Name: **Tushar Nankani** Roll No: **1902112** Batch: **C23**

**Microprocessor: Experiment 4**

**Aim**: Write an assembly language program to display the contents of 16 bit flag register.

**Theory:**

The Flag register is a Special Purpose Register. Depending upon the

value of result after any arithmetic and logical operation the flag bits

become set (1) or reset (0). In 8085 microprocessor, flag register consists

of 8 bits and only 5 of them are useful.

**1 Sign (S) flag: –**

This flag is set, when MSB (Most Significant Bit) of the result is 1.

Since negative binary numbers are represented in the 8085 CPU in

standard two’s complement notation, SF indicates sign of the result.

1-MSB is 1 (negative)

0-MSB is 0 (positive

If the MSB of the result of an operation is 1, this flag is set, otherwiseit

is reset.

**2 Zero (Z) flag: –**

This flag is set, when the result of operation is zero, else it is reset. 1-zero result

0-non-zero result

**3 Auxiliary Carry (AC) flag: –**

This flag is set whenever there has been a carry out of the lower nibble

into the higher nibble or a borrow from higher nibble into the lower

nibble of an 8 bit quantity, else AF is reset. This flag is used by decimal

arithmetic instructions.

1-carry out from bit 3 on addition or borrow into bit 3 on subtraction

0-otherwise

If there is a carry out of bit 3 and into bit 4 resulting from the execution

of an arithmetic operation, it is set otherwise reset. This flag is used for

BCD operation and is not available to the programmer to changethe

sequence of an instruction.

**4 Carry (CY) flag:** – If an instruction results in a carry (for addition

operation) or borrow(for subtraction or comparison) out of bit D7, then

this flag is set, otherwise reset.

This flag is set whenever there has been a carry out of, or a borrow into,

the higher order bit of the result. The flag is used by the instructions that

add amd subtract multibyte numbers.

1-carry out from MSB bit on addition or borrow into MSB bit on

subtraction

0-no carry out or borrow into MSB bit

**5 Parity (P) flag:–**

This flag is set whenever the result has even parity, an even number of 1

bits. If parity is odd, PF is cleared. 1-low byte has even number of 1 bits

0-low byte has odd parity

This flag is set when the result of an operation contains an even number

of 1’s and is reset otherwise.

**Control Flags** – The control flags enable or disable certain operations of

the microprocessor. There are 3 control flags in 8086 microprocessor

and these are:

**Directional Flag (D)** – This flag is specifically used in string

instructions.

If directional flag is set (1), then access the string data from higher

memory location towards lower memory location.

If directional flag is reset (0), then access the string data from lower

memory location towards higher memory location.

**Interrupt Flag (I)** – This flag is for interrupts.

If interrupt flag is set (1), the microprocessor will recognize interrupt

requests from the peripherals.

If interrupt flag is reset (0), the microprocessor will not recognize any

interrupt requests and will ignore them.

**Trap Flag (T)** – This flag is used for on-chip debugging. Setting trap

flag puts the microprocessor into single step mode for debugging. In

single stepping, the microprocessor executes a instruction and enters into

single step ISR.

If trap flag is set (1), the CPU automatically generates an internal

interrupt after each instruction, allowing a program to be inspected as it executes instruction by instruction. If trap flag is reset (0), no function is

performed.

The control and status signals are ALE, RD, WR, IO/M, S0, and S1 and

READY, The interrupt signals are TRAP, RST 7.5, RST 6.5, RST 5.5,

INTR. INTA is an interrupt acknowledgement signal indicating that the

processor has acknowledged an INTR interrupt.Serial I/O signals are

SID and SOD, DMA signals are HOLD and HLDA, and Reset signals

are RESET IN and RESET OUT.

The PUSHF (push flags) and PUSHFD (push flags double) mnemonics

reference the same opcode. The PUSHF instruction is intended for use

when the operand-size attribute is 16 and the PUSHFD instruction for

when the operand-size attribute is 32. Some assemblers may force the

operand size to 16 when PUSHF is used and to 32 when PUSHFD is

used. Others may treat these mnemonics as synonyms

(PUSHF/PUSHFD) and use the current setting of the operand-size

attribute to determine the size of values to be pushed from the stack,

regardless of the mnemonic used

When in virtual-8086 mode and the I/O privilege level (IOPL) is less

than 3, the PUSHF/PUSHFD instruction causes a general protection

exception (#GP).w

In the real-address mode, if the ESP or SP register is 1, 3, or 5 when the

PUSHA/PUSHAD instruction is executed, the processor shuts down due

to a lack of stack space. No exception is generated to indicate this

condition

**Algorithm:**

1.Start

2.Initialize data segment through AX register in the DS register.

3.Display the flag bit names as “X X X X O D I T SF ZF x AF X PF X CF ”

4.Push the contents of flag register to a stack

5.Pop the contents of stack to register to any 16 bit register (say BX =0000 0100 1000 1001)

6.Move the contents of BX to temporary variable say t

7.Move the 8000h number to AX.(AX8000h)

8.Move the count as 16(in decimal) to CX register (as 16 bit flag register)

9.Move the contents of temporary variable t to BX.

10.And the contents of BX and AX.

11.If zero flagis set then gotothe step no 14 otherwise goto step no. 12

12.Move the 31h to DL register.

13.Make the unconditional jump to a step no. 15

14.Move the 30h to DL register.

15.Preserve the (8000h )number from AX in t1 temporary variable.(As while displaying 30h or 31 h AH register get modified as 02h function is moved of INT 21h).

16.Display the contents of DL register.

17.Move the contents of t1 to AX registerback (As while displaying 30h or 31 h AH register get modified as 02h function is moved of INT 21h).

18.Rotate the contents of AX by 1 positions in right direction.

19.Repeat step no 5 to 17 till count CX reaches to 0.

20.Stop.

**CODE:**

data segment

msg1 db 0dh,0ah, "DISPLAY FLAG CONTENTS$";

msg2 db 0dh,0ah, "1.BEFORE SETTING DF AND RESETTING IF : $";

msg3 db 0dh,0ah, "2.AFTER SETTING DF AND RESETTING IF : $";

msg4 db 0dh,0ah, "XX XX XX XX OF DF IF TF SF ZF XX AF XX PF XX CF$";

msg5 db 0dh,0ah, "$";

flag dw ?

rotate dw ?

data ends

code segment

assume cs:code,ds:data

start:

mov ax,data

mov ds,ax

mov dx,offset msg1

mov ah,09h

int 21h

mov dx,offset msg2

mov ah,09h

int 21h

call disp

std

cli

mov dx,offset msg3

mov ah,09h

int 21h

call disp

mov ah,4ch

int 21h

disp proc

mov dx,offset msg4

mov ah,09h

int 21h

mov dx,offset msg5

mov ah,09h

int 21h

pushf

pop bx

mov flag,bx

mov dx,8000h

mov rotate,dx

mov cx,10h

next:

and bx,rotate

jz zero

mov dl,31h

mov ah,02h

int 21h

jmp space

zero:

mov dl,30h

mov ah,02h

int 21h

space:

mov dl,""

mov ah,02h

int 21h

mov dl,""

mov ah,02h

int 21h

mov bx,flag

ror rotate,01h

loop next

endp

code ends

end start

**OUTPUT:**

