Computer Org. & Assembly Language Lab

Lab#08: Conditional Structures

Agenda

- Conditional Structure
  - Block – Structured IF Statement
  - Compound Expressions
    - Logical AND Operator
    - Logical OR Operator
  - WHILE Loops

Conditional Structures

Block – Structured IF Statements

In most high-level languages, an IF statement implies that a Boolean expression is followed by two lists of statements: one performed when the expression is true, and another is performed when the expression is false:

```
if(expression)
{
}
else
{
}
// Using C/C++ syntax, two statements are executed if OPERAND1 is equal to
OPERAND2.
//Sample Code:
if(op1 == op2)
{
      X = 1;
      Y = 2;
}
```

In Assembly language this can be achieved using the cmp instruction.

```
Include irvine32.inc

.data
      msgStr BYTE "Both numbers are equal, if statement executed",0
      op1 DWORD 12345678h
      op2 DWORD 12345678h
      X BYTE ?
      Y BYTE ?

.code
main PROC

mov eax, op1
cmp eax, op2      ;compare eax to op2

je L1             ;jump to L1, if equal
jmp L2            ;otherwise, jump to L2

L1:
      Mov edx, OFFSET msgStr
      call WriteString
      call crlf
      mov X, 1
      mov Y, 2
```

```
L2:
exit
main ENDP
END main
```

Compound Expressions

Logical AND Operator

Logical AND operator returns true only if all the conditions are true as in the below example.

```
;pseudo code
if( var1 greater than var2 AND var2 greater than var3)
      then
            do this
//C++ syntax is given as sample code:

if(a > b && b > c)
{
      X = 1;
      Y = 2;
}
```

It's equivalent code in assembly is given below.

```
Include irvine32.inc

.data
msgStr BYTE "if with Logical AND is executed",0
val1 DWORD ?
val2 DWORD ?
val3 DWORD ?
X BYTE ?
Y BYTE ?

.code
main PROC
      call readint
      mov val1, eax

      call readint
      mov val2,eax

      call readint
      mov val3,eax
```

```
      mov eax, val1
      cmp eax, val2

      jbe next
            mov eax, val2

            cmp eax, val3

            jbe next

            mov edx, OFFSET msgStr
            call WriteString
            call crlf
            mov X, 1
            mov Y, 2
next:

exit
main ENDP
END main
```

Logical OR Operator

Logical AND operator returns true if any one of the given conditions is true, as in the below example.

```
if(val1 > val2 || val2 > val3)
      X = 1
```

Equivalent Assembly Code

```
Include irvine32.inc

.data
msgStr BYTE "if with Logical OR is executed",0
val1 DWORD ?
val2 DWORD ?
val3 DWORD ?
X BYTE      ?

.code
main PROC
      call readint
      mov val1, eax

      call readint
      mov val2,eax

      call readint
```

```
      mov val3,eax

      mov eax, val1
      cmp eax, val2

      ja L1             ;Jump if above i.e. if (val1 > val2)
            mov eax, val2
            cmp eax, val3     ;compare val2 with val3

            jbe next

      L1:   mov X, 1
            mov edx, OFFSET msgStr
            call WriteString
            call Crlf
      next:

exit
main ENDP
END main
```

WHILE Loops

The WHILE structure tests a condition first before performing a block of statements. As long as the loop condition remains true, the statements are repeated. The following loop is written in C++.

```
while(val1 < val2)
{
      val1++;
      Val2--;
}
```

Equivalent Assembly Code

```
Include irvine32.inc

.data

      val1 DWORD ?
      val2 DWORD ?

.code
main PROC
      call readint
      mov val1, eax
      call readint
      mov val2,eax
```

```
      mov eax, val1     ;copy variable to eax

      _while:
            cmp eax, val2     ; if not(val1 < val2)
            jnl endwhile      ;exit the loop

            inc eax           ;increment val1 (val1++)
            dec val2    ;decrement val2 (val2--)
      jmp _while

      endwhile:
            mov val1,eax
exit
main ENDP
END main
```

Practice Session

Write a simple program to convert the given pseudo code to assembly using assembly instructions.

1.

```
if((a1 > b1) && (b1 > c1))
{
      message="Condition o- I (Greater Than) is true";
}
else if((a1 == b1) && (b1 == c1))
{
      message="Condition – II (All equal True) is true";
}
else if((a1 < b1) && (b1  < c1))
{
      message="Condition – III (Less Than) is true";
}
else
{
      message="No condition found true";
}

if((a1 > b1) OR (b1 > c1))
{
      message="Last condition (Or Operator) is true"
}
```

2.

```
if( (( a1 > b1 ) && ( b1 < c1 )) || ( d1 != e1 ) )
{
       message="First Condition is true..!";
}
Else if(a1==c1 || a <= d1)
{
       message="Second Condition is true..!";
}

else
{
       message="No Condition is true..!";
}
```