|  |  |  |
| --- | --- | --- |
|  | 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.

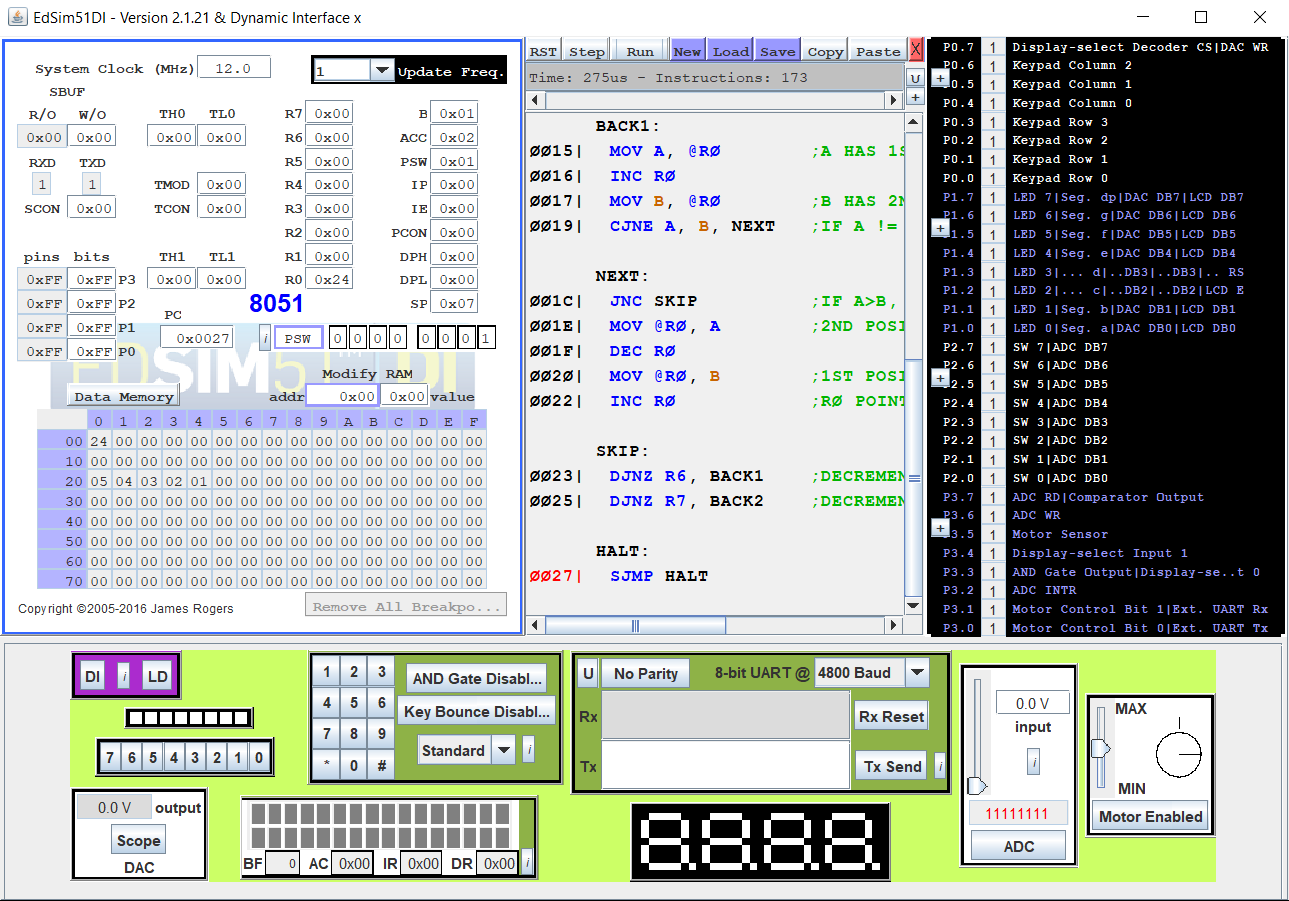
**PROGRAM – 1: 8051 ALP – DESCENDING ORDER SORT OF A LIST:**

**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 ≠ 0:
   1. While R6 ≠ 0:
      1. A ← [R0].
      2. B ← [R0 + 1].
      3. If A < B:
         1. [R0 + 1] ← A.
         2. [R0] ← B.
      4. Decrement R6.
   2. 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 ≠ 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 ≠ 0, GO TO BACK1. |
| DJNZ R7, BACK2 | DECREMENT R7. IF R7 ≠ 0, GO TO BACK2. |
|  |  |
| HALT: |  |
| SJMP HALT | HALT THE PROGRAM USING AN INFINITE LOOP. |

**SAMPLE I/O SNAPSHOT:**

****

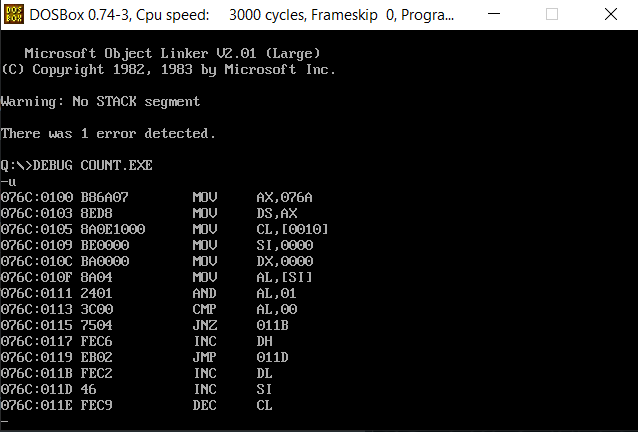
**PROGRAM – 2: 8086 ALP – COUNT ODD AND EVEN NUMBERS IN A LIST:**

**ALGORITHM:**

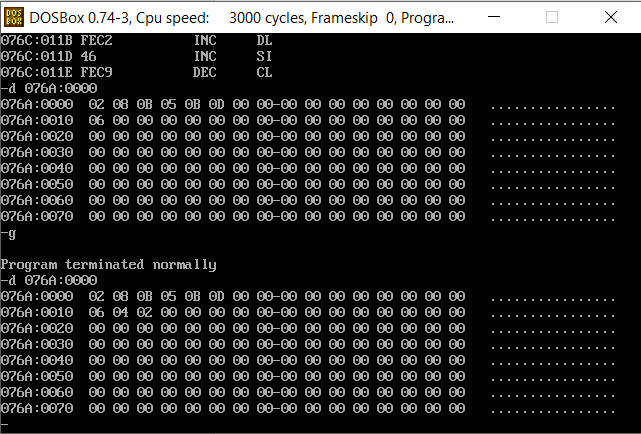
1. Begin.
2. Initialize the data segment.
3. Initialize an array(list) *arr* with some odd and even values.
4. Initialize a variable *arr\_size* with the array length.
5. Initialize variables to store the counts of odd numbers & even numbers in the list, i.e. *oddcnt* & *evencnt*.
6. Close the data segment.
7. Start the code segment.
8. Move the starting address of data segment to DS using AX register.
9. Set CL ← *arr\_size*.
10. Set SI ← base address of *arr.*
11. Clear DX register.
12. While CL ≠ 0:
    1. AL ← [SI].
    2. AL ← AL & 01H.
    3. If AL = 0:
       1. DH = DH + 1. (Stores even numbers count)
    4. Else:
       1. DL = DL + 1. (Stores odd number count)
    5. SI ← SI + 1.
    6. CL ← CL – 1.
13. *oddcnt* ← DL.
14. *evencnt* ← 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 | *ARR\_SIZE* = LENGTH OF ARRAY *ARR.* |
| 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 ← *ARR\_SIZE*. |
| MOV SI, OFFSET ARR | SI HAS THE BASE ADDRESS OF LIST *ARR.* |
| MOV DX, 0000H | CLEAR DX. DX WILL STORE THE COUNTS. |
|  |  |
| LOOP1: |  |
| MOV AL, [SI] | AL ← [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 ⟹ EVEN NUMBER. |
| JNZ ODD | OTHERWISE, JUMP TO LABEL *ODD.* |
| INC DH | DH = DH + 1, FOR EVEN COUNT. |
| JMP SKIP | JUMP TO LABEL *SKIP.* |
|  |  |
| 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 ≠ 0, GO BACK TO *LOOP1.* |
|  |  |
| HERE: |  |
| MOV ODDCNT, DL | *ODDCNT* ← DL. |
| MOV EVENCNT, DH | *EVENCNT* ← DH. |
|  |  |
| MOV AH,4CH | TERMINATE THE PROGRAM WITH DOS INTERRUPT. |
| INT 21H |  |
|  |  |
| CODE ENDS |  |
| END START |  |

**UNASSEMBLED CODE:**

****

**SAMPLE I/O SNAPSHOT:**

****

**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.