# END SEMESTER MP LAB PRACTICALS

**Name:** Rahul Ram M

**Roll No.:** 185001121

**Date:** 17/11/2020

## 29.a. Write an ALP using 8051 to sort a list of numbers in ascending order.

## Aim:

To write an ALP using 8051 to sort a list of numbers in ascending order.

## Algorithm:

- Move the length of the list to R1.
- Move the value of R1 to register A and from register A to R2.
- LOOP1: Move the value of R1 to register A.
- Decrement the value of register A and move the value of register A to R3.
- Move the starting location(10H) of the list to R0.
- LOOP2: Move the contents stored in the memory location specified by R0 to register A.
- Increment R0 to move to next location.
- Now move the contents stored in the memory location specified by R0 to register B.
- Clear the carry flag.
- Subtract A from B.
- If carry is produced, value at A and B are already in order so move to SKIP.
- Else A and B are out of order.
- Decrement R0 to move one step back in the memory.
- Copy the contents at that memory location to register A and store the value of register B to that location.
- Increment R0 to move to the next memory and store the value of register A to that location.
- SKIP: Using DJNZ decrement the value of R3 and check if it is 0. If not loop to LOOP2 else execute the next instruction.
- Using DJNZ decrement the value of R2 and check if it is 0. If not loop to LOOP1 else execute the next instruction.
- END: Using SJMP END, create an infinite loop to END by unconditional jump to END.

## Code:

```
MOV R1, #06H
```

```
MOV A, R1
MOV R2, A
LOOP1:      MOV A, R1
DEC A
MOV R3, A
MOV R0, #010H
LOOP2:      MOV A, @R0
INC R0
MOV B, @R0
CLR C
SUBB A, B
JC SKIP
DEC R0
MOV A, @R0
MOV @R0, B
INC R0
MOV @R0, A
SKIP:   DJNZ R3, LOOP2
DJNZ R2, LOOP1
END:    SJMP END
```

## Output:

**Unsorted:**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 10 | 6F | 5C | 34 | 72 | 2D | 12 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 20 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 30 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 50 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 60 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 70 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

**Sorted:**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 15 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 10 | 12 | 2D | 34 | 5C | 6F | 72 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 20 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 30 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 50 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 60 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 70 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

## Result:

ALP using 8051 to sort a list of numbers in ascending order is executed successfully.

## 29.b. Write an ALP using 8086 to find the number of ones and zeros in a 16-bit number.

## Aim:

To write an ALP using 8086 to find the number of ones and zeros in a 16-bit number.

## Algorithm:

- START: Move the starting address of data segment to AX register and move the data from AX register to DS register
- Initialize the BX register to 0.
- Copy the value of num to AX register.
- Move 10H (16) to CX register.
- L1: Using RCR (rotate through carry right), rotate the bits to the right in the AX register including the carry by 1.
- So, the least significant bit of AX register will move to CF.
- Using JC, check if the CF contains 0 or 1. If it is 0, jump to ONE.
- Else increment BL (number of zeros) and take unconditional jump to HERE.
- ONE: increment BH register (number of ones).
- HERE: Loop through l1 till CX becomes 0.
- Move the value of BL register to zeros and BH register to BH.
- Using INT21H with AH value as 4CH, terminate the program.

## Code:

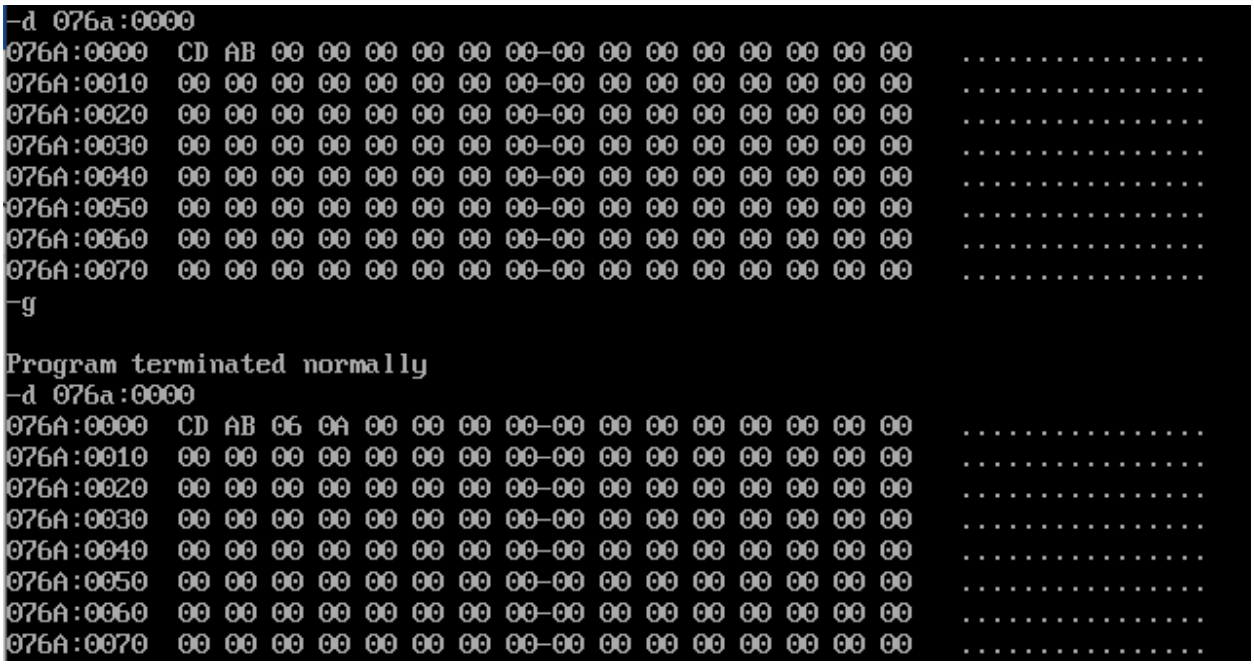
; counting number of zeros and ones in a 16-bit number

assume cs:code, ds:data

```
data segment
num dw 0abcdh
zeros db 00h
ones db 00h
data ends

code segment
org 0100h
start:    mov ax, data
mov ds, ax
mov bx, 0000h
mov ax, num
mov cx, 0010h
l1:       rcr ax, 1
jc one
inc bl
jmp here
one:      inc bh
here:     loop l1
mov ones, bh
mov zeros, bl
mov ah, 4ch
int 21h
code ends
end start
```

## Output:




## Result:

ALP using 8086 to find the number of ones and zeros in a 16-bit number is executed successfully.