Name: Minh Binh Nguyen

Team: x87

Part 4 report

I set up my Raspberry and connect it to my computer (using macOS) by using the terminal.

A screenshot of a cell phone

Description automatically generated

Then I create a folder named Project (because I don’t want to store all of my projects in the default folder, it’ll get messy later as more projects are coming)

So I have to direct the terminal to the new folder by using the command “cd ./Project”

Next, I open a new file called “first.s”

A screenshot of a cell phone

Description automatically generated

Here is the file, I type in the program following the instruction, and save it.

A screenshot of a computer

Description automatically generated

@ first program

.section .data

.section .text

.globl \_start

\_start:

mov r1, #5 @ load r1 with 5

sub r1, r1, #1 @ subtract 1 from r1

add r1, r1, #4 @ add 4 to r1

mov r7, #1 @ Program Termination: exit syscall

svc #0 @ Program Termination: wake kernel

.end

I assemble the first.s file, then link it to get an executable file “first”. Then run it using “./first”

A screenshot of a cell phone

Description automatically generated

Answer the question:

I didn’t see any output. Because this program doesn’t actually produce any output.

Just like any other programming language, if you don’t produce any output (print to the console screen, or print to file), you won’t see any output.

Next, I follow the instructions to make the program be able to be debugged.

A screenshot of a cell phone

Description automatically generated

Here is the debugging step:

A screenshot of a cell phone

Description automatically generated

Notice the data that the registers hold right before running line 11 (before exiting the program)

Let me explain.

Before line 6, all the registers hold 0’s.

In line 6, we store number 5 to r1. Now r1 holds 5.

In line 7, we subtract 1 from r1. Now r1 holds 4.

In line 8, we add 4 to r1. Now r1 holds 8.

In line 10, we store number 1 to r7 to make the program termination. Now r7 holds 1.

Part 2:

First, I prepare the connection between my computer and Raspberry, then direct to the Project folder and open a new file called “arithmetic1.s”

A screenshot of a cell phone

Description automatically generated

Type in the program.

A screenshot of a cell phone

Description automatically generated

@ arithmetic1 program

@ team: x87

.section .data

.section .text

.globl \_start

\_start:

@ load A, B, C, D into registers

mov r1, #10 @ r1 is A

mov r2, #11 @ r2 is B

mov r3, #7 @ r3 is C

mov r4, #2 @ r4 is D

@ calculate (A + B) and store the result in r5

add r5, r1, r2 @ r5 = r1 + r2

@ calculate (C \* D) and store the result in r6

mul r6, r3, r4 @ r6 = r3 \* r4

@ calculate the expression and store the reults in r1, which is also A

sub r1, r5, r6 @ r1 = r5 - r6

mov r7, #1 @ Program Termination: exit syscall

svc #0 @ Program Termination: wake kernel

.end

Then I assemble and link the file.

A screenshot of a cell phone

Description automatically generated

Then I load it in gdb, set up some breakpoints at line 14, 16, 18, 20 to see how my program works.

A screenshot of text

Description automatically generated

And here is the data that the registers hold at those breakpoints.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Explanation:

Before running line 14, I assign the given values to the registers accordingly.

Register r1 holds 10 (A)

Register r2 holds 11 (B)

Register r3 holds 7 (C)

Register r4 holds 2 (D)

The others hold 0’s

In line 14, I add r1 and r2, then assign the value to r5. Now r5 holds 21.

In line 16, I multiply r3 and r4, then assign the value to r6. Now r6 holds 14

In line 18, I subtract r6 from r5, then assign the value to r1 (A). Now r1 holds 7

In line 20, I assign number 1 to r7 to make the program termination. Now r7 holds 1.

And this is the picture of what my Project folder looks like.

A screen shot of a computer

Description automatically generated