This weeks assignment. You need to write a program that calculates Fibonacci numbers in our assembler language. To do that, you will have to log in to `rein001.utsc.utoronto.ca` and update your git repository. That's very simple using the following commands.

```
cd PSCB57
git pull
Acknowledge the merge. Simple save and exit the
   vi editor (using ESC :wq)
cd assignment_02
ls
cat INSTRUCTIONS.txt
```

Code 17: How to access this weeks assignment.

We are interested in assempler because it is extremely close to what actually happens on a computer. When we start using python next, you will see many high level instructions. These get translated into the most basic instructions for you, by something called a compiler, runtime or interpreter. Think of it as a translater between different languages.

The following figure illustrates how a program entered by a programmer is translated into machinecode.

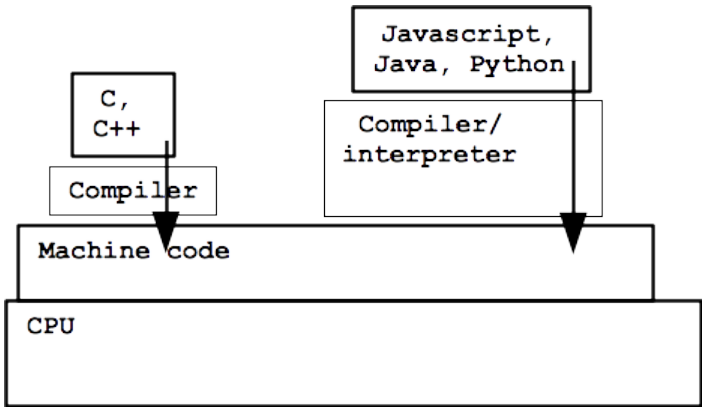

Figure 3: Software language stack. Source: `http://web.stanford.edu/class/cs101/`.

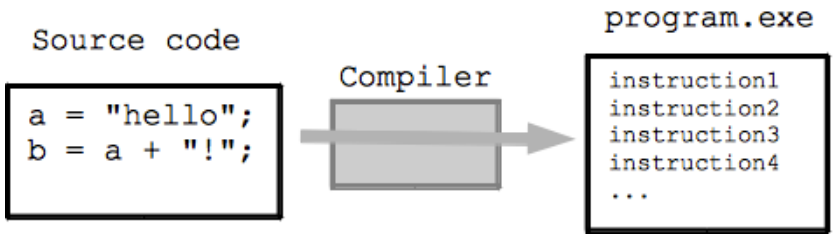

Figure 4: Compiler. Source: `http://web.stanford.edu/class/cs101/`.

Let start by revisiting the assembler instructions from last week. Notice that the instructions all have a very similar form. They have at most 4 parts:

```
COMMAND PARAMETER PARAMETER PARAMETER
```

Some of the commands only have one parameter, so the others could simply be set to zero.

Every command together with any possible combination of paramerers gives something called a machinecode, a single number that tells the computer what to do in one step.

Here are all the commands that we'll really need in our assembler language. I got rid of the COPY and SUB commands. It turns out, we can do everything without them.

Command	Argument 1	Argument 2	Argyment 3	Result
`SET`	s0	r1		r1 := s0 (The first argument is interpreted as a number, rather than a register address)
`PRINT`	r0			Print value of r0
`ADD`	r0	r1	r2	r2 := r1 + r2
`IF`	r0	r1		Execute next command only if r0 > r1
`JUMP`	r0			Jump r0 commands forwards (backwards if negative)

To write and run an assembler program you have to write down the commands in the right order. If you had taken this course 40 years ago, you'd be taking our your punch cards and start punching. We will write them in a text file. You should by now know how to edit a textfile on the linux system.

Every time you want to run your program, you need to save it first. Then, you run the interpreter and tell it to read in your text file. It will go through the text file, line by line, and execute exactly the command that you gave it.

Let's have a demo of the simplest possible example program. A program that outputs '1'.

```
SET 1 r0
PRINT r0
```

Code 18: Assembler program that outputs '1'.

Let's add two numbers.

```
SET 1 r0
SET 2 r1
ADD r0 r1 r2
PRINT r2
```

Code 19: Assembler program that adds two numbers and outputs the result.

Let's output all numbers from 0 to 9.

```
SET 0 r0
PRINT r0
SET 1 r0
PRINT r0
SET 2 r0
PRINT r0
SET 3 r0
PRINT r0
SET 4 r0
PRINT r0
SET 5 r0
PRINT r0
SET 6 r0
PRINT r0
SET 7 r0
PRINT r0
SET 8 r0
PRINT r0
SET 9 r0
PRINT r0
```

Code 20: Assembler program that outputs numbers from '0' to '9'.

This was rather dull. Let's find a smarter way.

```
SET 0 r0
SET 1 r2
SET -2 r3
PRINT r0
ADD r0 r2 r0
JUMP r3
```

Code 21: Assembler program that outputs all numbers.

The above program runs for ever. To make it stop we need a conditional statement. And now the correct way of only outputting the first 10 numbers

```
SET 0 r0
SET 9 r1
SET 1 r2
SET -3 r3
PRINT r0
ADD r0 r2 r0
IF r1 r0
JUMP r3
```

Code 22: Assembler program that outputs all numbers.

Our assembler language is very basic. For example, it has no commands for multiplication. How can we solve this?

I'll give you ten minutes. Come up with a assembler program that multiplies the two numbers in register 0 and 1 and store the result in register 2. You can assume that the two numbers are integers.

```
SET 3 r0
SET 4 r1
SET 0 r2
SET 0 r3
SET 1 r4
SET -4 r5
SET 3 r6
ADD r4 r3 r3
IF r3 r1
JUMP r6
ADD r0 r2 r2
JUMP r5
PRINT r2
```

Code 23: Assembler program multiplying two numbers.

Last but not least, let's write an algorithm for dividing two numbers, a and b. Let's assume that a and b are positive integers and that the result $c = a/b$ is also an integer.

```
SET 42  r0
SET 7   r1
SET -1  r2
SET 1   r3
SET 3   r7
SET -9  r8
SET -4  r9
 ADD r3 r2 r2
 SET 0 r4
 SET 0 r5
  ADD r3 r5 r5
  IF r5 r2
  JUMP r7
  ADD r1 r4 r4
  JUMP r9
 IF r0 r4
 JUMP r8
PRINT r2
```

Code 24: Assembler program dividing two integers.

Some other possible programs, I could ask you for:

- Square a given number
- Print out first N squares (1,2,4,8,16,...)
- Sutract a number from another number, only using additions.

End of lecture 2.