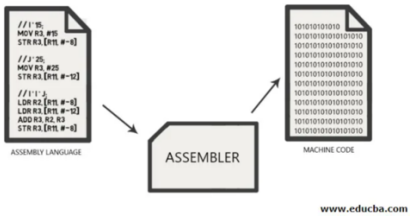
Assignment III: Writing ARM Assembly



DUE DATE: February 5, 2021 11:59pm

Purpose: The purpose of this assignment is to help you to learn the basic knowledge and skills in writing, testing, assembling and executing a simple ARM Assembly code.

Skills: This assignment will help you to practice the following skills that are essential to your success in this course and in CS discipline:

- Use basic add, load, store instructions in ARM assembly language
- Write, assemble and execute a low-level program
- Convert a simple C program into an ARM assembly language
- Create a `Makefile` to easily build an executable that might take many commands to create
- Use Raspberry Pi to assemble and execute programs written in ARM assembly language
- Use basic Unix commands
- Use a text editor in terminal to create and save files
- Remotely connect and work in another device on the same network

Knowledge: This assignment will help you to become familiar with the following important content knowledge in this discipline:

- Writing a program in a low-level programming language
- Understanding the fundamental difference between a low-level programming language (ARM assembly) and a high-level programming language (C)
- Understanding the execution of instructions at CPU level
- Understanding and using global variables
- Understanding data and instruction sections of an assembly code
- Generating a `Makefile` to speed up building process
- Working with Raspberry Pi
- SSH, SCP tools

This assignment has <u>two parts</u>.

PART-1 [50 points] Fibonacci number F6

In this part, you will use ARM assembly language to write a program to calculate and return the 6th Fibonacci number F_6

In mathematics, the **Fibonacci numbers**, commonly denoted F_n, form a sequence, called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is

$$F_0=0,\ F_1=1$$
$$F_n = F_{n-1} + F_{n-2}$$
$$\text{for } n > 1.$$

The beginning of the sequence is thus:

0, 1, 1, 2, 3, 5, 8, 13, 21, …

Your program will start with two values in data section: F0=0, and F1=1. You will write ARM assembly code using only what we learned in this week to calculate F_6.

Here are the requirements and limitations given:

1. Only use what we learned so far (ADD, MOV, LDR, STR, etc.)
2. Your data section should contain **F0** and **F1** labels.
3. Use only two registers in your code section: **R0** and **R1**.
4. No other registers are allowed in code section.
5. Create a `Makefile` to manage the build and clean tasks.
6. The program will be named **fib6.s**
7. The output of your program should be like this:

```
$ ./fib6
$ echo $?
8
```

PART-2 [50 points] Convert A C Program into ARM Assembly

In this part, you are asked to write an ARM assembly program **c2arm.s** to implement the same task as the following C program.

```
int a = 10;
int b = 5;
int c = 3;

int main(){
    a = b++ + c++;
    b = --c + a;
    c += a + b++;
    return a + b + c;
}
```

Hint: write and execute a C program.

Using a text editor, write the C code, and save the file with the extention ".c" (example: program.c)

Use `gcc` compilation tool to convert C code into an executable using the following command:

```
gcc -o program program.c
```

To run the executable, use the following command:

```
./program
```

How to submit:

You are asked to submit your work as a single zip file via CANVAS. Zip file will include the following two archive files for each part:

- SerceFatmaCS351_3.zip
  - Part1.zip
    - fib6.s
    - Makefile
  - Part2.zip
    - c2arm.s
    - Makefile

Please use the following file format while naming the zip file: LastNameFirstnameX_Y.zip where LastNameFirstname is your last name with the first letter in capital, followed by your first name with the first letter in capital; the X is the course code; the Y is the assignment #. (ex: SerceFatmaCS351_3.zip)

How to grade:

Part1 [50 points]

- (10pts) The data section.
- (20pts) The code section.
- (10pts) correctness.
- (10pts) `Makefile` should perform the build and clean tasks without any issue.

Part2 [50 points]

- (10pts) The data section.
- (20pts) The code section.
- (10pts) correctness.
- (10pts) `Makefile` should perform the build and clean tasks without any issue.