CS1100 - Lecture 2

Instructor : Jasine Babu

Teaching Assistants : Nikhila K N, Veena Prabhakaran

1 Introduction

In the last class, we have studied only instructions involving simple arithmetic operations, input and output operations, assigning values to variables etc. The arithmetic operators to be used with integers are $+, -, *, /$ and %. Here, $+, -$ have the usual interpretation, $*$ is the multiplication operator, % is the remainder (or mod) operator (e.g 10 % 3=1). Try to write a program to compute 10/3 to understand the precise meaning of / operator.

For all programs that we discussed so far, after one instruction is getting executed, the program counter was just getting increamented to the next instruction. Using the tools developed so far, even simple tasks like computing the maximum of two numbers does not seem possible. For these, we need to study some more operators and different ways of updating the program counter.

The comparison operators are $<, <=, >, >=, ==$ and $!=$. In C, the operator $=$ is used for assigning values and $==$ is used for checking equality of two expressions. Operator $!=$ is used for checking ineqality of two expressions. The result of comparisons are used to make decisions based on which the updation of program counter can be changed. By the term "control flow" of a program, we mean the details of how the program counter gets updated during the program execution.

One way of changing the sequential control flow of a program is by using an *if statement*.

In the program for finding the maximum of two numbers, if statements can be used among with comparison operators as follows.

```
int x, y, max
input x
input y
if (x>y)
{
   max <-- x
}
else
{
   max <-- y
}
output max
```

When the statement `if (x>y)` is executed, the current values of variables x and y are taken to the ALU and compared. When the condition evaluates to *true*, then it means

the instructions between the opening bracket { after the `if` instruction and its pairing closing bracket } are to be executed, but the instructions between the opening bracket { after the `else` and its pairing closing bracket } are to be skipped. When the condition evaluates to *false*, then it means the instructions between the opening bracket { after the `if` instruction and its pairing closing bracket } are to be skipped, but the instructions between the opening bracket { after the `else` and its pairing closing bracket } are to be executed. The program counter will be updated in one of these ways, depending on the result of the comparison.

The memory state diagram of the above program for the inputs x=10 and y=20 is shown below.

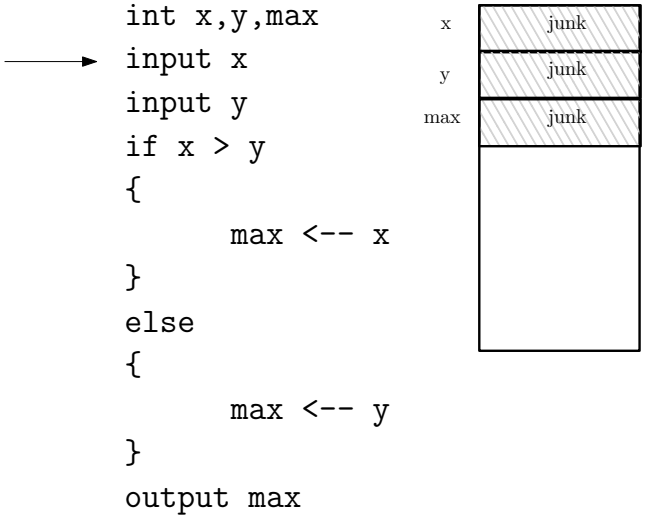

Figure 1: i

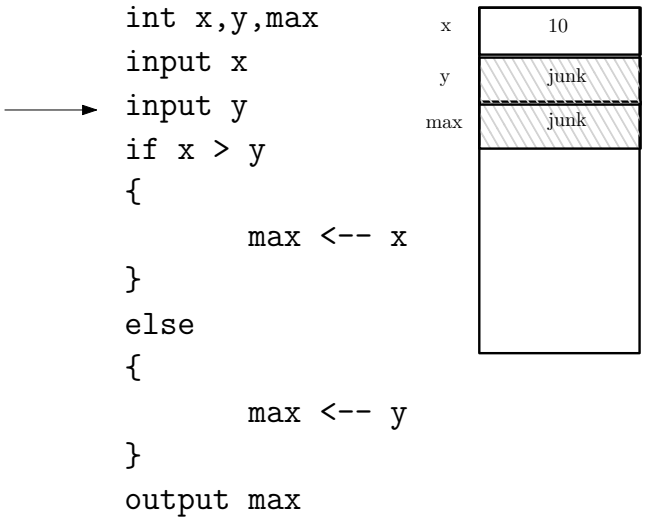

Figure 2: ii

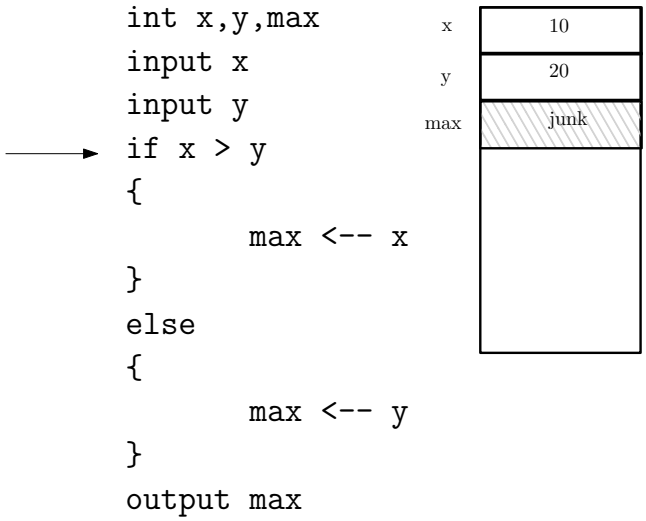

Figure 3: iii

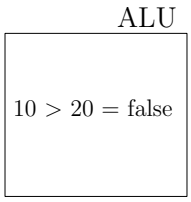

Figure 4: iv

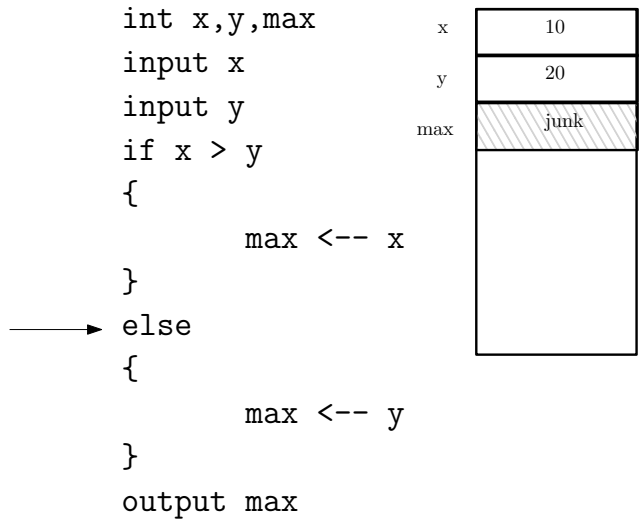

Figure 5: v

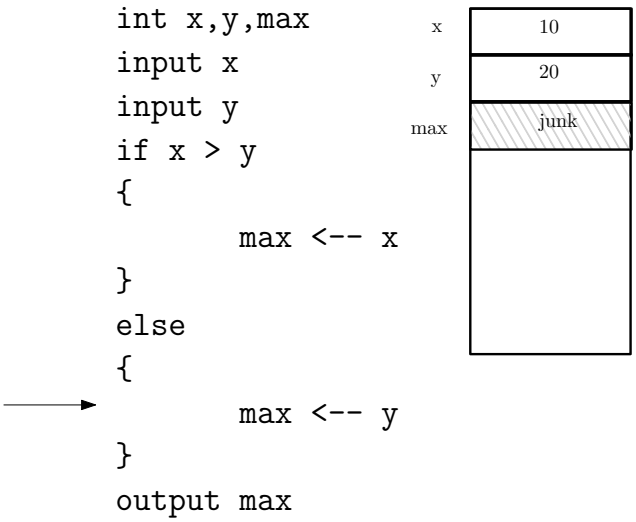

Figure 6: vi

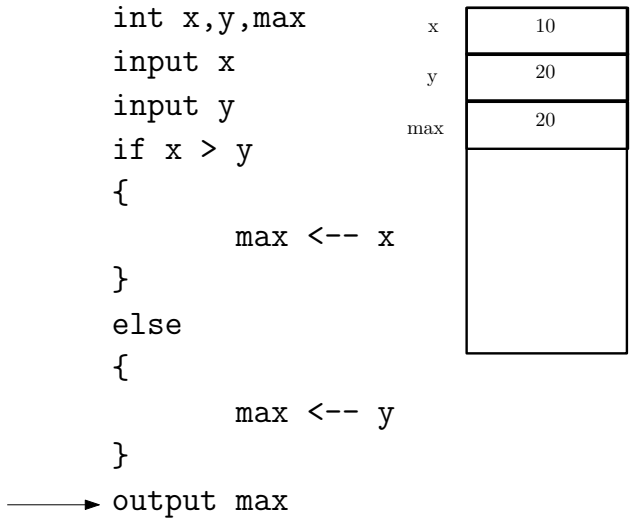

Figure 7: vii

The memory state diagram of the same program for the inputs x=10 and y=20 is shown below.

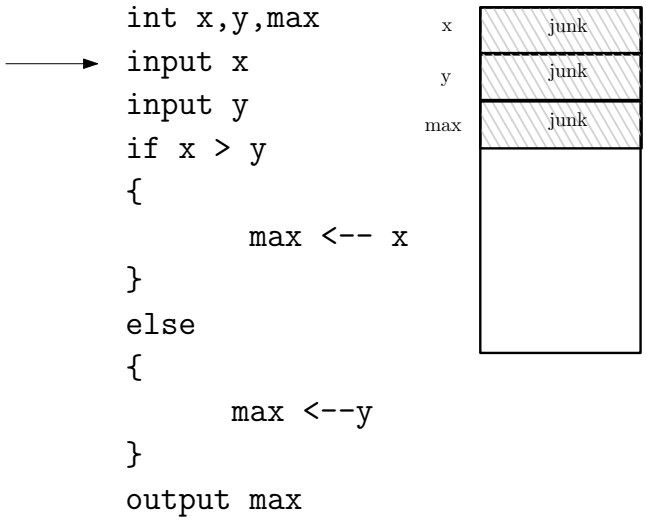

Figure 8: i

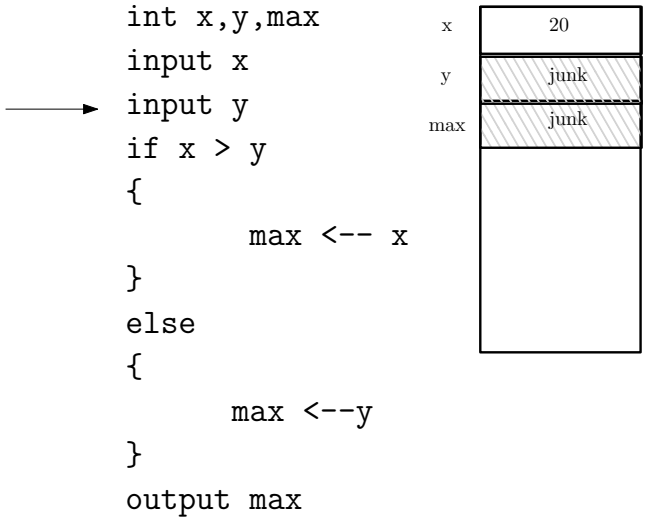

Figure 9: ii

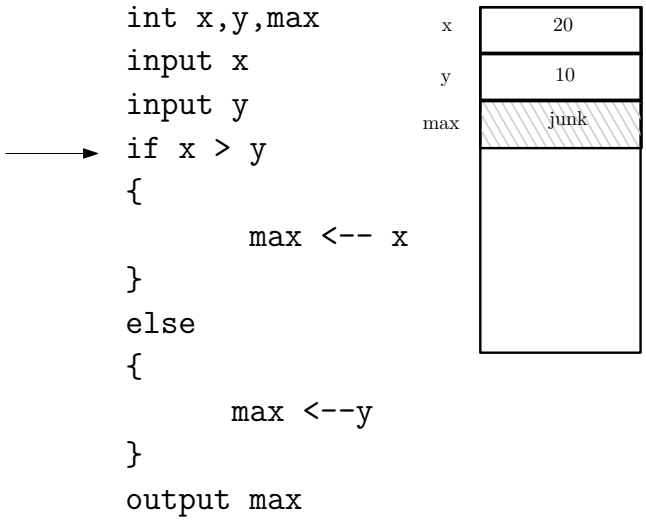

Figure 10: iii

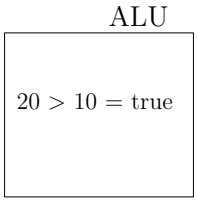

Figure 11: iv

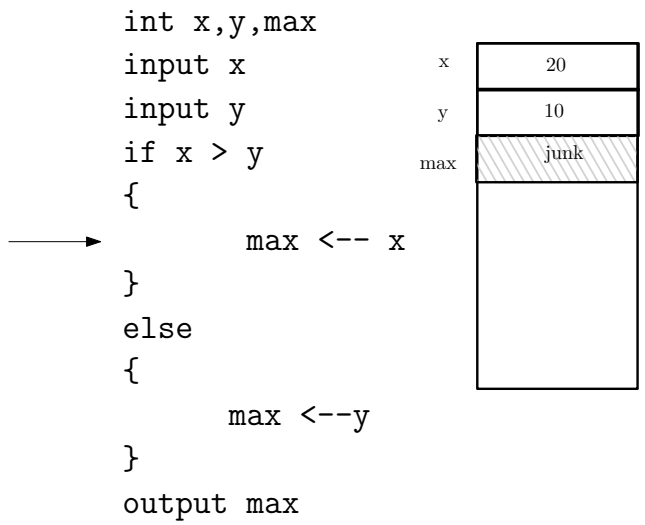

Figure 12: v

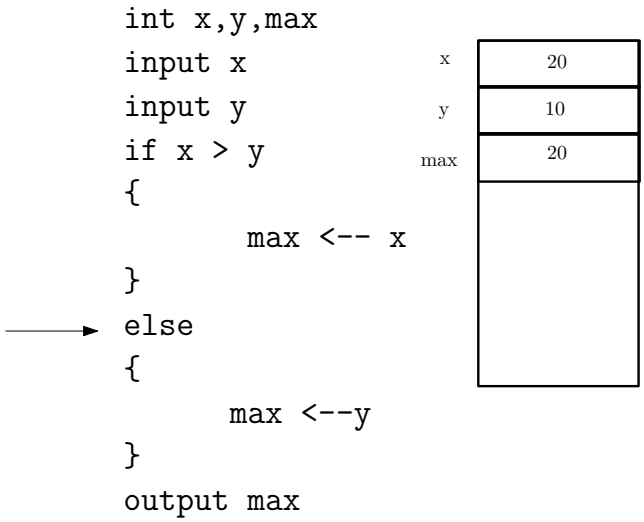

Figure 13: vi

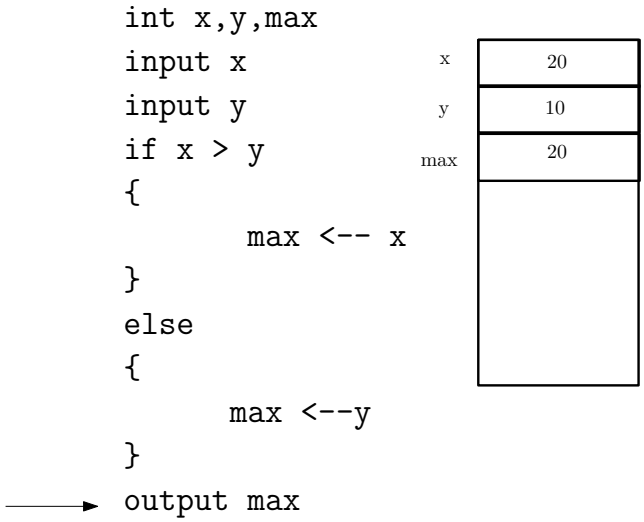

Figure 14: vii

A program to find the largest of three numbers is given below.

```
int x, y, z, max
input x
input y
input z
if x > y
    max <-- x
else
    max <-- y
if z > max
    max <-- z
output max
```

The memory state diagram of this program for the inputs x=10 , y=20 and z=30 is given below.

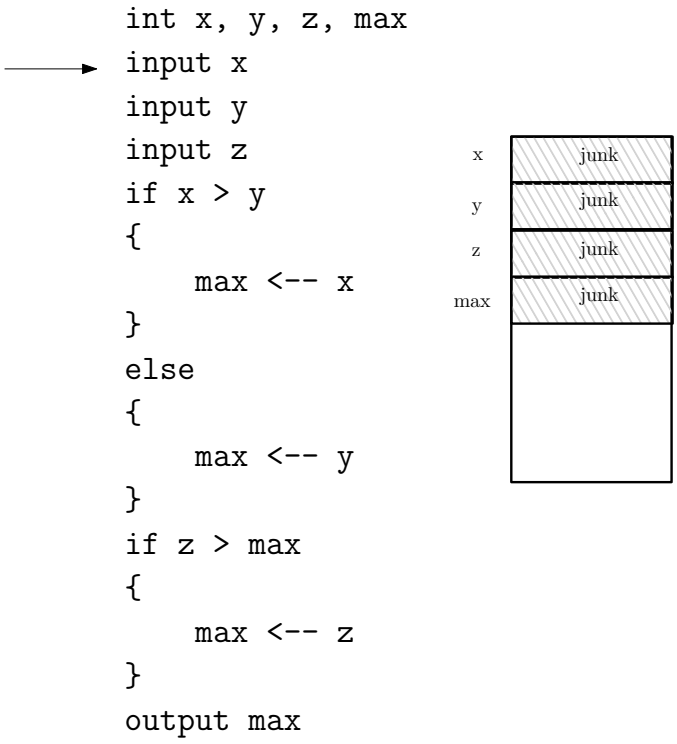

Figure 15: i

```
int x, y, z, max
input x
input y
input z
if x > y
{
    max <-- x
}
else
{
    max <-- y
}
if z > max
{
    max <-- z
}
output max
```

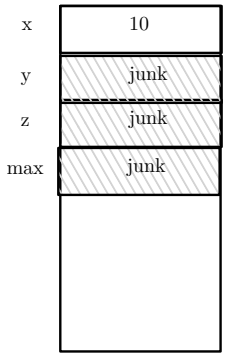

Figure 16: ii

```
int x, y, z, max
input x
input y
input z
if x > y
{
    max <-- x
}
else
{
    max <-- y
}
if z > max
{
    max <-- z
}
output max
```

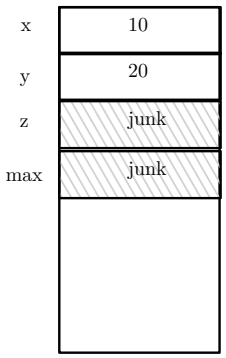

Figure 17: iii

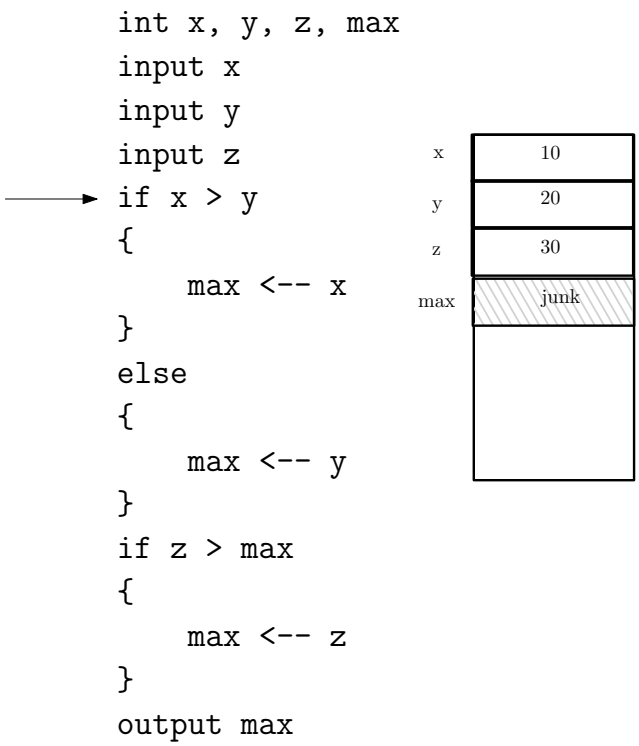

Figure 18: iv

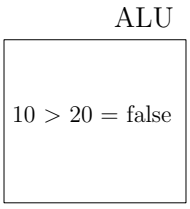

Figure 19: v

```
int x, y, z, max
input x
input y
input z
if x > y
{
    max <-- x
}
else
{
    max <-- y
}
if z > max
{
    max <-- z
}
output max
```

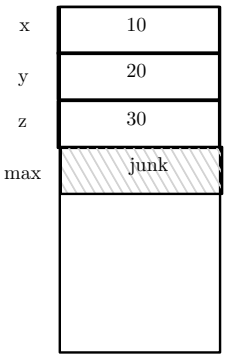

Figure 20: vi

```
int x, y, z, max
input x
input y
input z
if x > y
{
    max <-- x
}
else
{
    max <-- y
}
if z > max
{
    max <-- z
}
output max
```

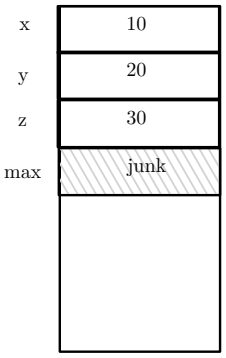

Figure 21: vii

```
int x, y, z, max
input x
input y
input z
if x > y
{
    max <-- x
}
else
{
    max <-- y
}
if z > max
{
    max <-- z
}
output max
```

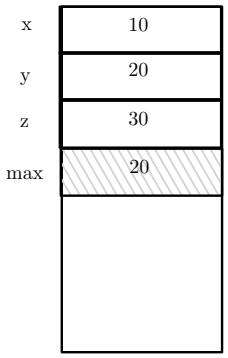

Figure 22: viii

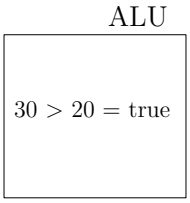

Figure 23: ix

```
int x, y, z, max
input x
input y
input z
if x > y
{
    max <-- x
}
else
{
    max <-- y
}
if z > max
{
    max <-- z
}
output max
```

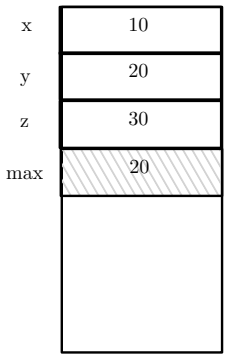

Figure 24: x

```
int x, y, z, max
input x
input y
input z
if x > y
{
    max <-- x
}
else
{
    max <-- y
}
if z > max
{
    max <-- z
}
output max
```

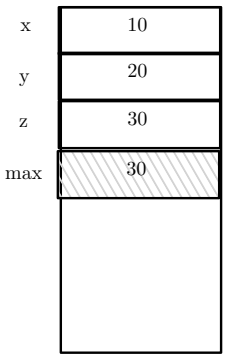

Figure 25: xi

Exercise:Draw the memory state diagram of the previous example for the inputs

1. x=30 , y=20 and z=10
2. x=10 , y=30 and z=20