



## Department of Computer Science and Engineering

<b>Course Code: CSE341</b>	<b>Credits: 1.5</b>
<b>Course Name: Microprocessors</b>	<b>Semester: Fall'18</b>

### Lab 05

#### Flow control instructions and Looping structures

##### I. Topic Overview:

Alongside the capability of making decisions, any functional program should also have a mechanism to repeat sections of code. In this lab, students will familiarize themselves with the loop instructions to achieve that. Alongside jump, the loop instruction is also used to transfer control to another part of the program. The students will then learn to implement different looping structures. This application will make it much easier to convert a pseudo code algorithm to assembly code.

##### II. Lesson Fit:

There is prerequisite to this lab. Students must have a basic idea on the following concepts:

- a. Jump instruction
- b. Some basic operations such as MOV, ADD, SUB, MUL and DIV
- c. Basic I/O operations
- d. Character encoding using ASCII

##### III. Learning Outcome:

After this lecture, the students will be able to:

- a. Control flow of the program
- b. Use loops to avoid repetition

#### IV. Anticipated Challenges and Possible Solutions

- a. Students may find it difficult to visualize how the program control flow changes when using loop/jump

**Solutions:**

- i. Step by step simulation
- b. Directly coding in assembly may come off as challenging

**Solutions:**

- i. Writing the pseudocode first then then converting it to assembly may help

#### V. Acceptance and Evaluation

Students will show their progress as they complete each problem. They will be marked according to their class performance. There may be students who might not be able to finish all the tasks, they will submit them later and give a viva to get their performance mark. A deduction of 30% marks is applicable for late submission. The marks distribution is as follows:

Code: 50%

Viva: 50%

#### VI. Activity Detail

- a. **Hour: 1**

**Discussion: Looping Structure**

Just like for and while loops in high level programming languages, loops can also be implemented in assembly. There are 2 ways of implementing a loop: explicit and implicit.

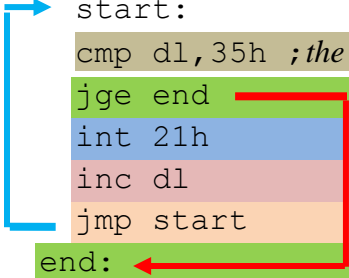
- **Explicit:** By using compare and jump instructions to decide whether to enter the loop or not and by using the **inc /add /sub** instruction for increments.

## Java

```
int x = 0;
while (x < 5){
    System.out.println(x);
    x++;
}
```

## Assembly

```
mov ah,2
mov dl,30h ; the emu8086 uses hexadecimal. 30h is 0
start:
    cmp dl,35h ; the emu8086 uses hexadecimal. 35h is 5
    jge end
    int 21h
    inc dl
    jmp start
end:
```



- **Implicit:** In this case we do not have to **check** whether the counter has reached the limit or not. This will be done automatically. The instructions will be **loop**. **loop destination\_line** is the syntax. CX will be used as the counter always in order for the loop instruction to execute.

## JAVA

```
int x = 0;
while (x < 5){
    System.out.println(x);
    x++;
}
```

## Assembly

```
mov cx,5 ;the bound will be in cx. (the number of
times the loop will run)
```

```
mov dl,30h
```

```
mov ah,2
```

```
start:
```

```
    int 21h
```

```
    inc dl
```

```
loop start
```

**NB.** CX will always start from the specified count and will always decrement by 1.

**Problem Task:** Task 01-Task03 (Page 5)

b. **Hour: 2**

**Discussion:** Discuss the properties of a repeat-until loop structure.

- **Repeat Until loop:** Repeat the loop until a condition is satisfied. For example you have been asked to take in characters from the user and print them until a space is pressed.

```
.code
repeat:
mov ah,1
int 21h
mov ah,2
mov dl,al
int 21h
cmp al,' '
jne repeat
```

**Problem Task:** Task 04 – 06 (Page 5-6)

c. **Hour: 3**

**Discussion:**

Check progress while the students carry on with the rest of the tasks.

**Problem Task:** Task 07 (Page 6-7)

**VII. Home Tasks:** All the unfinished lab tasks.

### **Lab 5 Activity List**

#### **Task 01**

Write a count-controlled loop to display a row of 80 stars.

#### **Task 02**

Write a sequence of instructions to do each of the following:

a. Put the sum of  $1 + 4 + 7 + \dots + 148$  in AX.

b. Put the sum  $100 + 95 + 90 + \dots + 5$  in AX.

#### **Task 03**

Read a five character password and overprint it by executing a carriage return and displaying five X's. You need not store the input characters anywhere.

#### **Task 04**

The following algorithm may be used to carry out multiplication of two positive numbers M and N by repeated addition.

Initialize product to 0

REPEAT

add M to product

decrement N

UNTIL N equals 0

Write a sequence of instructions to multiply AX by BX, and put the product in CX. You may ignore the possibility of overflow.

#### **Task 05**

Write a program to display the extended ASCII characters (ASCII codes 80h to FFh). Display 10 characters per line, separated by blanks. Stop after the extended characters have been displayed once.

#### **Task 06**

Write a program that will prompt the user to enter a hex digit character ("0" ... "9" or "A" ... "F"), display it on the next line in decimal, and ask the user if he or she wants to do it again. If the user types "y" or "Y", the program repeats; If the user types anything else, the program terminates. If the user enters an illegal character, prompt the user to try again

Sample execution:

ENTER A HEX DIGIT: 9

IN DECIMAL IT IS 9

DO YOU WANT TO DO IT AGAIN? y

ENTER A HEX DIGIT: c

ILLEGAL CHARACTER - ENTER 0 .. 9 OR A .. F: C

IN DECIMAL IT IS 12

DO YOU WANT TO DO IT AGAIN? N

#### **Task 07**

[Hard] Write a program that reads a string of capital letters, ending with a carriage return, and displays the longest sequence of consecutive alphabetically increasing capital letters read. Sample execution:

ENTER A STRING OF CAPITAL LETTERS:

FGHADEFGHC

THE LONGEST CONSECUTIVELY INCREASING STRING IS:

DEFGH