Lab 2: Hello World & Debugger

Victor Yuan

Microcontrollers and Embedded Systems

February 8, 2017

Introduction:

The Goal of this lab was to familiarize the programmer with the tools on how to debug a code with the IDE, Code Blocks.

Procedure:

Part 1:

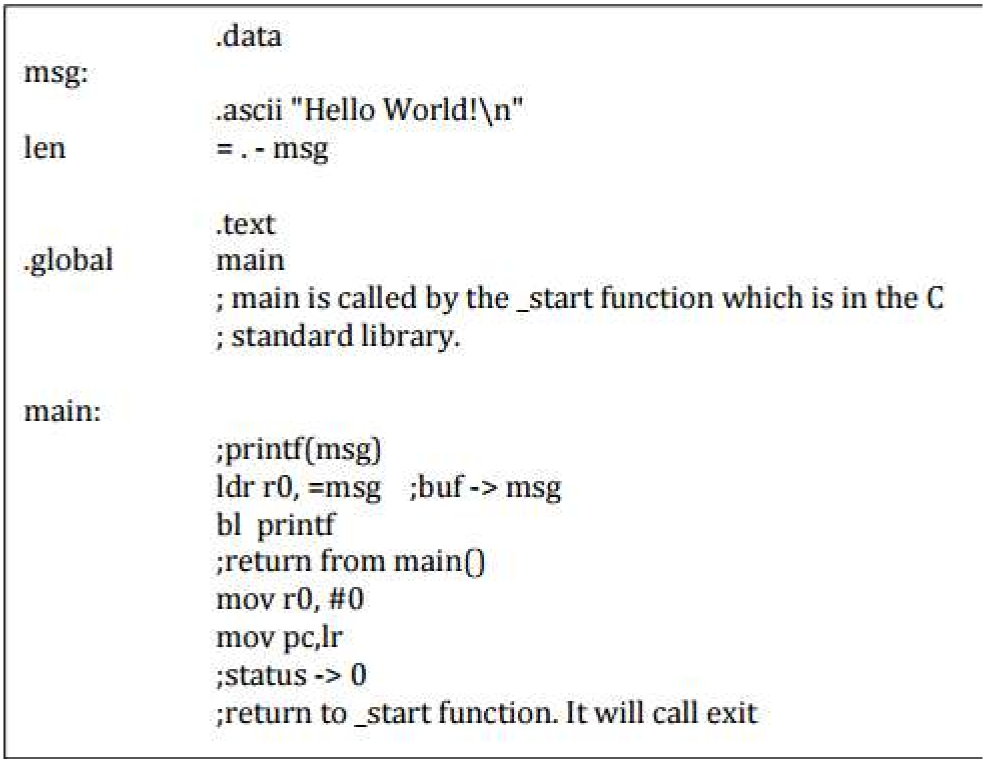
Type in the provided given code. Compile and run the code.

Part 2:

Open up Code Blocks. Create a project file import the code and run the debugging tool. Insert a break point. While pulling up the CPU Registers and Memory dump window.

Code:

Given code from Part 1, typed into a textile and compiled and ran:



```
                .data
msg:
                .ascii "Hello World!\n"
len             = . - msg

                .text
.global         main
                ; main is called by the _start function which is in the C
                ; standard library.

main:
                ;printf(msg)
                ldr r0, =msg   ;buf -> msg
                bl  printf
                ;return from main()
                mov r0, #0
                mov pc,lr
                ;status -> 0
                ;return to _start function. It will call exit
```

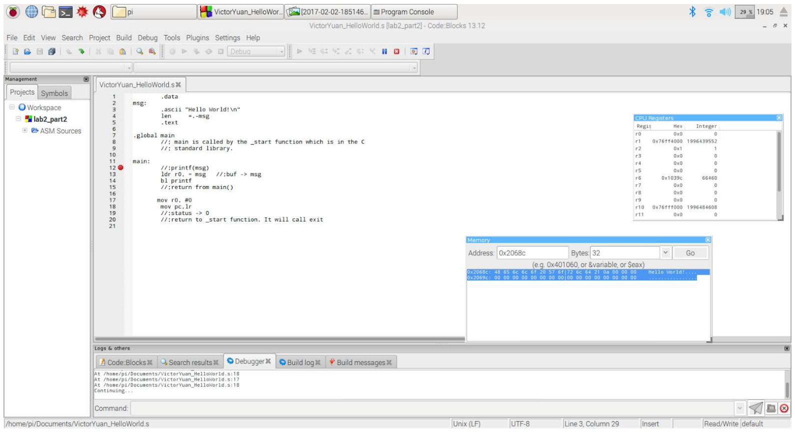
Results:

1. Monitor all changes made to register r0 and report the different values. What’s the significance of each value seen in r0?

   R0 started at 0x2068c20 moved to 0xd to 0x0 and stayed there for the rest of the program.

2. What is the starting address of msg (in Hex) in memory?

Program being debugged:



```
        .data
msg:
        .ascii "Hello World!\n"
        len     =.-msg
        .text

.global main
        //; main is called by the _start function which is in the C
        //; standard library.

main:
        //;printf(msg)
        ldr r0, = msg   //;buf -> msg
        bl printf
        //;return from main()

       mov r0, #0
        mov pc,lr
        //;status -> 0
        //;return to _start function. It will call exit
```

CPU Registers

Regi:	Hex	Integer
r0	0x0	0
r1	0x76ff4000	1996439552
r2	0x1	1
r3	0x0	0
r4	0x0	0
r5	0x0	0
r6	0x1039c	66460
r7	0x0	0
r8	0x0	0
r9	0x0	0
r10	0x76fff000	1996484608
r11	0x0	0

Memory

Address: 0x2068c Bytes: 32 Go

(e.g. 0x401060, or &variable, or $eax)

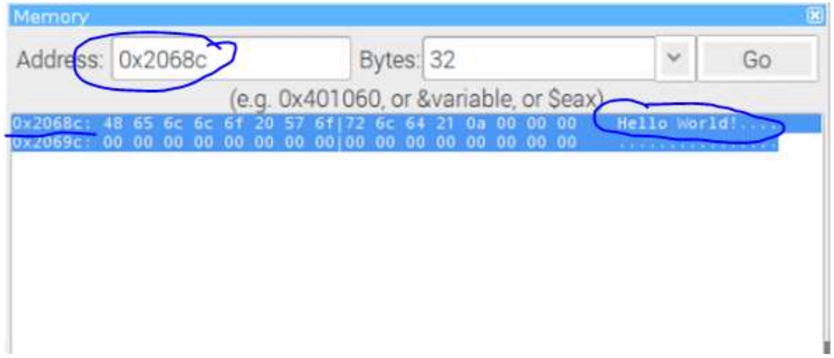
```
0x2068c: 48 65 6c 6c 6f 20 57 6f|72 6c 64 21 0a 00 00 00    Hello World!....
0x2069c: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00    ................
```

```
At /home/pi/Documents/VictorYuan_HelloWorld.s:18
At /home/pi/Documents/VictorYuan_HelloWorld.s:17
At /home/pi/Documents/VictorYuan_HelloWorld.s:18
Continuing...
```

Command:

/home/pi/Documents/VictorYuan_HelloWorld.s Unix (LF) UTF-8 Line 3, Column 29 Insert Read/Write default

Zoomed in on Memory dump/search:

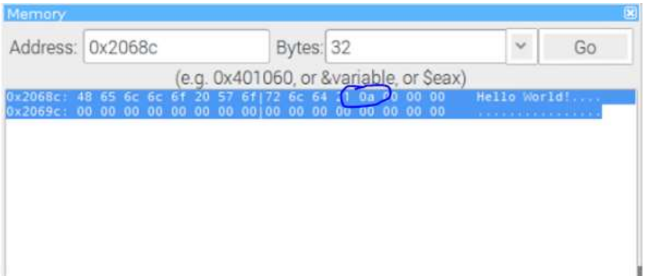

Memory

Address: 0x2068c Bytes: 32 Go

(e.g. 0x401060, or &variable, or $eax)

```
0x2068c: 48 65 6c 6c 6f 20 57 6f|72 6c 64 21 0a 00 00 00    Hello World!....
0x2069c: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00    ................
```

3. How did the program known to print each character until "\n" was reached?
   "\n" represented in ASCII is 0a once 0a was read the program realized the special character and stopped.


Memory

Address: 0x2068c Bytes: 32 Go

(e.g. 0x401060, or &variable, or $eax)

```
0x2068c: 48 65 6c 6c 6f 20 57 6f|72 6c 64 21 0a 00 00 00    Hello World!....
0x2069c: 00 00 00 00 00 00 00 00|00 00 00 00 00 00 00 00    ................
```

Conclusion:

Code Block was successfully installed onto the Raspberry. How to use a debugging tool was shown. Memory location of registers was shown.