Lab Manual for Computer Organization and Assembly Language

Lab-5

Control Structures

Contents

1. Introduction

To know more about Assembly Language Control Structures. We are used to using high-level structures rather than just branches. Therefore, it's useful to know how to translate these structures in assembly, so that we can just use the same patterns as when writing, say, C code. Most common control structures which you are going to use are loops, comparison statements (if else), jumps etc.

CMP Instruction:

- Compares the destination operand to the source operand.
- CMP (Compare) instruction performs a subtraction.

Syntax:

CMP destination, source

2. Conditional Jumps

2.1 Jump Based on unsigned comparison:

Mnemonic	Description
JA	Jump if above (if *leftOp* > *rightOp*)
JNBE	Jump if not below or equal (same as JA)
JAE	Jump if above or equal (if *leftOp* ≥ *rightOp*)
JNB	Jump if not below (same as JAE)
JB	Jump if below (if *leftOp* < *rightOp*)
JNAE	Jump if not above or equal (same as JB)
JBE	Jump if below or equal (if *leftOp* ≤ *rightOp*)
JNA	Jump if not above (same as JBE)

2.2 Jump Based on signed comparison:

Mnemonic	Description
JG	Jump if greater (if *leftOp* > *rightOp*)
JNLE	Jump if not less than or equal (same as JG)
JGE	Jump if greater than or equal (if *leftOp* ≥ *rightOp*)
JNL	Jump if not less (same as JGE)
JL	Jump if less (if *leftOp* < *rightOp*)
JNGE	Jump if not greater than or equal (same as JL)
JLE	Jump if less than or equal (if *leftOp* ≤ *rightOp*)
JNG	Jump if not greater (same as JLE)

Assembly language programmers can easily translate logical statements written in C++/Java into assembly language. For example:

```
if (op1 = = op2)
  X =
else
  X =
```

```
        mov eax,
        cmpeax,op
        jne
        mov
        jmp
L1      mov
L2
```

Implement the following pseudocode in assembly language. All values are 32- bit signed integers:

```
if( var1 <= var2 )
  var3 = 10;
else
{
  var3 = 6;
  var4 = 7;
}
```

```
        mov eax,var1
        cmp eax,var2
        jle L1
        mov var3,6
        mov var4,7
        jmp L2
L1:     mov var3,10
L2:
```

Compound expression with AND:

```
if (al > bl) AND (bl > cl)
    X = 1;
```

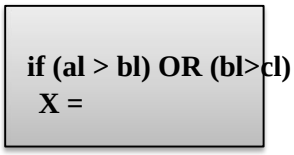
This is another possible implementation

```
        cmp al,bl     ; first expression...
        jbe next      ; quit if false
        cmpbl,cl      ; second expression...
        jbe next      ; quit if false
        mov X,1       ; both are true
next:
```

This is one possible implementation . . .

```
        cmp al,bl     ; first expression...
        ja  L1
        jmp next
L1:
        cmp bl,cl     ; second expression...
        ja  L2
        jmp next
L2:                   ; both are true
        mov X,1       ; set X to 1
next:
```

Compound expression with OR:



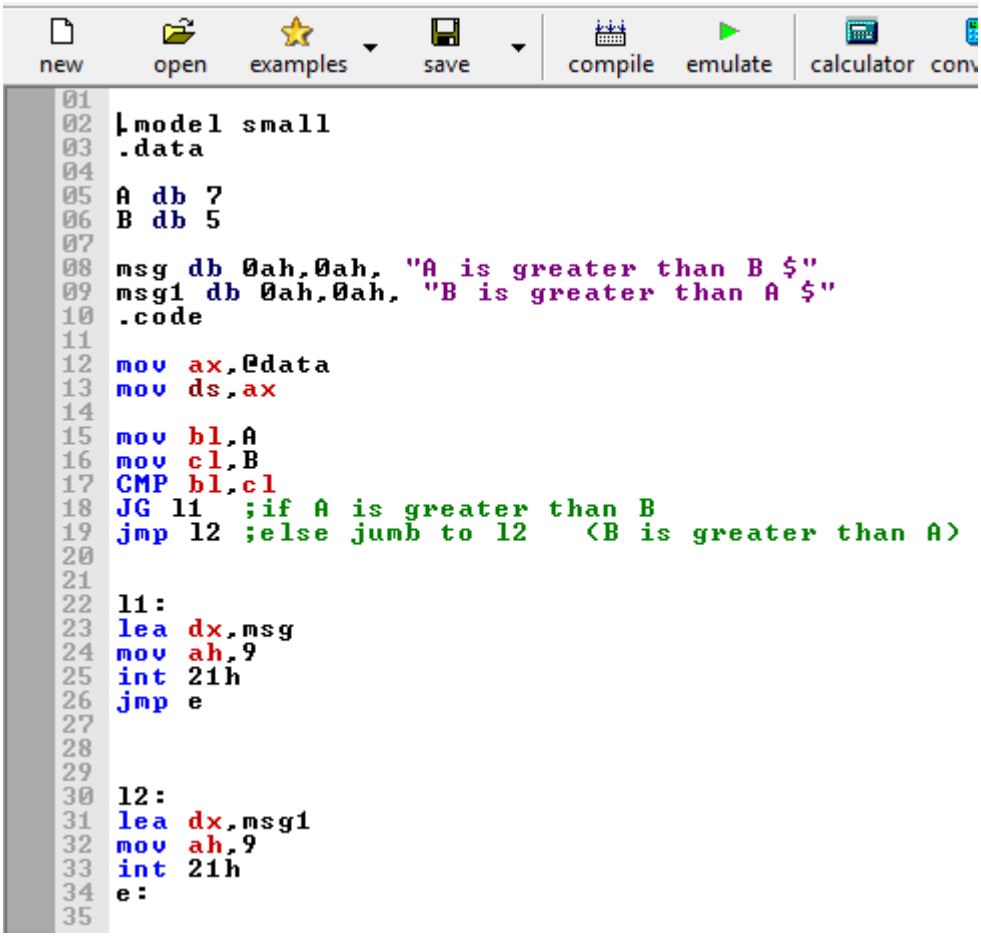

```
        cmp     ; is AL >
        ja      ;
        cmp     ; no: is BL >
        jbe     ; no: skip next
L1      mov     ; set X to
next
```

.

3. Walkthrough Task

Write an assembly language that compares the values A and B and prints a message if the value in A is greater than B, less than B.

Solution:



The above program shows the Comparison between two numbers

4. Procedure& Tools

Procedure and tools installation discussed in first lab (Lab01).

4.1 Tools

- Installing Microprocessor Intel 8086 emulator.

5. Practice Tasks

This section will provide more practice exercises which you need to finish during the lab. You need to finish the tasks in the required time. When you finish them, put these tasks in the following folder:
\\fs\assignments$\

5.1 Practice Task 1 [Expected time = 15mins]

Write an assembly language program that allow user to input one-digit number and determine if it is even or odd.

5.2 Practice Task 2 [Expected time = 15mins]

Write an assembly code that output a letter grade for 10 numbered grades according to the following table:

Numbered Grade	Letter Grade
90-100	A
80-90	B
70-80	C
60-70	D
0-59	F
Other	Not Valid

5.3 Out comes

After completing this lab, student will be able to compile and run basic Assembly arithmetic operation with their associative properties.

6. Evaluation Task (Unseen) [Expected time = 30mins for tasks]

The lab instructor will give you unseen task depending upon the progress of the class.

7. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 3: Evaluation of the Lab

Sr. No.	Task No	Description	Marks
1	4	Problem Modeling	20
2	6	Procedures and Tools	10
3	7	Practice tasks and Testing	35
4	8	Evaluation Tasks (Unseen)	20
5		Comments	5
6		Good Programming Practices	10

8. Further Reading

This section provides the references to further polish your skills.

8.1 Slides

The slides and reading material can be accessed from the folder of the class instructor available at \\fs\lectures$\