COMPUTER ARCHITECTURE

ELEMENT 2

Name: Kaloyan Lalov

Student ID: 21368509

Computer Games Technology

University of West London

Table of Contents

Week 7 Exercises:

Describe the Von Neumann Model, its components, and the instruction processing cycle in 500 words.

The **Von Neumann Model** also known as the von Neumann model is based on the stored-program computer concept, where instruction data and program data are stored in the same memory.

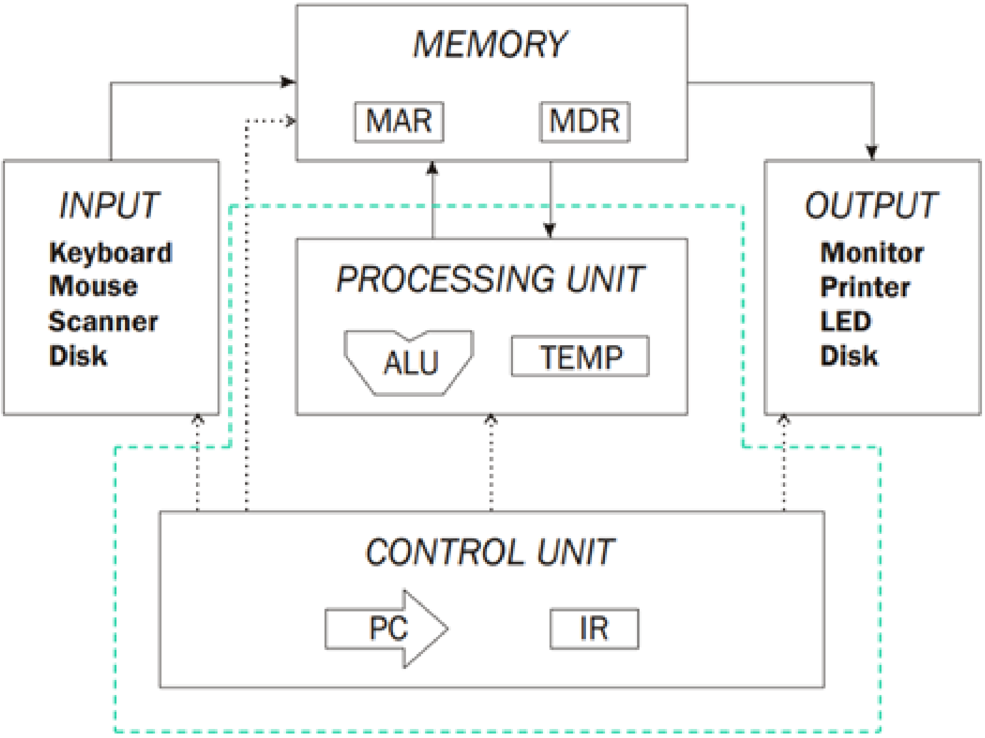
A stored-program digital computer keeps both program instructions and data in read-write, random-access memory.

It's a computer architecture based on a 1945 description by the mathematician and physicist John von Neumann and others in the First Draft of a report on the EDVAC. This design is still used in most computers produced today.

That document describes a design architecture for an electronic digital computer with:

- A processing unit that contains an arithmetic logic unit and processor registers
- A control unit that contains an instruction register and program counter
- Memory that stores data and instructions
- External mass storage
- Input and output mechanisms

Von Neumann Model



Components of a CPU

The two typical components of a CPU include the following:

- The *arithmetic logic unit (ALU)*, which performs arithmetic and logical operations.
- The *control unit (CU)*, which extracts instructions from memory and decodes and executes them, calling on the ALU when necessary.

Control Unit

A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions from microprograms and directs other units and models by providing control and timing signals. A CU component is considered the processor brain because it issues orders to just about everything and ensures correct instruction execution.

Instruction Register (IR) contains the current instruction.

The instruction is the fundamental unit of work. Specifies two things: opcode: operation to be performed operands: data/locations to be used for operation

Program Counter (PC) contains the address of the next instruction to be executed.

Memory

Memory is the area where the computer stores or remembers data. Memory provides the CPU with its instructions. There are different types of memory, and each one plays an important role in the running of a computer system. Memory is sometimes called primary memory.

Storage

There is a key difference between memory and storage. Programs are kept on a storage device and copied into the computer's memory before they are executed. Storage is also called **secondary storage**.

Input and Output Mechanism

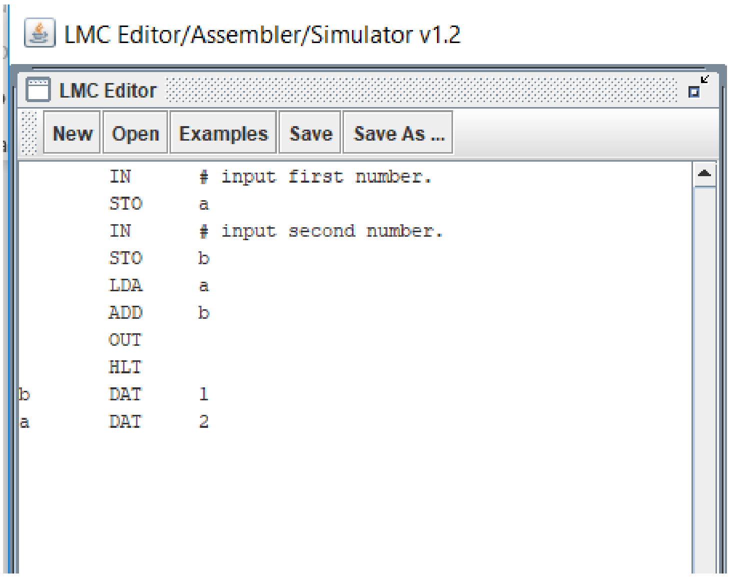
Each system has inputs, outputs, processes, constraints and mechanisms. A process is an action that transforms given inputs into outputs under certain constraints or restrictions and with the aid of some mechanisms. For example, the process of making coffee by a coffee maker can take inputs such as coffee, filter, water, and electricity, and result in outputs such as coffee, used filter, used coffee and grounds.

Week 8: Calculator - Part 1

In order to run assembly code, we use an application written in java called “LMC” (Little Man’s Computer)which is a very basic language with 9 instructions in total.

Firstly, we begin with a program that does simple addition between two values inputted by the user.

Addition



The command “IN” takes an input from the user.

The command “STO” stores the inputted value of the user

The command “ADD” adds the value of the input to the value that the program is already pointing to in the memory.

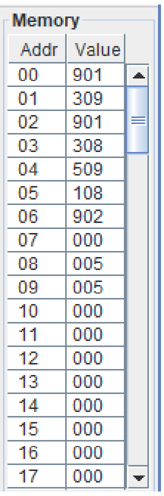
The command “OUT” is the equivalent of the command “print in high-level programming

The command HLT is used to exit the program.

The command “DAT” is equivalent of the command “int” in high-level programming. The format of the command is:

variable name(tab) DAT (tab) value

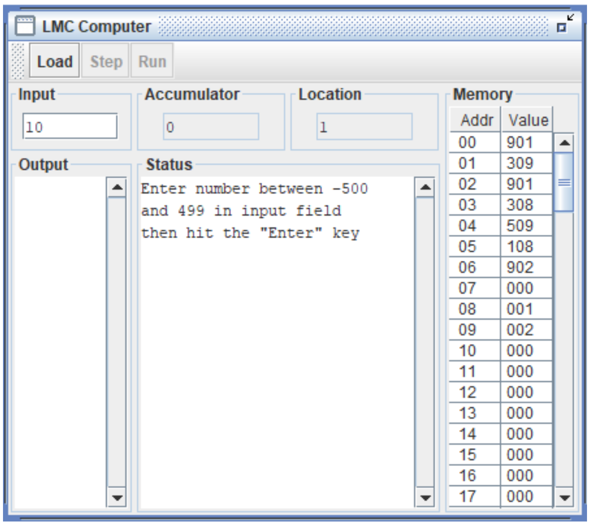
This is the memory.

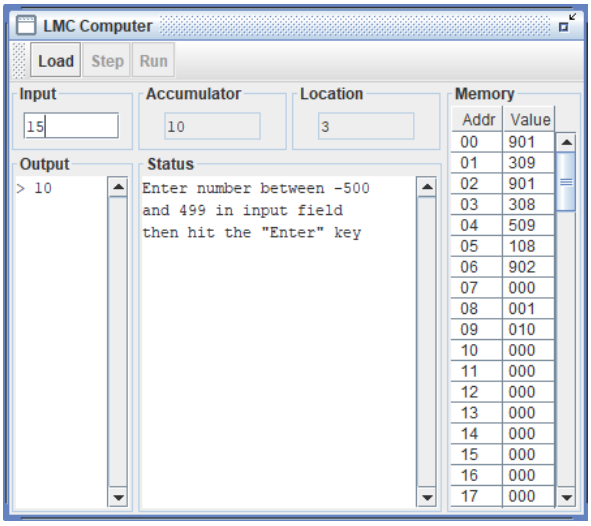


Memory

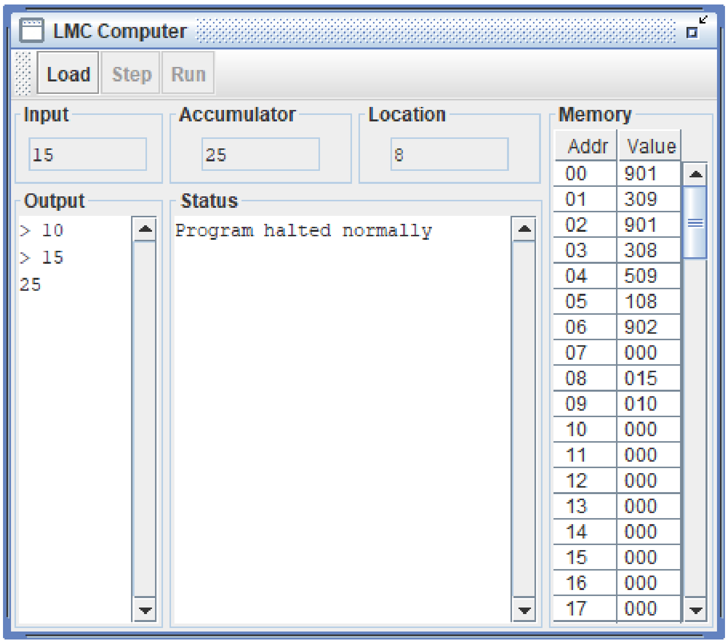
Addr	Value
00	901
01	309
02	901
03	308
04	509
05	108
06	902
07	000
08	005
09	005
10	000
11	000
12	000
13	000
14	000
15	000
16	000
17	000

This is how you input the first value “a” and after that the second value “b”



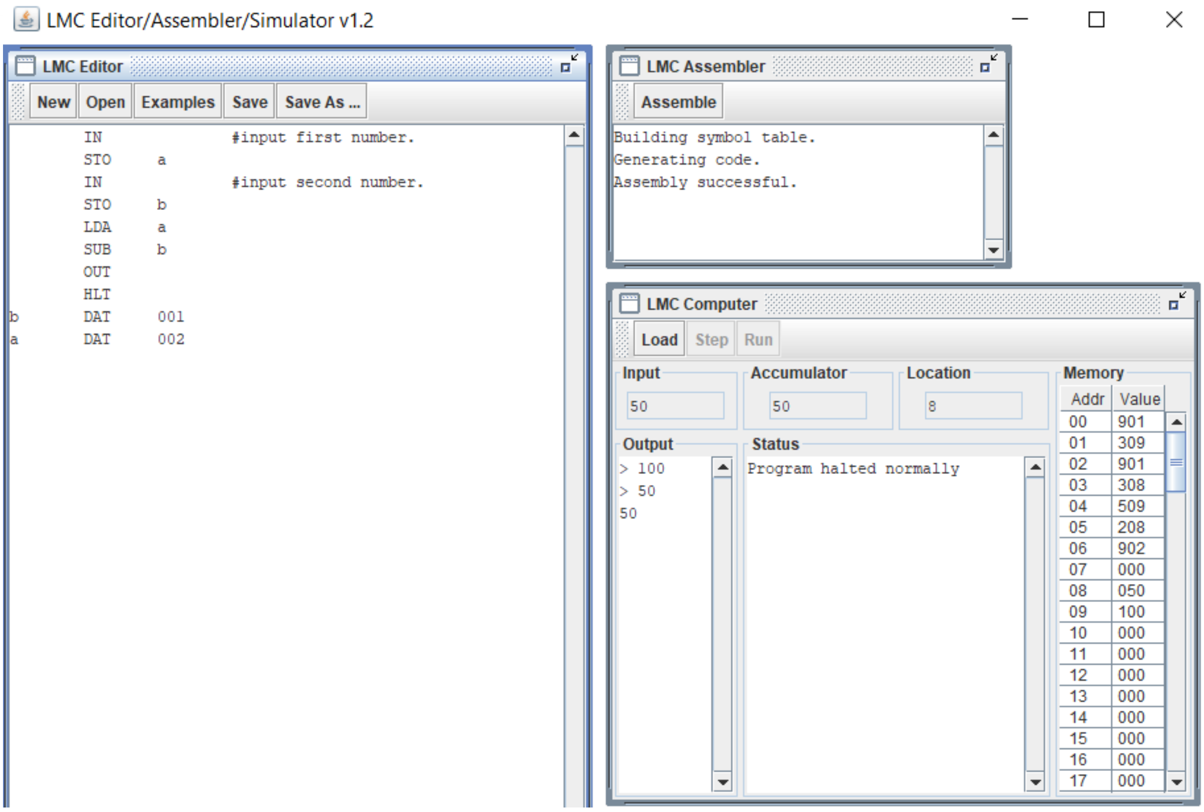


Output



The difference in this program is that in this program we use SUB instead of ADD, for subtraction.

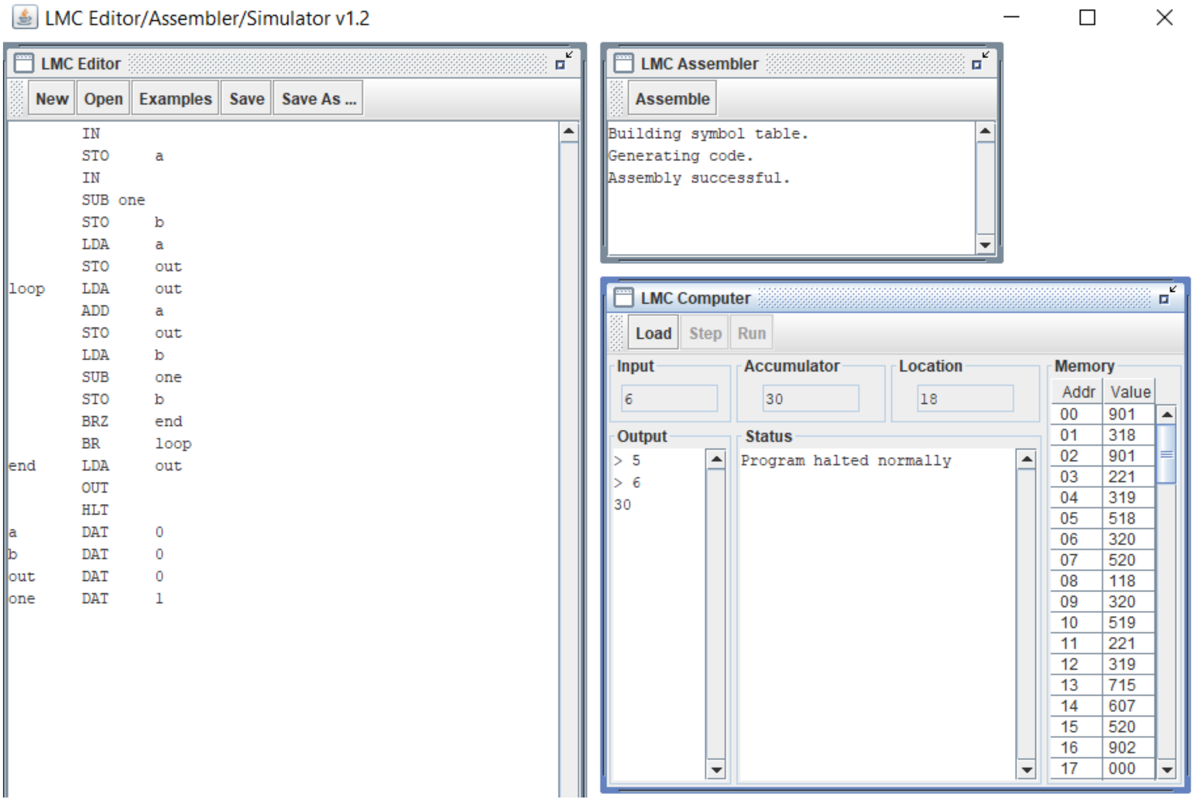
Subtraction



The first input is 100 and the second is 50 and the output is 50.

Week 9: Calculator - Part 2

Multiplication

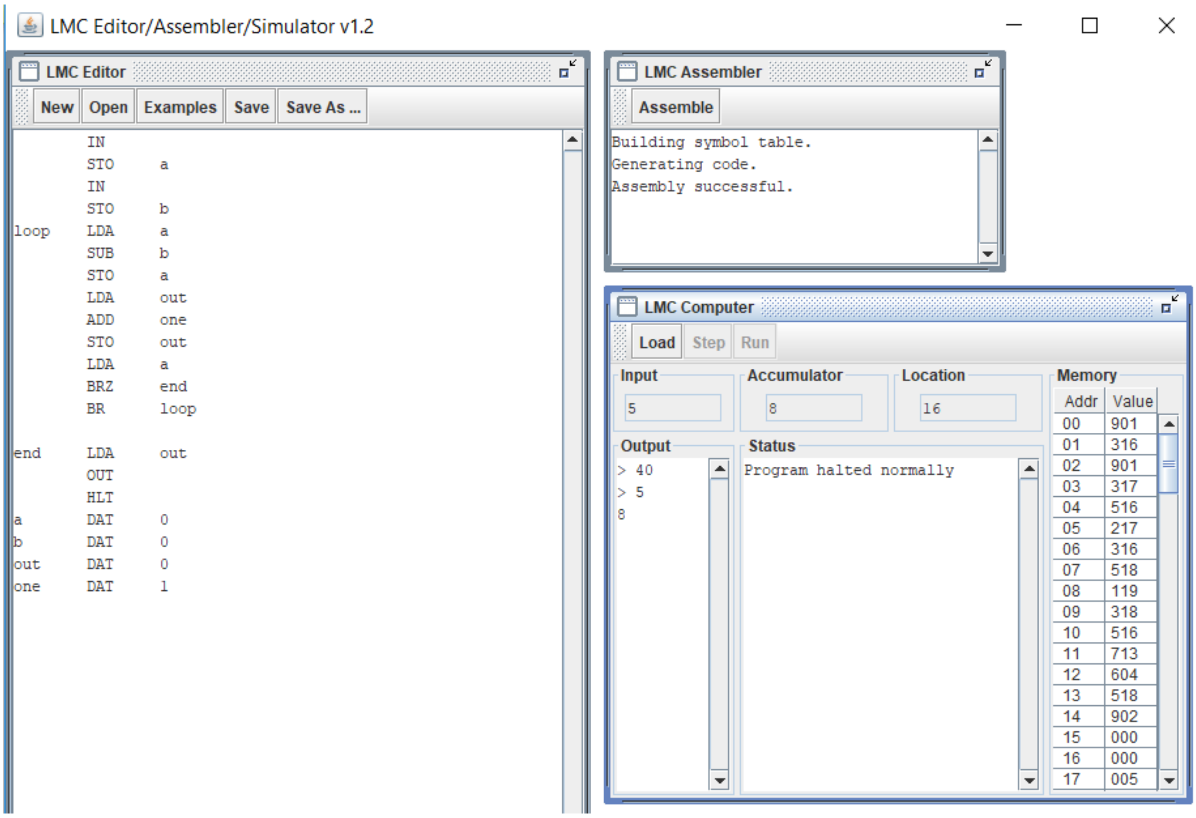


For this program, we ask for 2 values from the user. The second of which gets copied to another variable called ‘out’. ‘out’ is the sum of “a” being added to itself each iteration of the loop. “b” decrements by 1 every time “a” adds to itself, thus when “b” reaches 0, “a” will have added itself “b” amount of times. Finally, we check if “b” is 0, by BRZ (branch if zero) and sends off the program to continue from label ‘end’ where it loads the ‘out’ (sum of as), displays it, then quits the program. While ”b” isn’t 0, it will skip BRZ and go to ‘BR loop’ which GOTOs back to the loop label.

The division program is quite similar

Instead of adding "a" by itself, we decrement the dividend with the divisor in order to get the quotient. For that, we have an "out" variable that starts from 0, and counts up how many times it is necessary to subtract "b" from "a" in order to get 0.

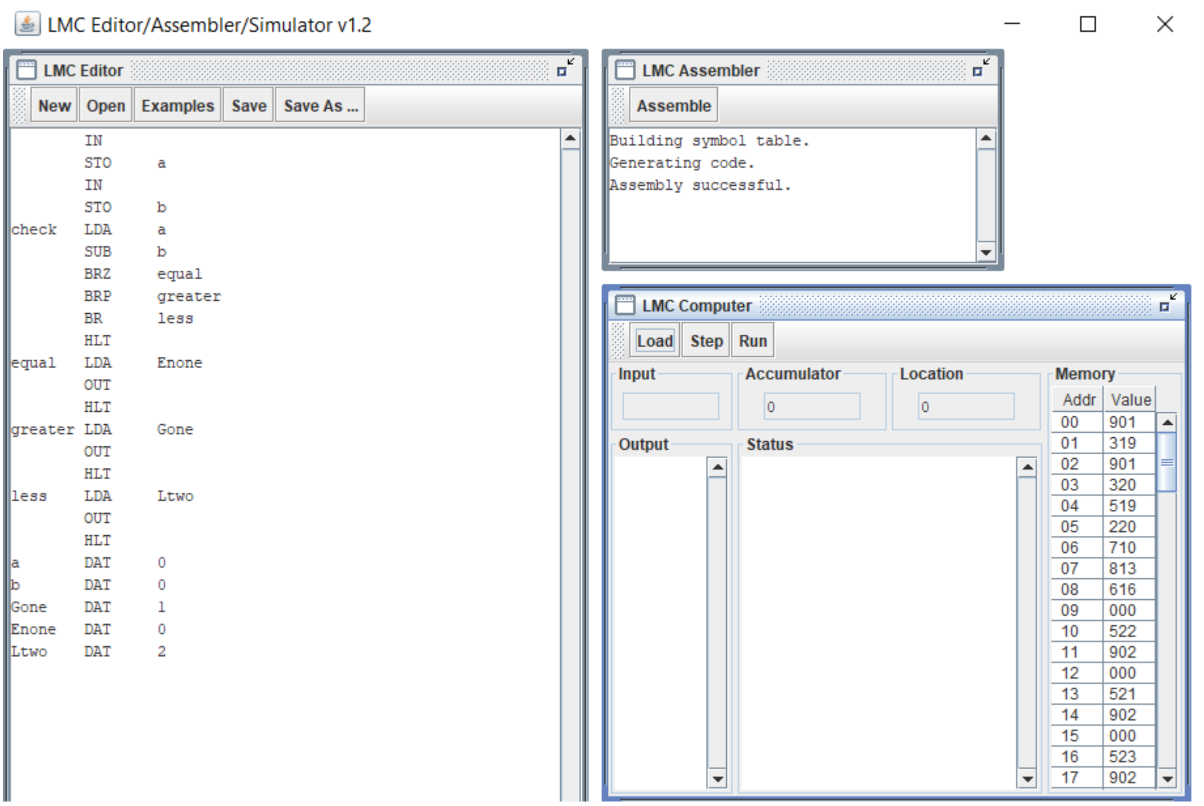
Division



COMPUTER ARCHITECTURE

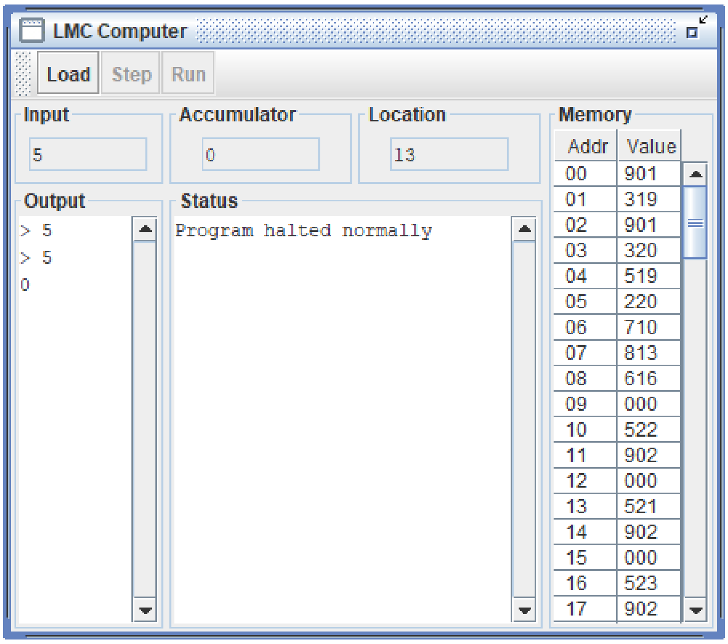
Week 10: Algorithms - Part 1

Equal, greater or less than



This program takes 2 inputs – “a” and “b”, and compares them by performing 3 checks.

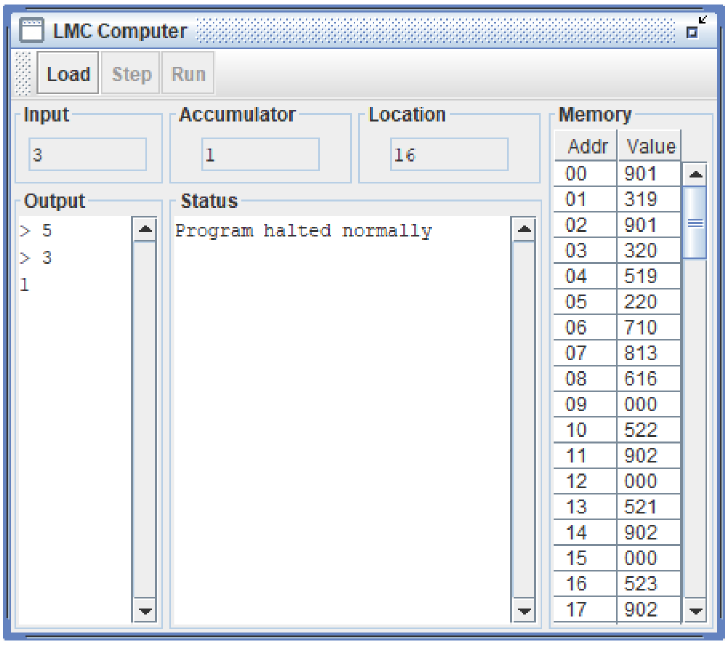
First outcome



The first one which is also what’s passed in the screenshot is the ‘equality’ check, which subtracts to numbers and if the outcome is 0, then they’re equal (BRZ). It then proceeds to go to the ‘equal’ label where it prints the result.

Note: for this program, the three possible outputs are: 0 for equal, 1 for greater and 2 for less. They all have respective labels for each of them (“Gone” used for greater one, “Enone” is for equal none, “Ltwo” used for less that two.

Second outcome



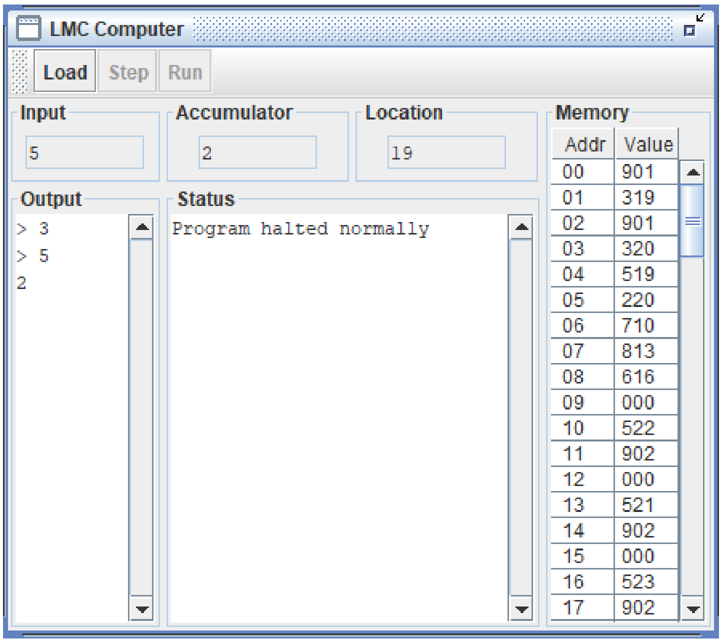
In this example, we enter the numbers 5 and 3, and we get the output code of ‘1’ (greater) because 5 is greater than 3

It performs the BRP (branch if positive) meaning if after the subtraction from “a” and “b”, the outcome is still positive, then we can be sure that a > b.

Last situation a < b

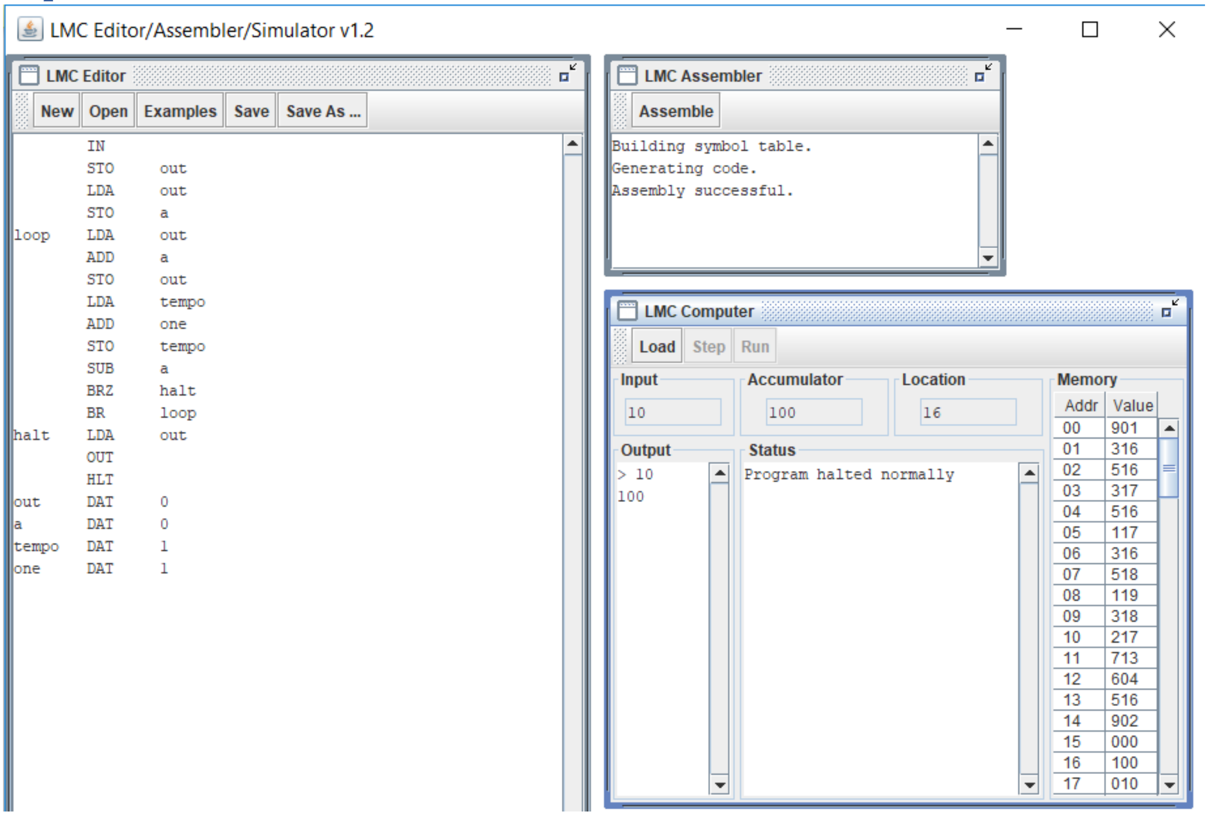
In this situation we use the BR (branch) command.

Third outcome



The next exercise is squaring the input of the user.

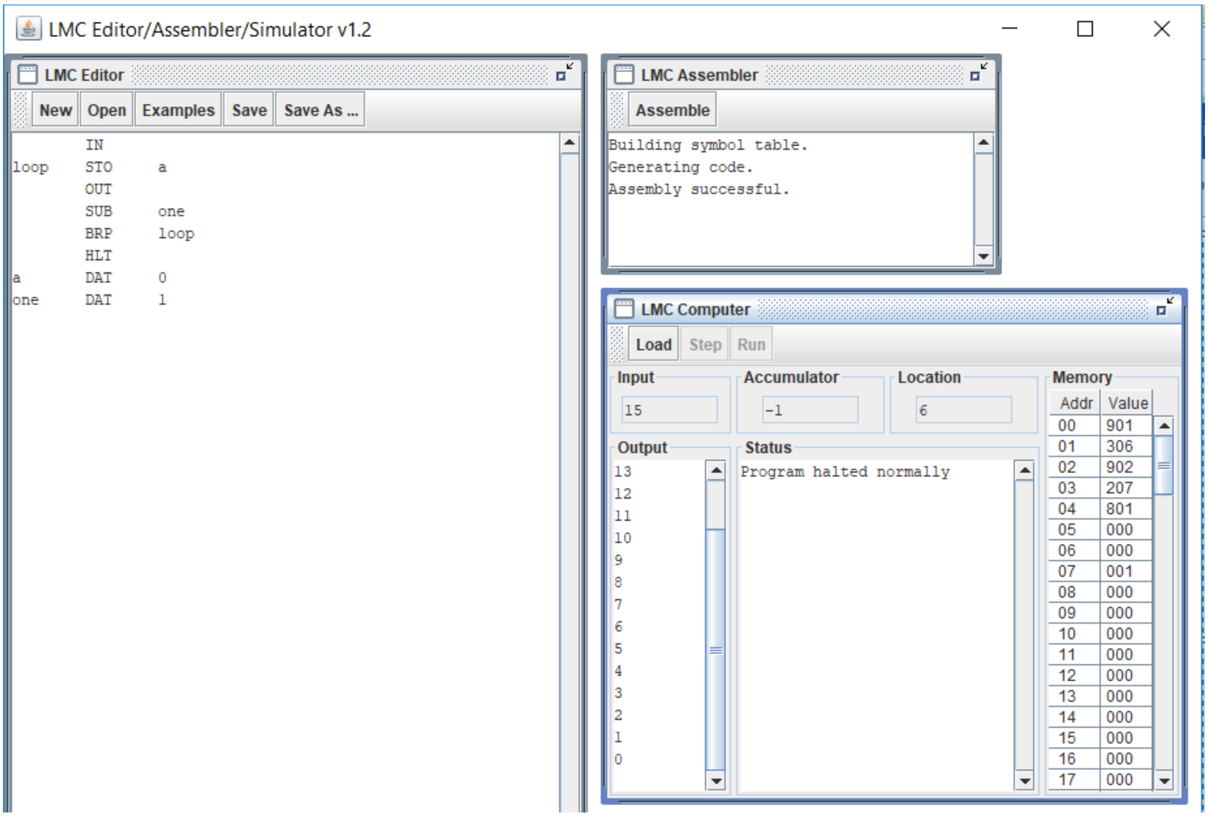
Square



Similarly to multiplication, we add the inputted number to itself by its value amount of times.

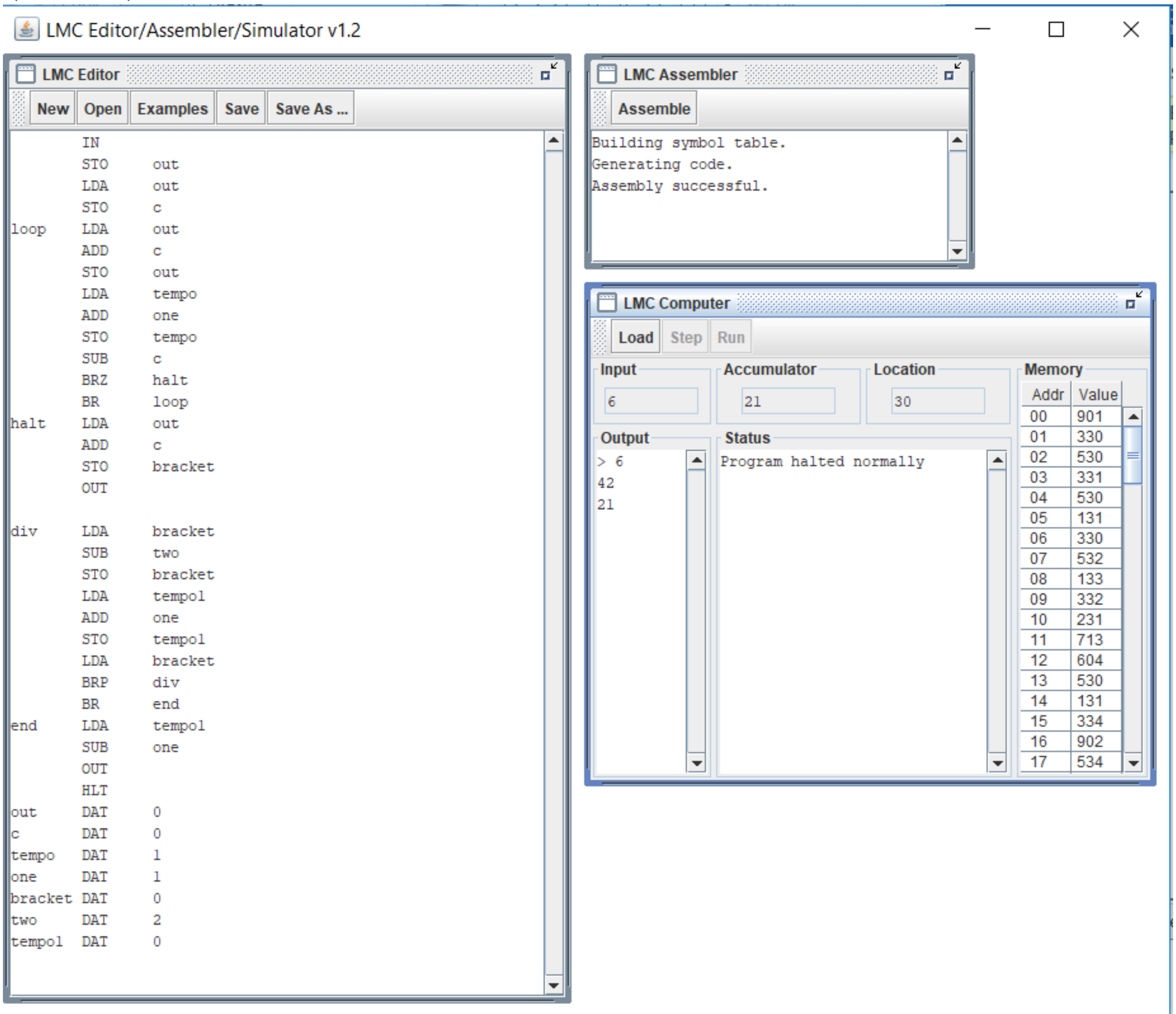
For example, we add 10 to itself 10 times. In order for the program to know when to stop adding, we use a ‘tempo’ counter that increments by 1. However, LMC doesn’t support equal to number comparison, only equal to 0. (BRZ). For that reason, we subtract the input from “tempo”, then wait as “tempo” slowly progresses up to 0, one by one, where it branches off to the halt label and prints the sum of input.

Countdown



This program asks for input once, then subtracts 1 to the current number whilst printing the current value until it reaches zero, where it stops.

$(n^2 + n) / 2$

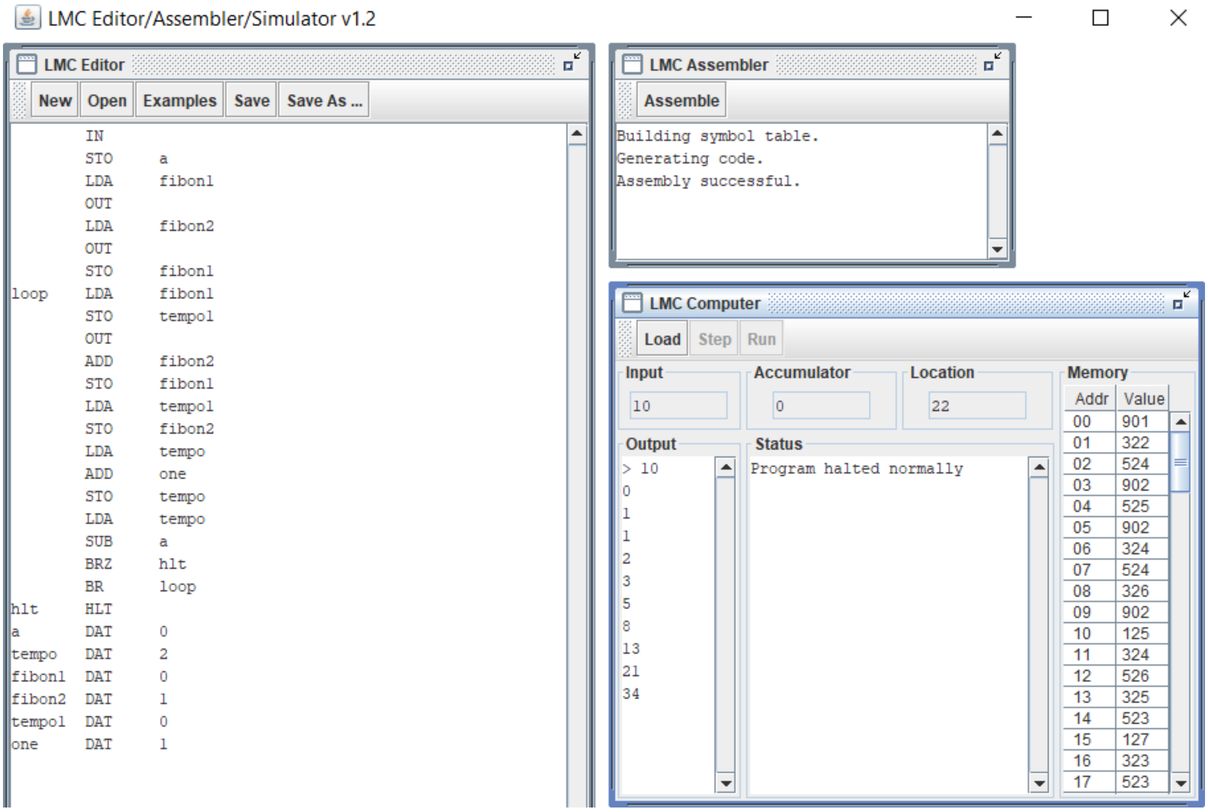


This program uses 2 different temporary variables which are essential for each loop. I will explain which action happens one by one. Firstly, “6^2” can be accomplished by the exact same method shown in the square example.

We add the number we want to square to itself its value amount of times. Next up, by performing simple addition (“+6”) and lastly, in order to divide by 2, we subtract by the number itself.

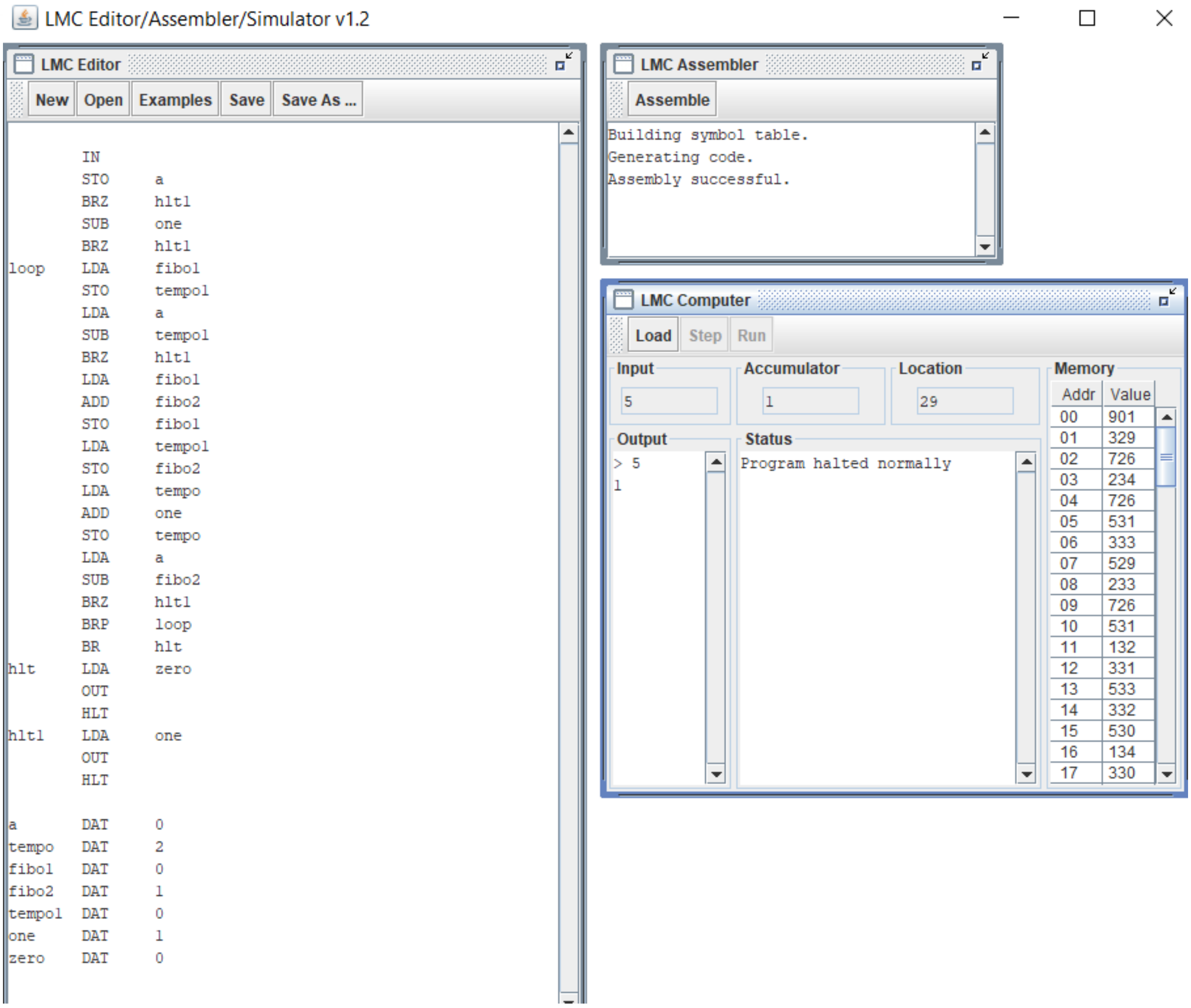
Week 11: Algorithms - Part 2

Fibonacci sequence



This program runs manual checks to see if the inputted value is either 1 or 0. If so, it displays the outcomes, and goes to the labels where it stops. Then it takes the previous number and adds it to the current one, while overwriting the previous one every time and then displaying it. It loads the “tempo” counter that starts from 2 (because of the exceptions) and then adds 1 every loop, subtracts the initial value from user input, to check if it's equal to zero; if it has reached the maximum desired value.

Check if a number is a Fibonacci number.



If the output is 1, the number you have inputted is a Fibonacci number, whereas if the output is 0, it isn’t a Fibonacci number.