LAB 2: Programming in Assembly Language

Objective

1. Understanding some of the instructions set: MOV, ADD, LOOP,

2. Practice Assembly Language Programs
In Lab 1, you have been introduced to BG 8088, its features and monitor commands to operate BGC 8088 with an example of a small assembly language program. Could you recall the use of monitor commands `a`, `i`, `u` and `g`.

In today’s lab, we will move further ahead to understand more on Intel’s Instruction Set, which will be used throughout this course. More monitor commands will be introduced. These monitor commands are used to deal with memory location. I.e. Writing into memory location via software, entering values into memory location.

Programs are collections or set of instructions, which has some dedicated functions and meanings. Assembly language is one kind of programming language, which is different from others. It is a low level language where the programmer interacts directly with hardwares. It gives more control over the hardware with few commands compare to high-level languages in microcomputers with less amount of memory.

Instructions are commands which direct the microprocessor to do some task. Each instruction contains an opcode and operand. Opcode is the command in form of mnemonics while operand is data to be processed. There are source operand and destination operands.

**Basic and Commonly used Instruction Sets**

- **MOV AX, 0**: Move to register AX value of 0 (hexa)
- **INC AX**: Increase the content of AX by 1
- **DEC AX**: Decrease the content of AX by 1
- **ADD AX, BX**: Add the content of BX to AX and store in AX
- **SUB AX, BX**: Subtracts the contents of memory location, whose address is in BX, from the value in AL and stores the result in AL
- **LOOP address**: Check for register CX, decrement it by one and check if its not equal 0 (zero), jump to address specified else if CX is 0, then proceed with next instruction

**Instruction set related to memory location**

- **Mov [123],ax**: Copy the contents of AX to memory location 123
- **Mov ax,[123]**: Copy the contents of memory location 123 to AX
- **Mov ax,[bx]**: Copy the contents of memory location, whose address is in BX, to AX

**More Monitor Commands (For Memory)**

In order to write programs that access memory, you should be able to display or write to memory. Following is the monitor commands which used to access the memories.

- ‘f’ command fills a range if memory locations with a certain value 
  i.e.:  `f start_address, last_address, fill_value`
b. ‘e’ command is used to enter a single memory location
   i.e. e address, value

c. ‘d’ command displays the content of a range of memory address
   i.e. d start_address, last_address

**Step 1 : Based on the above monitor commands**

a. Fill in the memory block from 100 (h) to 10F (h) with value 1
b. Fill in memory location 105 (h) with value 5
c. Display their contents. What you observe. How many bits is each memory location?
   How many ‘digits’ represents each memory location?

**Step 1 : The LOOP statement**

Type in the following program and execute it. ++ refers to the address of the statement ‘inc ax’.

```
    Mov ax,0
    Mov cx,10
    ++ Inc ax
    Loop ++
```

a. Run the program. Which of the registers do you expect to be effected by the program? Explain the flow of the program.
Step 3. Repetition using LOOP
The following program writes the value 01 into memory locations from 0100 to 010F.

```
Mov ax,1
Mov bx,0100
Mov cx,f
** Mov [bx],AL
inc bx
Loop **
```

b. Explain the flow of program with a flow chart.

c. Run the program. Display memory location from 0100 to 010F.

c. Explain why the instruction ‘MOV [BX],AL’ was used instead of ‘MOV [BX],AX’.
d. Program

   Improve the program so that it writes the values from 00 to 0F successively into the addresses instead of always writing 01.

Improved Program

Step 4: Reading to and writing from memory

Enter two values into addresses of your choice. Write a program that reads the values from those addresses, adds the two values, and then writes the result to a new address. (The address space is from 0000 to 0FFF.) Verify the results by displaying the contents of the addresses.

Address Chosen: __________________________________
Value Chosen: __________________________________
e. Program:

f. Execute your program and verify your result.
Questions