Lab 4

September 2, 2021

Delivery date: September 2, 2021

Assembly programming in Raspbian

Javier Mondragon Martin del Campo

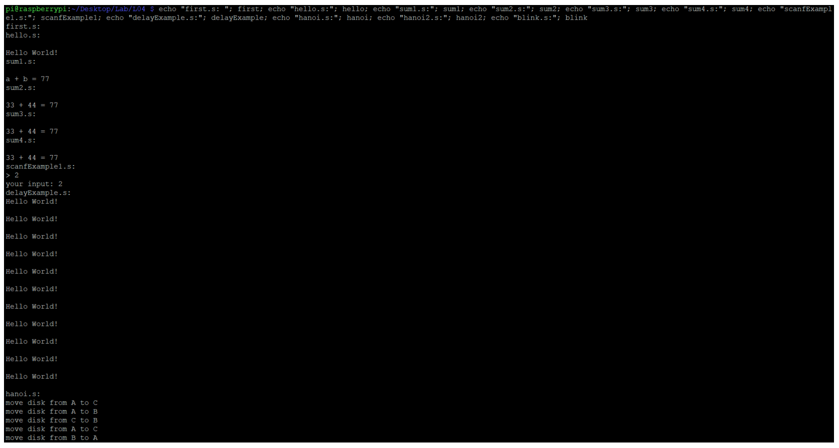
A01365137

Prof. Matías Vázquez Piñón

Tecnológico de Monterrey

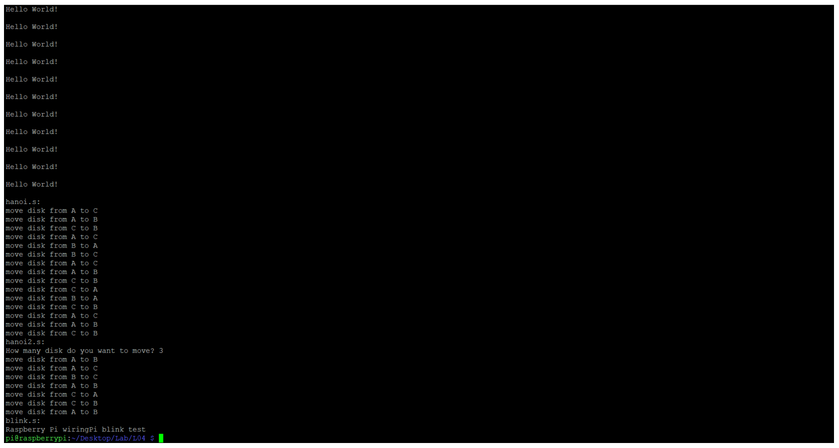
1 Activity

The "./" command has been added to the user's path in order to run a program without writing it down. Git code: `https://github.com/javiermomc/Sistemas_Embebidos/tree/main/L04` Screenshots from terminal joining all the programs on a single line:



```
pi@raspberrypi:~/Desktop/Lab/L04 $ echo "first.s: "; first; echo "hello.s:"; hello; echo "sum1.s:"; sum1; echo "sum2.s:"; sum2; echo "sum3.s:"; sum3; echo "sum4.s:"; sum4; echo "scanfExampl
e1.s:"; scanfExample1; echo "delayExample.s:"; delayExample; echo "hanoi.s:"; hanoi; echo "hanoi2.s:"; hanoi2; echo "blink.s:"; blink
first.s:
hello.s:

Hello World!
sum1.s:

a + b = 77
sum2.s:

33 + 44 = 77
sum3.s:

33 + 44 = 77
sum4.s:

33 + 44 = 77
scanfExample1.s:
> 2
your input: 2
delayExample.s:
Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

hanoi.s:
move disk from A to C
move disk from A to B
move disk from C to B
move disk from A to C
move disk from B to A
```

(a) Part 1



```
Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

Hello World!

hanoi.s:
move disk from A to C
move disk from A to B
move disk from C to B
move disk from A to C
move disk from B to A
move disk from B to C
move disk from A to C
move disk from A to B
move disk from C to B
move disk from C to A
move disk from B to A
move disk from C to B
move disk from A to C
move disk from A to B
move disk from C to B
hanoi2.s:
How many disk do you want to move? 3
move disk from A to B
move disk from A to C
move disk from B to C
move disk from A to B
move disk from C to A
move disk from C to B
move disk from A to B
blink.s:
Raspberry Pi wiringPi blink test
pi@raspberrypi:~/Desktop/Lab/L04 $
```

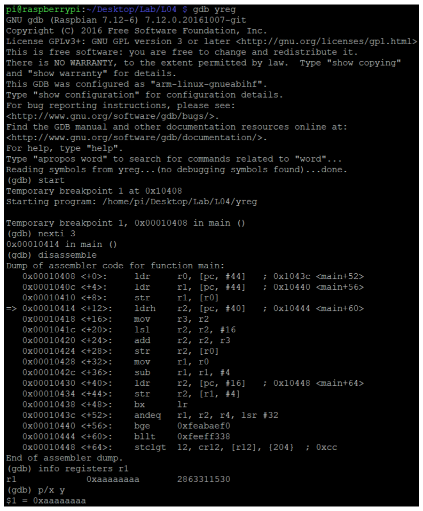
(b) Part 2

Figure 1: Output from Act. 1

2 Activity

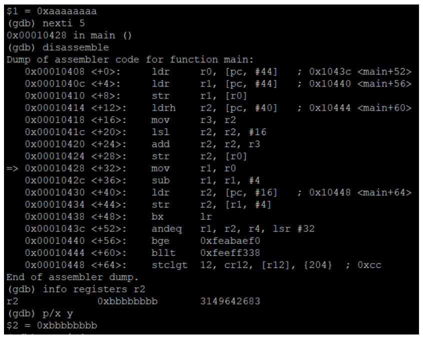
2.1 Part 1

Git code: `https://github.com/javiermomc/Sistemas_Embebidos/tree/main/L04/yreg.s` GDB outputs displaying the registers and variables values:



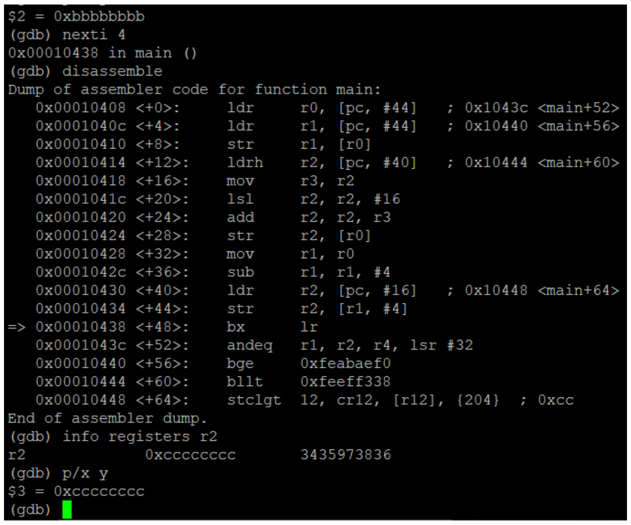
```
pi@raspberrypi:~/Desktop/Lab/L04 $ gdb yreg
GNU gdb (Raspbian 7.12-6) 7.12.0.20161007-git
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabihf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from yreg...(no debugging symbols found)...done.
(gdb) start
Temporary breakpoint 1 at 0x10408
Starting program: /home/pi/Desktop/Lab/L04/yreg

Temporary breakpoint 1, 0x00010408 in main ()
(gdb) nexti 3
0x00010414 in main ()
(gdb) disassemble
Dump of assembler code for function main:
   0x00010408 <+0>:     ldr     r0, [pc, #44]   ; 0x1043c <main+52>
   0x0001040c <+4>:     ldr     r1, [pc, #44]   ; 0x10440 <main+56>
   0x00010410 <+8>:     str     r1, [r0]
=> 0x00010414 <+12>:    ldrh    r2, [pc, #40]   ; 0x10444 <main+60>
   0x00010418 <+16>:    mov     r3, r2
   0x0001041c <+20>:    lsl     r2, r2, #16
   0x00010420 <+24>:    add     r2, r2, r3
   0x00010424 <+28>:    str     r2, [r0]
   0x00010428 <+32>:    mov     r1, r0
   0x0001042c <+36>:    sub     r1, r1, #4
   0x00010430 <+40>:    ldr     r2, [pc, #16]   ; 0x10448 <main+64>
   0x00010434 <+44>:    str     r2, [r1, #4]
   0x00010438 <+48>:    bx      lr
   0x0001043c <+52>:    andeq   r1, r2, r4, lsr #32
   0x00010440 <+56>:    bge     0xfeabaef0
   0x00010444 <+60>:    bllt    0xfeeff338
   0x00010448 <+64>:    stclgt  12, cr12, [r12], {204}  ; 0xcc
End of assembler dump.
(gdb) info registers r1
r1             0xaaaaaaaa       2863311530
(gdb) p/x y
$1 = 0xaaaaaaaa
```

(a) Part 1



```
$1 = 0xaaaaaaaa
(gdb) nexti 5
0x00010428 in main ()
(gdb) disassemble
Dump of assembler code for function main:
   0x00010408 <+0>:     ldr     r0, [pc, #44]   ; 0x1043c <main+52>
   0x0001040c <+4>:     ldr     r1, [pc, #44]   ; 0x10440 <main+56>
   0x00010410 <+8>:     str     r1, [r0]
   0x00010414 <+12>:    ldrh    r2, [pc, #40]   ; 0x10444 <main+60>
   0x00010418 <+16>:    mov     r3, r2
   0x0001041c <+20>:    lsl     r2, r2, #16
   0x00010420 <+24>:    add     r2, r2, r3
   0x00010424 <+28>:    str     r2, [r0]
=> 0x00010428 <+32>:    mov     r1, r0
   0x0001042c <+36>:    sub     r1, r1, #4
   0x00010430 <+40>:    ldr     r2, [pc, #16]   ; 0x10448 <main+64>
   0x00010434 <+44>:    str     r2, [r1, #4]
   0x00010438 <+48>:    bx      lr
   0x0001043c <+52>:    andeq   r1, r2, r4, lsr #32
   0x00010440 <+56>:    bge     0xfeabaef0
   0x00010444 <+60>:    bllt    0xfeeff338
   0x00010448 <+64>:    stclgt  12, cr12, [r12], {204}  ; 0xcc
End of assembler dump.
(gdb) info registers r2
r2             0xbbbbbbbb       3149642683
(gdb) p/x y
$2 = 0xbbbbbbbb
```

(b) Part 2



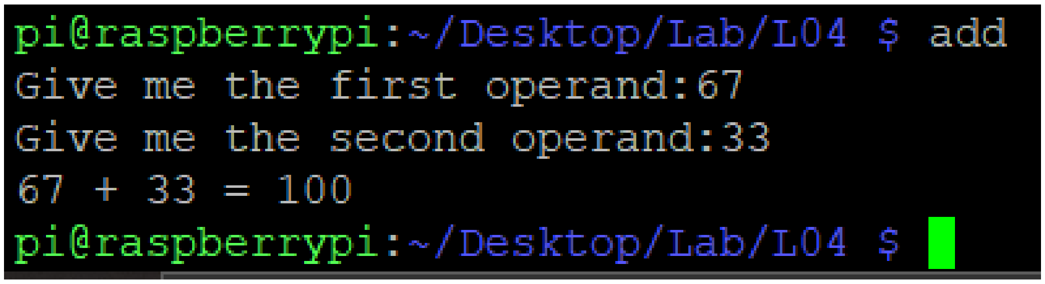
```
$2 = 0xbbbbbbbb
(gdb) nexti 4
0x00010438 in main ()
(gdb) disassemble
Dump of assembler code for function main:
   0x00010408 <+0>:     ldr     r0, [pc, #44]   ; 0x1043c <main+52>
   0x0001040c <+4>:     ldr     r1, [pc, #44]   ; 0x10440 <main+56>
   0x00010410 <+8>:     str     r1, [r0]
   0x00010414 <+12>:    ldrh    r2, [pc, #40]   ; 0x10444 <main+60>
   0x00010418 <+16>:    mov     r3, r2
   0x0001041c <+20>:    lsl     r2, r2, #16
   0x00010420 <+24>:    add     r2, r2, r3
   0x00010424 <+28>:    str     r2, [r0]
   0x00010428 <+32>:    mov     r1, r0
   0x0001042c <+36>:    sub     r1, r1, #4
   0x00010430 <+40>:    ldr     r2, [pc, #16]   ; 0x10448 <main+64>
   0x00010434 <+44>:    str     r2, [r1, #4]
=> 0x00010438 <+48>:    bx      lr
   0x0001043c <+52>:    andeq   r1, r2, r4, lsr #32
   0x00010440 <+56>:    bge     0xfeabaef0
   0x00010444 <+60>:    bllt    0xfeeff338
   0x00010448 <+64>:    stclgt  12, cr12, [r12], {204}  ; 0xcc
End of assembler dump.
(gdb) info registers r2
r2             0xcccccccc       3435973836
(gdb) p/x y
$3 = 0xcccccccc
(gdb)
```

(c) Part 3

Figure 2: Output from Act. 2. Section 1

2.2 Part 2

Git code: `https://github.com/javiermomc/Sistemas_Embebidos/tree/main/L04/add.s` Adder program testing 67+33:

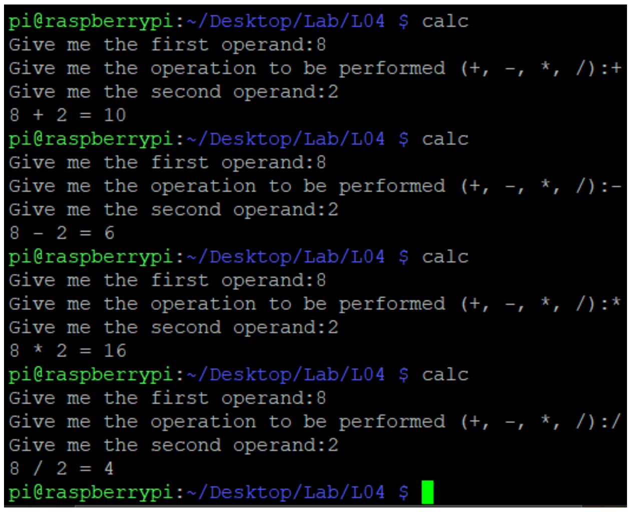


```
pi@raspberrypi:~/Desktop/Lab/L04 $ add
Give me the first operand:67
Give me the second operand:33
67 + 33 = 100
pi@raspberrypi:~/Desktop/Lab/L04 $ 
```

Figure 3: Output from Act. 2. Section 2

2.3 Part 3 & 4

Git code: `https://github.com/javiermomc/Sistemas_Embebidos/tree/main/L04/calc.s` For this section, there was an investigation regarding the division and multiplication in the arm reference (2). There was multiple testing adding, subtracting, multiplying and division on the following image:

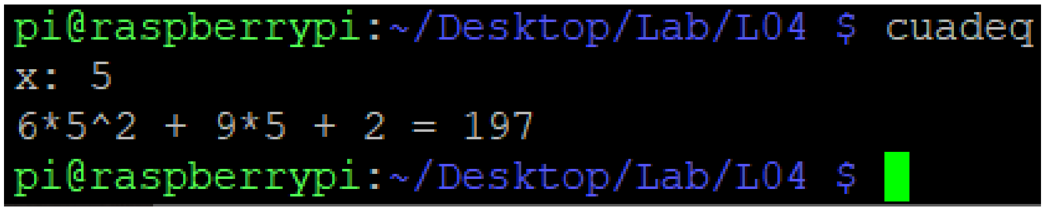


```
pi@raspberrypi:~/Desktop/Lab/L04 $ calc
Give me the first operand:8
Give me the operation to be performed (+, -, *, /):+
Give me the second operand:2
8 + 2 = 10
pi@raspberrypi:~/Desktop/Lab/L04 $ calc
Give me the first operand:8
Give me the operation to be performed (+, -, *, /):-
Give me the second operand:2
8 - 2 = 6
pi@raspberrypi:~/Desktop/Lab/L04 $ calc
Give me the first operand:8
Give me the operation to be performed (+, -, *, /):*
Give me the second operand:2
8 * 2 = 16
pi@raspberrypi:~/Desktop/Lab/L04 $ calc
Give me the first operand:8
Give me the operation to be performed (+, -, *, /):/
Give me the second operand:2
8 / 2 = 4
pi@raspberrypi:~/Desktop/Lab/L04 $ 
```

Figure 4: Output from Act. 2. Section 3 & 4

2.4 Part 5

Git code: `https://github.com/javiermomc/Sistemas_Embebidos/tree/main/L04/cuadeq.s` Testing the code with input of 5:



```
pi@raspberrypi:~/Desktop/Lab/L04 $ cuadeq
x: 5
6*5^2 + 9*5 + 2 = 197
pi@raspberrypi:~/Desktop/Lab/L04 $ 
```

Figure 5: Output from Act. 2. Section 5

3 Conclusions

This lab help me get more involve in the raspberry environment and apply some knowledge from Operating Systems course. Some of the concepts for the assembly code used in raspberry were clarified, more specifically some operations like "ldr". Also developed a script for faster compiling called "cas" (already on git).

4 Bibliography

1. https://github.com/matias-vazquez/SistemasEmbebidos

2. https://developer.arm.com/documentation/den0024/a/The-A64-instruction-set/Data-processing-instructions/Multiply-and-divide-instructions