INC SI	SI<-SI+1
41C	DEC CH	CH<-CH-1
41E	JNZ 40F	JUMP TO 40F IF ZF=0
420	DEC CL	CL<-CL-1
422	JNZ 407	JUMP TO 407 IF ZF=0
424	HLT	END

Example Explanation –

1. **MOV SI, 500**: set the value of SI to 500.
2. **MOV CL, [SI]**: load data from offset SI to register CL.
3. **DEC CL**: decrease value of register CL BY 1.
4. **MOV SI, 500**: set the value of SI to 500.
5. **MOV CH, [SI]**: load data from offset SI to register CH.
6. **DEC CH**: decrease value of register CH BY 1.
7. **INC SI**: increase value of SI BY 1.
8. **MOV AL, [SI]**: load value from offset SI to register AL.
9. **INC SI**: increase value of SI BY 1.
10. **CMP AL, [SI]**: compares value of register AL and [SI] (AL-[SI]).
11. **JC 41C**: jump to address 41C if carry generated.
12. **XCHG AL, [SI]**: exchange the contents of register AL and SI.

13. **DEC SI**: decrease value of SI by 1.
14. **XCHG AL, [SI]**: exchange the contents of register AL and SI.
15. **INC SI**: increase value of SI by 1.
16. **DEC CH**: decrease value of register CH by 1.
17. **JNZ 40F**: jump to address 40F if zero flat reset.
18. **DEC CL**: decrease value of register CL by 1.
19. **JNZ 407**: jump to address 407 if zero flat reset.
20. **HLT**: stop.

Output:

Input Data	04	F9	F2	39	05
Memory Address(offset)	500	501	502	503	504

Output Data	05	39	F2	F9
Memory Address(offset)	501	502	503	504

Example explanation:

Pass-1:
F9 F2 39 05
F2 F9 39 05
F2 39 F9 05
F2 39 05 F9 (1 number got fix)

Pass-2:
F2 39 05 F9
39 F2 05 F9
39 05 F2 F9 (2 number got fix)

Pass-3:
39 05 F2 F9
05 39 F2 F9 (sorted)

Answer the following Questions:

a) Which registers have you used to access the array?

ANS: Base registers are used to access the array. An array element can be accessed through an index number. Array elements are accessed by base register using an integer index. Array index starts with 0 and goes till size of array minus 1. Name of the array is also a pointer to the first element of array.

(b) What function is used to display the result on the screen?

ANS:An output function is a function that an optimization function calls at each iteration of its algorithm. Typically, you use an output function to generate graphical output, record the history of the data the algorithm generates, or halt the algorithm based on the data at the current iteration.

Outcome:

CO5- Understand the fundamental concepts of microprocessors.

Conclusion:

Successfully implemented 8086 Assembly program to sort numbers using bubble sort algorithm.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Douglas Hall, Microprocessors and interfacing, McGraw Hill

Successfully implemented 8086 Assembly program to sort numbers using bubble sort algorithm.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:
Books/ Journals/ Websites:

1. Douglas Hall, Microprocessors and interfacing, McGraw Hill
2. https://www.geeksforgeeks.org/8086-program-sort-integer-array-ascending/