মেট্রোপলিটন ইউনিভার্সিটি
Metropolitan
University

Department of Computer Science and Engineering

Course Code: CSE238	Credits: 1.5
Course Name: Microprocessor & Interfacing Lab	Faculty: FRS

Lab 04

Flow control instructions and Branching Structures

Discussion:

Conditional Jumps:

```
if (x > 5){

// code

}
```

In the line if (x > 5), a comparison is done between the content of x and 5. The decision whether to execute the enclosed code or not depends on the result of the comparison. In assembly the comparison is done by a piece of code called CMP. The syntax of CMP is CMP destination, source. The comparison is done by destination - source. The result is not saved anywhere but affects the flags. The result of subtraction can be 0, AX > BX or AX < BX. Below is a piece of code.

```
MOV AX, first_number

MOV BX, second_number

CMP AX, BX
```

The comparison is done in the third line. Now there could be 5 possibilities AX == BX, AX > BX, AX < BX, AX >= BX, AX <= BX. We will take the decision based on one of these options. This decision is performed by "Jump" instruction denote as "J".

Condition	Jump Instruction	Explanation
AX == BX	JE	jump if destination and source are equal
AX > BX	JG	jump if destination > source
AX < BX	JL	jump if destination < source
AX >= BX	JGE	jump if destination >= source
AX <= BX	JLE	jump if destination <= source

But jump where???

Jump in that line which we want to execute when one of the above conditions is satisfied. How will we get the line number? We do not have to worry about the line numbers because we will name the line(s) ourselves.

```
int x = 5;

if (x > 4) {
    //code
}
```

```
MOV AX, 5
CMP AX, 4
JG My_Line

My_Line:
;code
```

Node the declaration of the line name ends with a colon (:)

Unconditional jump:

For skipping a portion of code, we use/need unconditional jump. There is no comparison, it is just a jump from one line to the one we want. The instruction is called JMP and its syntax is JMP destination_line_name.

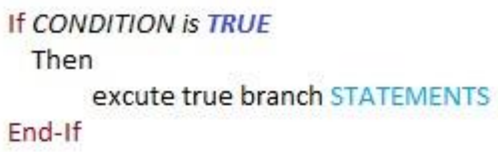
```
MOV AX, 5
MOV BX, 8
JMP My_Line
MOV AH, 4
MOV DL, 6
My_Line:
MOV DL, 7
```

In the above code, when the 3rd line is executing the program jumps to the line named My_Line without executing the codes in between, This is how codes are skipped.

Branching Structures:

In high-level languages, branching structures enable a program to take different paths, depending on conditions. In this section, we'll look at three structures.

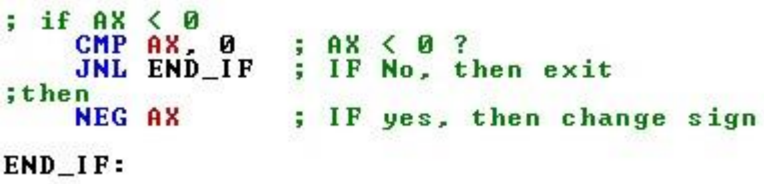
If-Then: The If-Then structure may be expressed in pseudo code as follows,



```
If CONDITION is TRUE
  Then
        excute true branch STATEMENTS
End-If
```

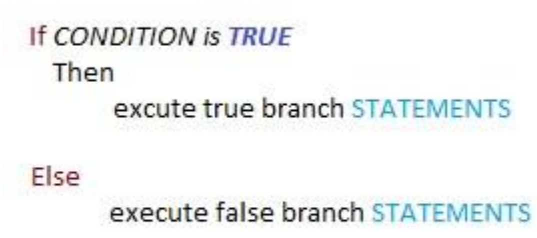
The condition is an expression that is true or false. If it is true, the true-branch statements are executed. If it is false, nothing is done, and the program goes on to whatever follows.

Example: Replace the number in AX by its absolute value Solution:

A pseudocode algorithm is:	**It can be coded as follows:**
If AX < 0 Then replace AX by -AX End-If	 ; if AX < 0 CMP AX, 0 ; AX < 0 ? JNL END_IF ; IF No, then exit ;then NEG AX ; IF yes, then change sign END_IF:

The condition AX < 0 is expressed by CMP AX, 0. If AX is not less than 0, there is nothing to do, so we use a JNL (jump if not less) to jump around the NEG AX. If condition AX < 0 is true, the program goes on to execute NEG AX.

Branching Structures (If-Then-Else):



```
If CONDITION is TRUE
  Then
        excute true branch STATEMENTS

Else
        execute false branch STATEMENTS
```

The JMP instruction comes in use when if - else condition is employed. Let us see the use of conditional and unconditional jumps together in a program where we will find the greater of 2 numbers.

Java Solution:

```
System.out.println("Enter the first number: ");
int x = sc.nextInt();

System.out.println("Enter the second number: ");
int y = sc.nextInt();

if (x > y) {
    System.out.println(x + " is greater");
} else {
    System.out.println(y + " is greater");
}
```

Assembly Solution:

```
data segment
  pkey db "press any key...$"
  a db "Enter first number$"
  b db "Enter second number$"
  c db " is larger$"
ends

stack segment
  dw 128 dup(0)
ends

code segment
start:
  ;set segment registers
  mov ax, data
  mov ds, ax
  mov es, ax
  ;add your code here

  lea dx, a  ;Print a
  mov ah, 9
  int 21h

  mov ah, 1
  int 21h
  mov bl, al ;move input to bl

  lea dx, b  ;Print b
  mov ah, 9
  int 21h

  mov ah, 1
  int 21h
  mov cl, al ;move input to cl

  cmp cl, bl ;compare two inputs
  jg cl_greater

  mov dl, bl
  mov ah, 2
  int 21h

  lea dx, c
  mov ah, 9
  int 21h
  jmp end

  cl_greater:
    mov dl, cl
    mov ah, 2
    int 21h

    lea dx, c
    mov ah, 9
    int 21h

  end:
    lea dx, pkey
    mov ah, 9
    int 21h  ;print pkey

    ;wait for any key...
    mov ah, 1
    int 21h

  ;exit to DOS
  mov ax, 4c00h
  int 21h

ends
end start ;set entry point and
stop the assembler.
```

Tell us why was the “JMP end‟ statement crucial in this program.

Case:

A CASE is a multiway branch structure that tests a register, variable, or expression for particular values or a range of values. The general form is as follows:

```
CASE expression
Values_1: statements_1
Values_2: statements_2
..
..
Values_n: statements_n
END_CASE
```

In this structure, expression is tested; if its value is a member of the set values_1, then statements_1 are executed.

Problem Solving

Task 01

Take a number in AX, and if it’s a negative number, replace it by 5.

Task 02

Suppose AL and BL contain extended ASCII characters. Display the one that comes first in the character sequence.

Task 03

If AX contains a negative number, put -1 in BX; if AX contains 0, put 0 In BX; if AX contains a positive number, put 1 in BX.

Task 04

If AL contains 1 or 3, display “o”; if AL contains 2 or 4 display "e".

Task 05

Read a character, and if it's an uppercase letter, display it.

Task 06

Read a character. If it's "y" or "Y", display it; otherwise, terminate the program.

Task 07

Write an assembly program to check whether a number is even or odd.

Task 08

Write a program to input any alphabet and check whether it is vowel or consonant.

Task 09

Write a program to check whether a number is divisible by 5 and 11 or not.

Task 10

Write a program to find the maximum and minimum between three numbers.

Sample execution: Input: 2 3 4

Output: Maximum number is 4

Minimum number is 1

Task 11

Write a program that takes as input all sides of a triangle and check whether triangle is valid or not. If the sides form a triangle, print “Y”, otherwise print “N”.

Task 12

Write a program that takes a digit as an input and outputs the following. If the digit is within 0-3, it prints “i”, If it’s within 4-6, it prints “k”, If it’s within 7-9, it prints “l” and if it’s 10, it prints “m”.

Task 13

Write a case to print the name of the day of the week. Consider the first day of the week is Saturday.

Sample execution: Input: 3

Output: Monday

Task 14

Write a case to print the total number of days in a month.

Sample execution: Input: 3

Output: 31 days

Code Template

```
.MODEL SMALL

.STACK 100H

.DATA


.CODE
MAIN PROC

;iniitialize DS

MOV AX,@DATA
MOV DS,AX

;enter your code here






;exit to DOS

MOV AX,4C00H
INT 21H

MAIN ENDP
END MAIN
```