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

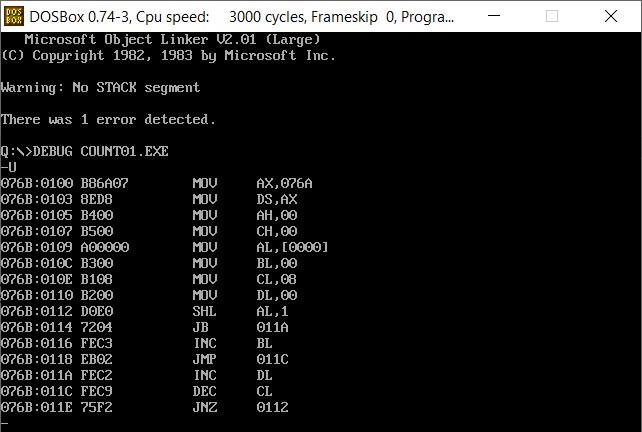
**PROGRAM – 1: 8086 ALP – COUNT NUMBER OF 0s AND 1s IN AN 8-BIT NUMBER:**

**ALGORITHM:**

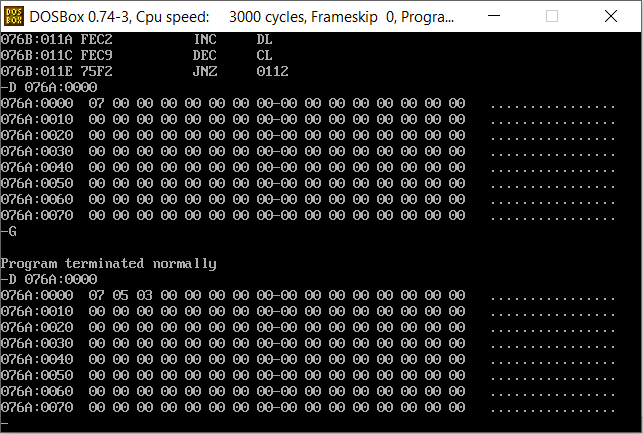
1. Begin.
2. Initialize the data segment.
3. Initialize a variable, say *“num”*, to store the 8-bit number.
4. Initialize variables, say *“zeroes”* and *“ones”* to store the number of zeroes and ones in the 8-bit number as output.
5. Close the data segment.
6. Initialize the code segment.
7. Load the 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 ≠ 0:
    1. Shift left the value in AL register by 1.
    2. If CF ≠ 1:
       1. BL ← BL + 1.
    3. Else:
       1. DL ← DL + 1.
    4. CL ← 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 | *NUM* = 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 ← *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 *ONE.* |
| INC BL | ELSE, INCREMENT ZEROES COUNT, I.E BL. |
| JMP SKIP | GO TO *SKIP.* |
|  |  |
| ONE: |  |
| INC DL | DL = DL + 1. |
|  |  |
| SKIP: |  |
| DEC CL | CL = CL + 1. |
| JNZ LOOPER | IF CL ≠ 0, GO BACK TO *LOOPER.* |
|  |  |
| MOV ZEROES, BL | *ZEROES* ← BL. |
| MOV ONES, DL | *ONES* ← DL. |
|  |  |
| MOV AH, 4CH | TERMINATE THE PROGRAM WITH DOS INTERRUPT. |
| INT 21H |  |
|  |  |
| CODE ENDS |  |
|  |  |
| END START |  |

**UNASSEMBLED CODE:**

****

**SAMPLE I/O SNAPSHOT:**

****

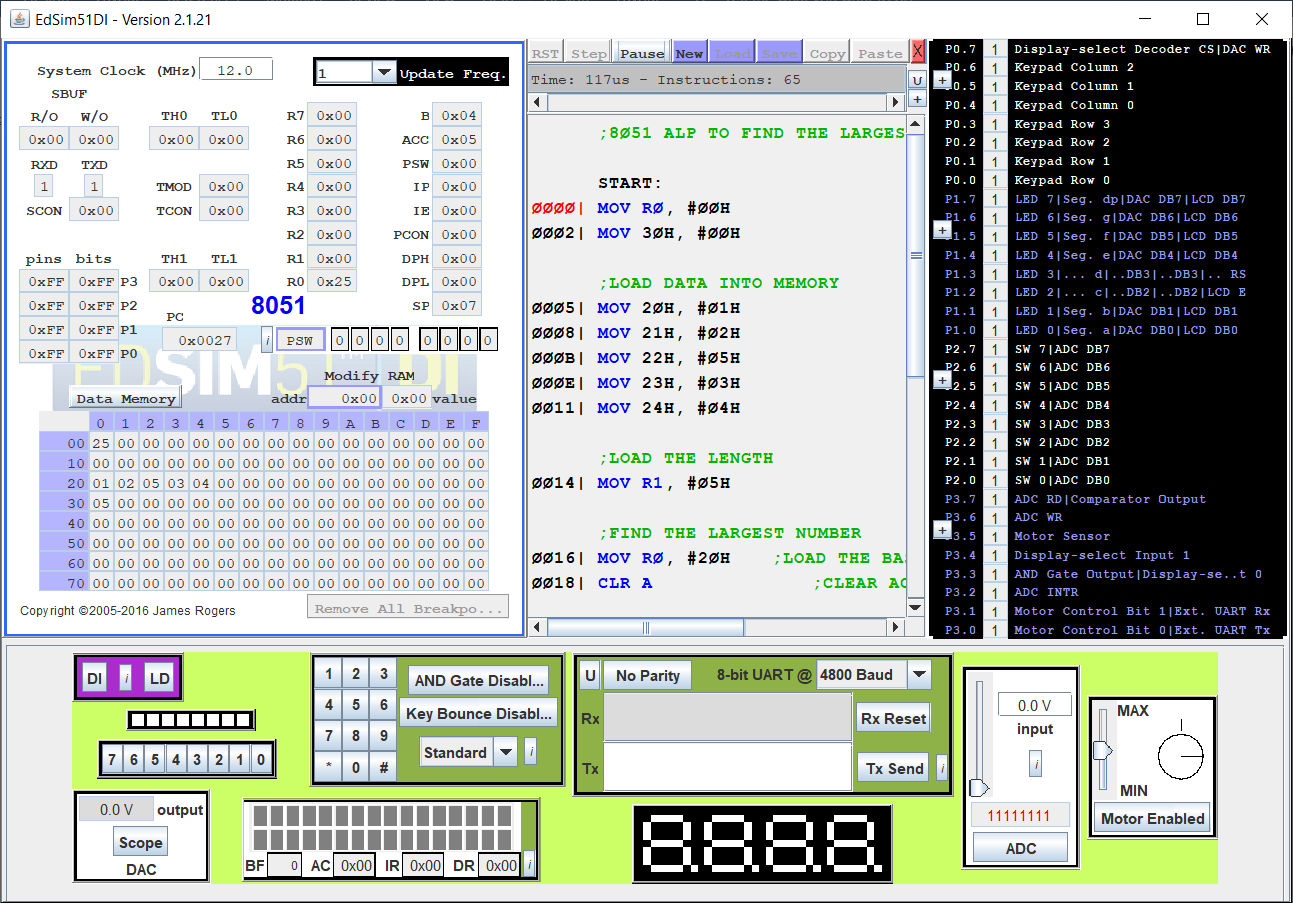
**PROGRAM – 2: 8051 ALP TO FIND THE LARGEST NUMBER IN A GIVEN LIST:**

**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 ≠ 0:
   1. Transfer to B the value pointed by the address stored at R0.
   2. Compare the values of A and B.
   3. If A < B:
      1. A ← B.
   4. Else:
      1. R0 ← R0 + 1.
   5. R1 ← 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 ≠ B, JUMP TO NEXT. |
|  |  |
| NEXT: |  |
| JNC SKIP | IF A < B, THEN CARRY FLAG = 1. ELSE CARRY FLAG = 0. |
| MOV A, B | A ← B. A IS THE CURRENT LARGEST VALUE. |
|  |  |
| SKIP: |  |
| INC R0 | GOTO NEXT ELEMENT BY INCREMENTING R0 BY 1. |
| DJNZ R1, BACK | IF R1 ≠ 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:**

****

**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.