**CS 390 – Week 6**

**Assignment 7**

**Matt Hartigan**

**15-June-2019**

1. **Give an informal semantic definition for a switch statement.**

Example switch statement:

switch (input\_num) {

case 1:

x = input\_num;

break;

case 2:

x = input\_num;

break;

default:

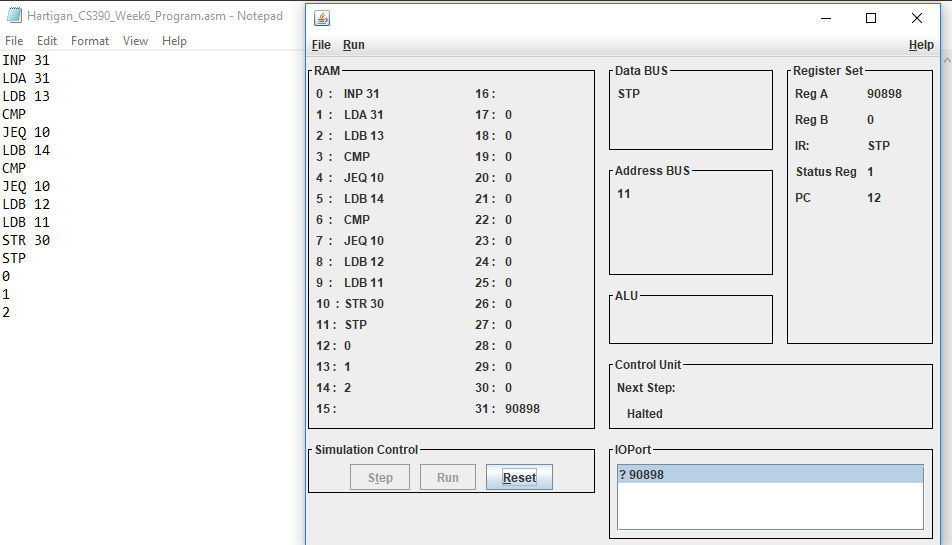
x = 0;

}

Informal semantic definition:

This switch statement takes an input parameter and attempts to match it to each of the ‘case’ values. If a match is found, a statement assigns the input value to the variable x. If no match is found, a statement assigns the default value of zero the variable x.

1. **Using the Model Assembler code from Topic 4, give an Operational Semantics definition for Java’s switch statement. Briefly explain why your definition provides the requested operational semantics.**



Operational semantics is used to define a program by capturing how it affects the machine that is running it. By converting the switch statement into the Modeler assembly language, we show how executing the switch statement will affect our machine. This is especially apparent if we step through each program instruction, observing how we take an input value, store it in location 31, then perform a series of comparisons before eventually making the assignment to location 30. By walking through the program, we see how the machine’s state is affected.

1. **Using the denotational semantics addition example given in this topic as a guide, look up in Wikipedia “Lambda Calculus” the Lambda Calculus expressions for true, false, and logical-And. Create the Lambda Calculus denotation (expression) for the following syntactic expression: [[ true && false ]]**

From Wikipedia:

TRUE := λ*x*.λ*y*.*x*

FALSE := λ*x*.λ*y*.*y*

AND := λ*p*.λ*q*.*p* *q* *p*

Thus, we perform the following check:

AND TRUE FALSE

(λ*p*.λ*q*.*p* *q* *p)* TRUE FALSE

TRUE FALSE TRUE

*(*λ*x*.λ*y*.*x)* FALSE TRUE

FALSE

Second sanity check:

AND TRUE TRUE

(λ*p*.λ*q*.*p* *q* *p)* TRUE TRUE

TRUE TRUE TRUE

*(*λ*x*.λ*y*.*x)* TRUE TRUE

TRUE

Therefore, we can define [[ true && false ]] using the Lambda Calculus as:

(λ*p*.λ*q*.*p* *q* *p)(*λ*x*.λ*y*.*x)(*λ*x*.λ*y*.*y)*

1. **Using axiomatic semantics, prove the statements below are not equivalent unless the two operands are equal. Again, be precise with a formal proof.**

Statements:

a = x – y

b = y – x

if { x = y } a = x – y { a = 0 } and { x = y } b = y – x { b = 0 } then a = b

1. **Java’s indexOf(int ch) returns the first occurrence of the character in the character sequence represented by corresponding String object, or -1 if the character doesn’t occur in the String. Using axiomatic semantics, give formal pre- and post- conditions for this indexOf method.**

Precondition: { (string length > 0) AND (ch is in input string) }

Postcondition { output integer >= 0 }