Microprocessor Lab – **Warren Fernandes 8940** Academic year: 2020-2021

EXPERIMENT 4

OBJECTIVE: Program involves sorting an array in ascending order using Bubble sort algorithm. The objective of this program is to give an overview of Compare and Jump instructions. Use of Indirect Addressing mode for array addressing is expected.

Aim: ARRANGING NUMBER IN ASCENDING ORDER.

Theory: 1) compare instruction:

CMP destination, source

This instruction compares the source with destination. The source and destination must be of same size. Comparison is done by internally subtracting source from destination. The result of this is not stored anywhere instead flag bits are affected.

2) JMP instruction:

if condition is true then, it is similar to an intra-segment branch and if false then branch does not take place and next sequential instruction is executed. The destination must be in range of -128 to 127 from address of instruction.

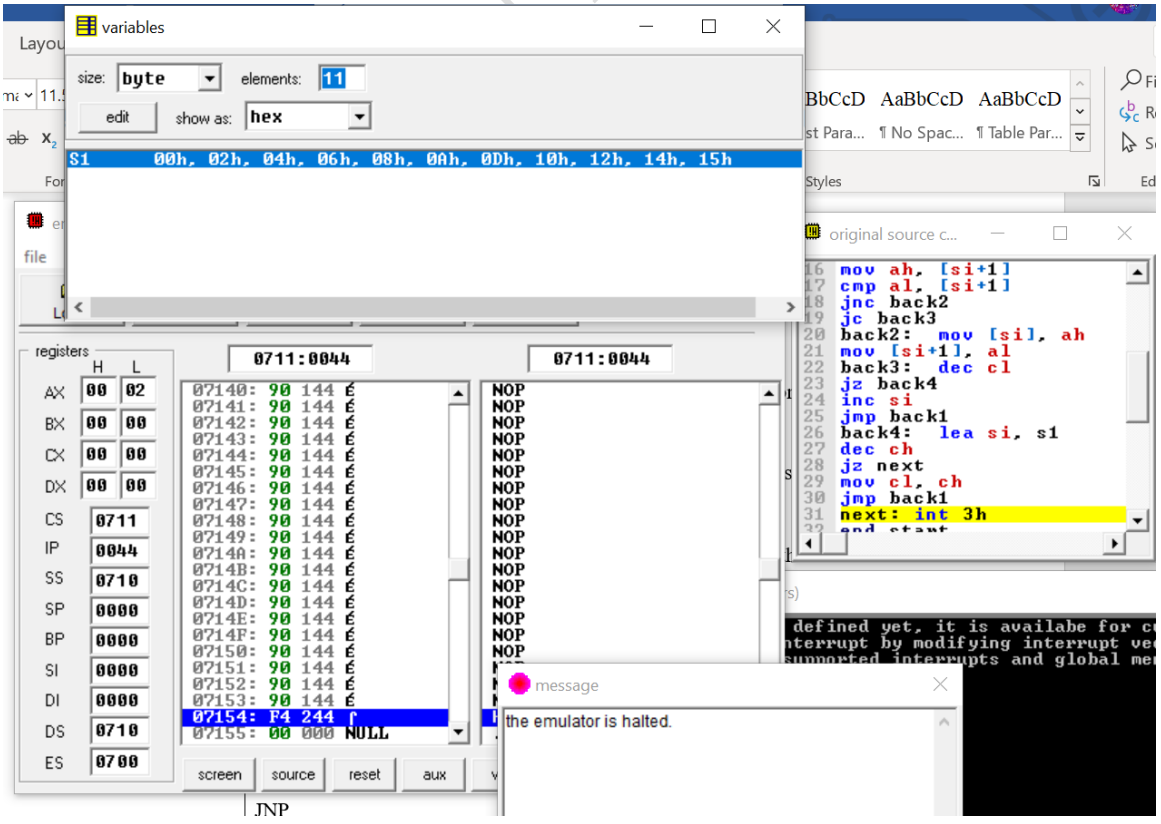
Algorithm:

1. Initialize the data segment.
2. Initialize the array to be sorted.
3. Store number of iterations required in one register.
4. Store number of comparisons required in another register.
5. Load the effective address of array in index register, SI.
6. Load the first element of the array (contents of SI) in a register AL.
7. Compare AL with the next element of an array (content of SI+1).
8. Check for carry flag.
9. If carry=0 first number > second number. Swap the 2 numbers using temporary register.
10. Increment SI register to points to the next element of an array.
11. Decrement comparison counter by 1.
12. Check if it is 0. If no then repeat steps 7 to 11.
13. Decrement counter for iteration by 1.
14. Check if it is 0.If no then repeat steps 6 through 13.
15. Stop.

CODE:

```
.8086
.model small
.data
s1 db 10h, 12h, 08h, 02h, 04h, 0Ah, 0Dh, 06h, 14h, 15h
.code
start:
mov ax, @data
mov ds, ax
mov cl, 0Ah
mov ch, 0Ah
lea si, s1
back1:  mov al, [si]
        mov ah, [si+1]
        cmp al, [si+1]
        jnc back2
        jc back3
back2:  mov [si], ah
        mov [si+1], al
back3:  dec cl
        jz back4
        inc si
        jmp back1
back4:  lea si, s1
        dec ch
        jz next
        mov cl, ch
        jmp back1
next: int 3h
end start
```

SCREENSHOTS:



Post Lab:

1) Discuss Control transfer instruction in detail.

   Ans.

   - JMP – Unconditional
     1. Direct Jump

        The new branch location is specified directly in the instruction. The new address is calculated by adding the 8 or 16 bit displacement to the IP
     2. Indirect Jump

        The new branch address is specified indirectly through a register or a memory location. The value in the IP is replaced with the new value
   - JCondition

     This is conditional branch instruction. If the condition is TRUE then it is similar to direct jump and if FALSe then branch does not take place and next sequential instruction is executed.

Mnemonic	**Description**
JC	Carry
JNC	Not Carry
JZ	Equal or Zero
JNZ	Not Equal or Not Zeor
JP	Parity or Parity Even
JNP	Not Parity or Paruty Odd

   - CALL
     1. Near Call

        The new subroutine called must be in the same segment. The call address can be specified directly in the instruction or indirectly through registers or memory locations.
     2. Far Call

        The new subroutine called is in another segment. Here CS and IP both get new values.
   - RET

     RET – Return instruction causes the control to return to the main program from the subroutine

2) What is the difference between near jump and far jump?

   Ans.

   NEAR – The procedure targets within the same code segment

   FAR – In this procedure the target is outside the code segment and the size of the pointer is double word