Assignment-SubLC3 Virtual Machine-Description

Version: Refer to the version in the file name

Contents

1 Problem to Solve

Develop a virtual machine (VM) / simulator for the SubLC3 instruction set. Call this virtual machine SubLC3VM.

The SubLC3 programming language is described in Appendix A.

SubLC3VM consists of the following two phases:

1.1 Load Phase

Load the SubLC3 program, which is saved on an external file called “mySubLC3”, into the memory. Use an array of size MAX_MEMORY_SIZE = 500 to store the instructions.

1.2 Fetch-execute-cycle Phase

Execute the program using the following steps:

```
initialize the program counter
repeat
    fetch the instruction pointed by the counter
    increment the counter
    decode the instruction
    execute the instruction
end when operation = HALT
```

2 Input

1. Test the virtual machine thoroughly. You may start testing your program by using the sample SubLC3 program given in Appendix A. Name the input file “mySubLC3.txt”. A file can contain one large SubLC3 program consisting of different sections. The whole program ends with one HALT statement.
2. Store the input file in the same location as that of your program file.
3. In your program, open the input file using the path "**src\\vmPackage\\mySubLC3.txt**"

3 Output

1. Display your name and class information, a line of asterisks (*), the SubLC3 source file, a line of asterisks (*), and finally the result of the SubLC3 execution.
2. The output must be well formatted and readable.

4 Naming of the Program Modules

Use the following names for your program modules:

Module	Name
Java Project	VmProject
Package	vmPackage
Class	VirtualMachine
Input file	mySubLC3.txt

5 What to turn in

a. Attach your .java program(s)

b. Attach your input file, mySubLC3.txt

c. Include the result of the execution of your program in the comment section of D2L, as described below.

d. To keep the format of the output intact when you insert it in the D2L comment section, you may use the following steps in Eclipse:

   i. Run-> Run Configurations -> Common -> Output file (check it) -> File System (specify where you want to store it and the name you want for the text file).

   ii. Open the file using the Microsoft Word.

   iii. Copy the contents and paste them in the D2L comment section.

Please do not zip your files.

Appendix A – SubLC3 Programming Language

6 Introduction to SubLC3

The SubLC3 instructions are divided into three different categories: operation, data movement, and control.

Each instruction except for the comment line (which starts with a semicolon) and a label line (which is an identifier) has the following format:

Operator Operands

Hence, a line which does not start with either a semicolon or an operator is a label.

7 Operation Instructions

7.1 ADD Destination Source1 Source2

ADD accepts two integer values, Source1 and Source2, and stores their sum in the *Destination* variable, Destination.

Each Source can be either a variable or an integer constant.

Examples:

Code:
; Add 3 to the value of num1 and store the result in sum

```
ADD sum num1 3
```

Code:
; Add the values of num1 and num2 and store the result in sum

```
ADD sum num1 num2
```

Code:
; Add 40 and 6 and store the result in sum

```
ADD sum 40 6
```

7.2 SUB Destination Source1 Source2

SUB accepts two integer values, Source1 and Source2, and stores the result of (Source1 – Source2) in the *Destination* variable, Destination.
Each Source can be either a variable or an integer constant.

7.3 MUL Destination Source1 Source2

MUL accepts two integer values, Source1 and Source2, and stores the result of (Source1 * Source2) in the *Destination* variable, Destination.
Each Source can be either a variable or an integer constant.

7.4 DIV Destination Source1 Source2

DIV accepts two integer values, Source1 and Source2, and stores the result of (Source1 / Source2) in the *Destination* variable, Destination.
Each Source can be either a variable or an integer constant.

7.5 IN Variable

Inputs an integer value and stores it in Variable.

Example:

Code:
; Input an integer value and store it in the variable num

```
IN num
```

7.6 OUT Value

Display Value.

Value can be either an integer variable or a string of characters enclosed in quotation marks (" ")

Example:

Code:
; Display the value of the variable sum

```
OUT sum
```

If the value of sum is 1234, the output will be

```
1234
```

Code:
; Display “Hello World!”

```
OUT "Hello World!"
```

---

8 Data Movement Instructions

8.1 STO Destination Source

The STO instruction stores the value of Source in Destination variable.

Source can be either a variable or an integer constant.

Example:

Code:
; Store 23 in age

```
STO age 23
```

Code:
; Store the value of num2 in num1

```
STO num1 num2
```

---

9 Control Instructions

9.1 BRn Variable Label

If the value of Variable is negative, jump to Label.

Example:

Code:
; If the value of age is negative, jump to sumLabel

```
BRn age sumLabel
; Other statements
sumLabel
; Other statements
```

9.2 BRz Variable Label

If the value of Variable is zero, jump to Label.
Example:

Code:
; If the value of age is zero, jump to sumLabel

```
BRz age sumLabel
; Other statements
sumLabel
; Other statements
```

9.3 BRp Variable Label

If the value of Variable is positive, jump to Label.

Example:

Code:
; If the value of age is positive, jump to sumLabel

```
BRp age sumLabel
; Other statements
sumLabel
; Other statements
```

9.4 BRzp Variable Label

If the value of Variable is zero or positive, jump to Label.

Example:

Code:
; If the value of age is zero or positive, jump to sumLabel

```
BRzp age sumLabel
; Other statements
sumLabel
; Other statements
```

9.5 BRzn Variable Label

If the value of Variable is zero or negative, jump to Label.

Example:

Code:
; If the value of age is zero or negative, jump to sumLabel

```
BRzn age sumLabel
; Other statements
sumLabel
; Other statements
```

9.6 JMP Label

Jump to Label.

Example:

Code:
; Jump to sumLabel

```
JMP sumLabel
; Other statements
sumLabel
; Other statements
```

9.7 HALT

End the program execution.

---

10 Comments, Labels and Identifiers

Comment:

Semicolon (;) at the beginning of a line indicates a comment line, which will be ignored by SubLC3VM.

10.1 Identifier:

The name of an identifier starts with a letter, followed by a sequence of zero or more letters, digits or underscores.

An identifier may represent either a variable (of type integer) or a label.

10.2 Label:

Label is an identifier, which indicates a location within the SubLC3 program. It is used as a destination for a JMP or BRx instruction.

11 An Example Program in SubLC3

```
; Display a sequence of numbers in reverse order
;
; Initialize the values
STO increment 2
OUT "Enter the initial number:"
IN number
;
; Display the values
OUT "The values are:"
SUB tempNum number 1
loopStart
BRn tempNum loopEnd
OUT number
SUB number number increment
SUB tempNum number 1
JMP loopStart
loopEnd
;
; Test the last value of number
OUT "The last value of number is”
OUT number
BRzp number ifEnd
OUT "The last value of number is negative.”
JMP ifEnd
OUT "The last value of number is not negative.”
ifEnd
;
OUT "H A V E   A   N I C E   D A Y !"
HALT
```

12 The Equivalent Program in Java

```
 // Display a sequence of numbers in reverse order

Scanner scan = new Scanner(System.in);

// Initialize the values
int number;
int increment = 2;
System.out.println("Enter the initial number:");
number = scan.nextInt();

// Display the values
System.out.println("The values are:");
while (number >= 1){
    System.out.println(number);
    number = number - increment;
}

// Test the last value of number
System.out.println("The last value of number is");
System.out.println(number);
if (number < 0)
    System.out.println("The last value of number is negative.");
else
    System.out.println("The last value of number is not negative.");

System.out.println("H A V E  A  N I C E  D A Y !");
```

13 Sample execution of the program

```
Enter the initial number:
5
The values are:
5
3
1
The last value of number is
-1
The last value of number is negative.
H A V E  A  N I C E  D A Y !
```

14 Summary of SubLC3 Statements

Statement	Description
ADD Destination Source1	ADD accepts two integer values, Source1 and Source2, and stores their sum in the *Destination* variable, Destination. Each Source can be either a variable or an integer constant.
BRn Variable Label	If the value of Variable is negative, jump to Label.
BRp Variable Label	If the value of Variable is positive, jump to Label.
BRz Variable Label	If the value of Variable is negative, jump to Label.
BRzn Variable Label	If the value of Variable is zero or negative, jump to Label.
BRzp Variable Label	If the value of Variable is zero or positive, jump to Label.
Comment	Semicolon (;) at the beginning of a line indicates a comment line, which will be ignored by SubLC3VM.
DIV Destination Source1 Source2	DIV accepts two integer values, Source1 and Source2, and stores the result of (Source1 / Source2) in the *Destination* variable, Destination. Each Source can be either a variable or an integer constant.
HALT	End the program execution.
Identifier	The name of an identifier starts with a letter, followed by a sequence of zero or more letters, digits or underscores. An identifier may represent either a variable (of type integer) or a label.
IN Variable	Inputs an integer value and stores it in Variable.
JMP Label	Jump to Label.
Label	Label is an identifier, which indicates a location within the SubLC3 program. It is used as a destination for a JMP or BRx instruction.
MUL Destination Source1 Source2	MUL accepts two integer values, Source1 and Source2, and stores the result of (Source1 * Source2) in the *Destination* variable, Destination. Each Source can be either a variable or an integer constant.
OUT Value	Display Value. Value can be either an integer variable or a string of characters enclosed in quotation marks (" ")
STO Destination Source	The STO instruction stores the value of Source in Destination variable. Source can be either a variable or an integer constant.
SUB Destination Source1 Source2	SUB accepts two integer values, Source1 and Source2, and stores the result of (Source1 – Source2) in the *Destination* variable, Destination. Each Source can be either a variable or an integer constant.

15 Reference

These instructions are adapted from the LC3 instruction set:

http://www.lc3help.com/tutorials.htm?article=Basic_LC-3_Instructions/