KHULNA UNIVERSITY OF ENGINEERING AND TECHNOLOGY, KUET

SESSIONAL REPORT

Course No: CSE 2204

Department of: Computer Science and Engineering

Experiment No: **13**

Name of the Experiment: To write an assembly program which convert a BCD number to Binary number.

Remarks

Date of Performance: 31.05.21

Date of Submission: 02.06.21

Name: Rifat Arefin

Roll: 1807117

Year: 2nd

Semester: 2nd

No. of experiment: 13

Name of experiment: To write an assembly program which convert a BCD number to Binary number.

Objectives:

1. To get introduced with CALL instruction and its classification.
2. To learn about procedure.
3. To learn and implement how to convert a BCD number to Binary number.

Introduction:

In the program, we use CALL instruction to transfer execution to a procedure. There are two basic types of calls, near and far.

A near call is a call to a procedure

which is in the same code segment as the CALL instruction. When the 8086 executes a neare CALL instruction, it decrements the stack pointere by 2 and copies the offset of the next instruction after the call on to the stack. A RET instruction at the end of the procedure will return execution to the instruction after the call by copying the offset ~~set~~ saved back to IP.

A fare call is a call to a procedure which is in different section. When the 8086 executes a fare call, it decrements the stack pointere by 2 and copies the contents of the CS register to the stack. It then decrement the stack pointere by 2 again and copies the offset of the instruction after the CALL instruction to the stack.. Finally it loads the CS with

the segment base of the segment which contains the procedure and loads IP with the offset of the first instruction of the procedure in the segment. A RET instruction at the end of the procedure will return execution to the next instruction after the CALL by restoring the saved values of CS. and IP from the stack

In this program, we use procedure. A procedure is a set of code that can be branched to and returned from in such a way that the code is as if it were inserted at the point from which it is branched to.

In this program, we use 'and', 'ror' instructions. Using 'AND' instruction we perform logical and

operation.

Again ROR instruction is used to rotate from LSB to MSB.

Example:

```
MOV  AL, 1Ch   ; AL = 00011100 b
ROR  AL, 1     ; AL = 00001110 b
RET
```

Apparatus Required: emu 8086, Laptop.

Methodology:

Code:

```
org 100h
Call  BCD to Bin ; call procedure 'BCD to Bin'
                   the execution jump to the
                   procedure 'BCD to Bin'
ret
```

```
BCD to Bin  proc near   ; Starting of a near
                        ~~call~~ procedure "BCD to Bin"
  pushf                 ; store the conditions of flags to stack
  mov al, 56h ;  al is initialized with 56h.
  mov bl, al  ;  copy ~~move~~ the value of al to bl

  and bl, 00Fh ; 'and' with 00Fh to the value
                 of bl register to
  And ~~bl~~, 0F0h ;  Perform AND operation ~~with~~ to
                      al register value with 0F0h.
  ror al, 04h  ;  We use ROR instruction to al
                  register for 4 times because
                  If we perform AND operation
                  then, AL = 50h = 11110000
                  So if we rotate right for 4
                  times then, AL = 5h
  mov cl, 0Ah
  mul cl       ;  perform multiplication ~~with~~ to the
                  value of al with the value of
                  cl.
```

```
add   al, bl   ; perform addition ~~bit~~ to the
                 value of al with the value of bl.
popf           ; restore the initial conditions of flags, before
                 the procedure performed
ret
BCDtoBin  endp ; end of procedure.
```

Result and Discussion :

Through the ~~instruction~~ experiment, we performed BCD number to Binary number. We carefully implemented according to algorithm to perform our task properly. From this experiment, we used CALL instructions for procedure. We learnt procedure perfectly from this experiment. We also tried for various inputs to check our program.

By checking with various inputs, we ensured that our program was perfect.

Conclusion: Procedure is one of the basic things of the assembly language programming. We tried to learn procedure perfectly from the experiment. So it would be very much necessary matter for us. to perform solving various problem. We learnt some new instructions. And by ensuring our experiment ~~com~~ to be correct, we can say that we have clear idea about procedures and other new instructions and we have fullfilled our objectives properly

References:

1. Microprocessor and Interfacing - by D.V. Hall
2. C:/emu8086/documentation/index.html.