Lab 3: Printf & Scanf

Victor Yuan

Microcontrollers and Embedded Systems

February 15, 2017

Introduction:

The Goal of this lab was to write a program in assembly to add two integers and display the result.

Procedure:

Part 1: Directives

Prepare two prompts for the first and second number using the syntax:

msgX: .asciz “insert message here\n”

X for any variable

Then provide the result in the syntax:

Res: .asciz “insert message here… %d\n”

The %d is the place holder for the 10 digit number string representation.

Then provide the the address of the format string, so that scanf can be used to read in the input from the user use the syntax:

Str1: .asciz "%d"

At this point two more variables are needed in my case I used x and y any letter variable works use the syntax:

(Insert variable here): .word 0

Part 2: Main

In the main load prompt into a register then printf the contents of the register the user then inputs a variable. Using scanf pass the value of the user’s input into r1 which contains location of variable x. Repeat the prompt for the second variable use scanf to read the value into the location of variable y.

Once the variables have been initialized load r0 with location of variable x. Load r2 with the value of x by indirectly referring to r0 using brackets []. Once r2 has been loaded with the value of variable x load r0 with location of variable y reapt the same process by loading r3 with the value of Y.

Once r2 and r3 are loaded add them together and store the result in r1 this will be stored into str1. Then print the statement and test out the program.

Code:

Directives:

```
                .data
msg1:
                .asciz "Enter num1\n"
msg2:
                .asciz "Enter num2\n"

res:
                .asciz "num1 + num2 =%d\n"

str1:           .asciz "%d"

x:              .word 0
y:              .word 0
```

Main:

Prompt for number 1 + scanf, reading in values for x and storing them

```
                .text
.global         main
main:

                ldr r0, =msg1
                bl printf
                ldr r0, =str1
                ldr r1, =x
                bl scanf
```

Prompt for number 2 + scanf, reading in values for y and storing them

```
                ldr r0, =msg2
                bl printf
                ldr r0, =str1
                ldr r1, =y
                bl scanf
```

Loading in r0 with x referring to the value of x indirectly and storing in r2

```
               ldr r0, =x
               ldr r2, [r0]
```

Loading in r0 with y referring to the value of y indirectly and storing in r3

```
ldr r0, =y
ldr r3, [r0]
```

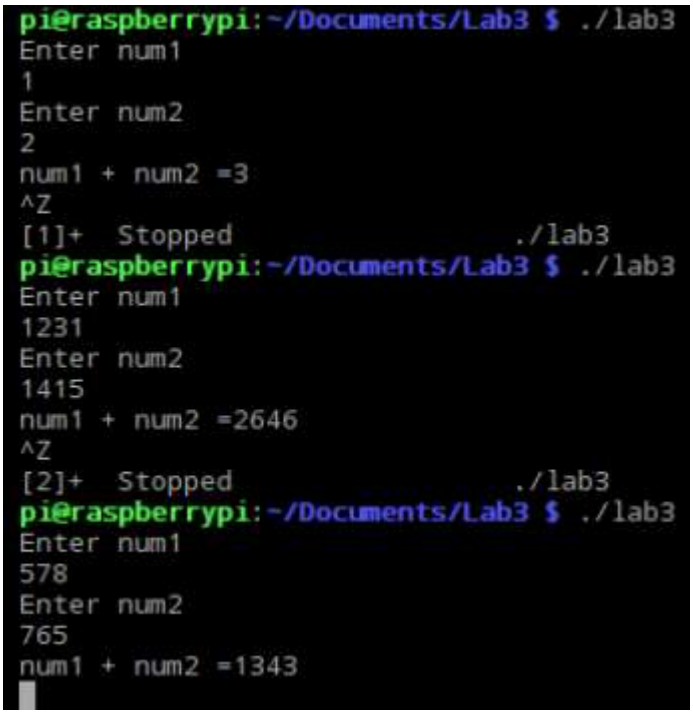
Addition of r2,r3 and storing result into r1 and printing the finale result, “res”.

```
ldr r0, =res
bl printf

stop:   B stop
```

B stop was added in to prevent a segmentation fault.

Results:



Shown above are 3 tests of the program, all 3 tests are successful.

Conclusion:

One thing I learned is that str1 automatically stores any value seen in r1, since whenever add was called and the storage register was not r1 the program would still compile however the result would be wrong as it would display a 0.

Extras:

Full code

```
                .data
msg1:
                .asciz "Enter num1\n"
msg2:
                .asciz "Enter num2\n"


res:
                .asciz "num1 + num2 =%d\n"


str1:           .asciz "%d"


x:              .word 0
y:              .word 0


                .text
.global         main
main:
                ldr r0, =msg1
        bl printf
                ldr r0, =str1
                ldr r1, =x
                bl scanf


                ldr r0, =msg2
                bl printf
                ldr r0, =str1
                ldr r1, =y
```

```
                bl scanf



                ldr r0, =x
                ldr r2, [r0]

                ldr r0, =y
                ldr r3, [r0]

                add r1,r2,r3



                ldr r0, =res
                bl printf

stop:           B stop
```

Compiling and running the code