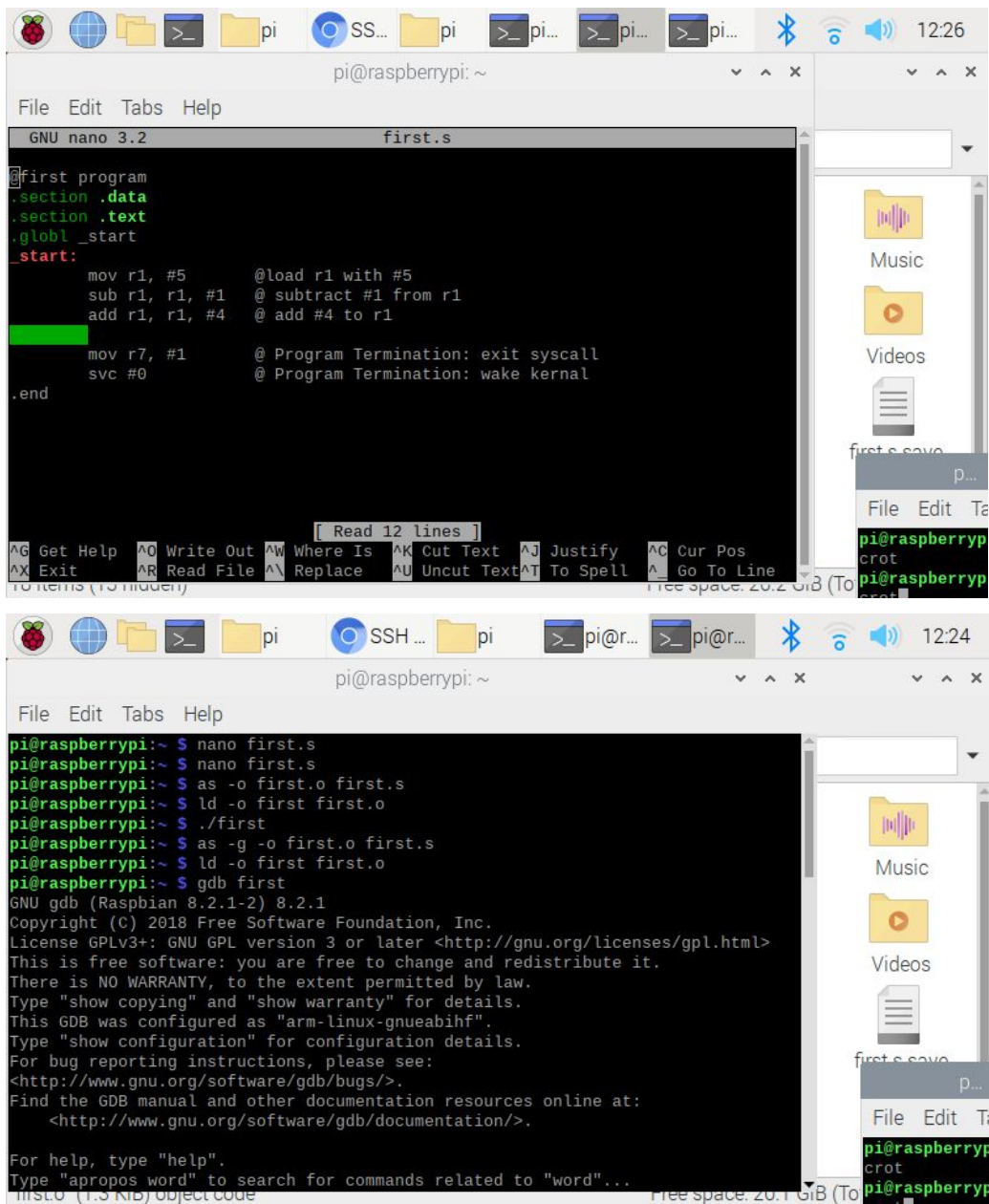


Matthew Maloof
Team x87

Part 4 Report

The first thing that I did was connect my Raspberry Pi to github and SSH. After doing so, I created a file first.s with the contents below and saved it. I ran the program using ./first and nothing was outputted because it doesn't produce an output without viewing the registers.



The first screenshot shows the GNU nano 3.2 editor with the file first.s. The code is as follows:

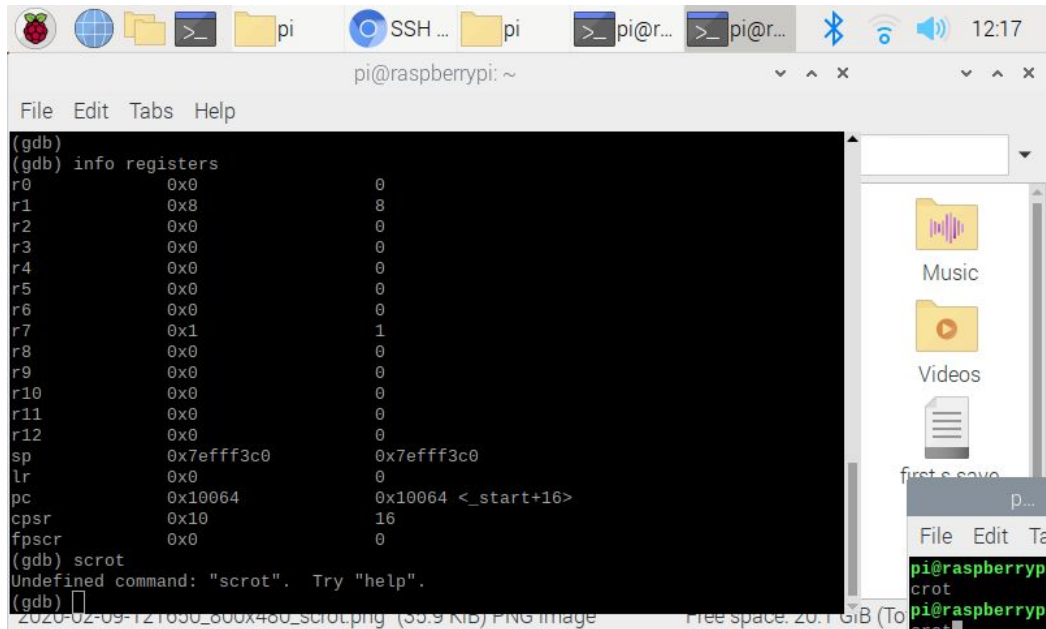
```
@first program
.section .data
.section .text
.globl _start
_start:
    mov r1, #5      @load r1 with #5
    sub r1, r1, #1  @ subtract #1 from r1
    add r1, r1, #4  @ add #4 to r1

    mov r7, #1      @ Program Termination: exit syscall
    svc #0          @ Program Termination: wake kernal
.end
```

The second screenshot shows the terminal output after running the program. The user has executed the following commands:

```
pi@raspberrypi:~$ nano first.s
pi@raspberrypi:~$ nano first.s
pi@raspberrypi:~$ as -o first.o first.s
pi@raspberrypi:~$ ld -o first first.o
pi@raspberrypi:~$ ./first
pi@raspberrypi:~$ as -g -o first.o first.s
pi@raspberrypi:~$ ld -o first first.o
pi@raspberrypi:~$ gdb first
GNU gdb (Raspbian 8.2.1-2) 8.2.1
Copyright (C) 2018 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-gnueabi".
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"...
```

---- Next, I opened gdp with the file to debug the program and view the outputs. On the next page is the result of each output for the file "first".



The screenshot shows a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~'. The terminal output is as follows:

```
(gdb) info registers
r0      0x0      0
r1      0x8      8
r2      0x0      0
r3      0x0      0
r4      0x0      0
r5      0x0      0
r6      0x0      0
r7      0x1      1
r8      0x0      0
r9      0x0      0
r10     0x0      0
r11     0x0      0
r12     0x0      0
sp      0x7efff3c0 0x7efff3c0
lr      0x0      0
pc      0x10064    0x10064 <_start+16>
cpsr    0x10     16
fpscr   0x0      0
(gdb) scrot
Undefined command: "scrot". Try "help".
(gdb)
```

Below the terminal window, a file manager sidebar is visible, showing folders for Music and Videos. At the bottom of the terminal window, a status bar indicates 'Free space: 20.1 GiB (To'.

Above is the output shown during debugging. This ultimately shows a 1 for register 7.

```
pi@raspberrypi: ~
File Edit Tabs Help
9      mov r4, #2      @moves #2 into register 4 (D)
10
(gdb)
11      add r5, r1, r2 @adds r1 (A) and r2 (B) and places result into r5
12
13      mul r6, r3, r4 @multiplies value of r3 (C) and r4 (D) and places
into r6
14
15      sub r1, r5, r6 @subtracts r5 and r6 and places result into r1 (A
)
16
17      mov r7, #1      @Program Termination: exit syscall
18
19      svc #0          @Program Termination: wake kernel
20
.end
(gdb) b 11
Breakpoint 1 at 0x10064: file arithmetic1.s, line 11.
(gdb) b 13
Breakpoint 2 at 0x10068: file arithmetic1.s, line 13.
(gdb) b 15
Breakpoint 3 at 0x1006c: file arithmetic1.s, line 15.
(gdb) b 17
Breakpoint 4 at 0x10070: file arithmetic1.s, line 17.
(gdb) █
```

Next, breakpoints were set for lines 11, 13, 15, and 17 for part 2 of the assignment, with the creation of arithmetic1 program in the top of the screenshot.

```
pi@raspberrypi: ~
File Edit Tabs Help
Breakpoint 1, _start () at arithmetic1.s:11
11      add r5, r1, r2 @adds r1 (A) and r2 (B) and places result into r5
(gdb)
(gdb) info registers
r0          0x0          0
r1          0xa         10
r2          0xb         11
r3          0x7         7
r4          0x2         2
r5          0x0         0
r6          0x0         0
r7          0x0         0
r8          0x0         0
r9          0x0         0
r10         0x0         0
r11         0x0         0
r12         0x0         0
sp          0x7efff3b0   0x7efff3b0
lr          0x0         0
pc          0x10064     0x10064 <_start+16>
cpsr       0x10        16
fpscr      0x0         0
(gdb)
r0          0x0         0
```

Breakpoint 1 is above, with it showing values A(10),B(11),C(7) and D(2) in registers 1, 2, 3, 4 respectively.

```
pi@raspberrypi: ~
File Edit Tabs Help

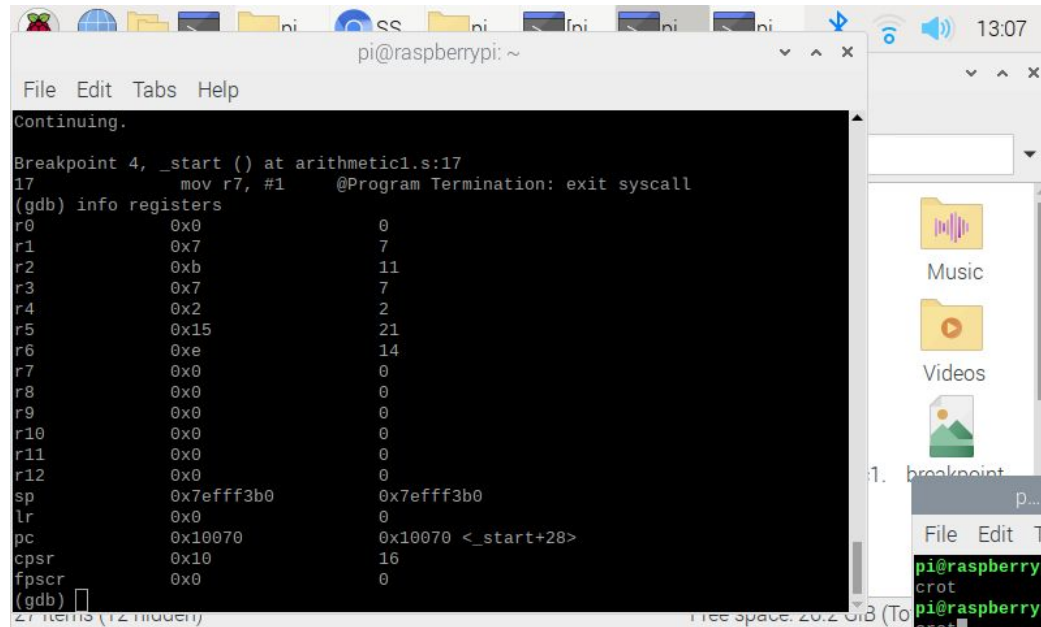
Breakpoint 2, _start () at arithmetic1.s:13
13      mul r6, r3, r4 @multiplies value of r3 (C) and r4 (D) and places
    into r6
(gdb) info registers
r0          0x0          0
r1          0xa          10
r2          0xb          11
r3          0x7          7
r4          0x2          2
r5          0x15         21
r6          0x0          0
r7          0x0          0
r8          0x0          0
r9          0x0          0
r10         0x0          0
r11         0x0          0
r12         0x0          0
sp          0x7efff3b0    0x7efff3b0
lr          0x0          0
pc          0x10068       0x10068 <_start+20>
cpsr        0x10         16
fpscr       0x0          0
(gdb)
```

Breakpoint 2 above shows the addition being done between registers 1 and 2 and placing result in register 5. ($10+11=21$)

```
pi@raspberrypi: ~
File Edit Tabs Help

Breakpoint 3, _start () at arithmetic1.s:15
15      sub r1, r5, r6 @subtracts r5 and r6 and places result into r1 (A
    )
(gdb) info registers
r0          0x0          0
r1          0xa          10
r2          0xb          11
r3          0x7          7
r4          0x2          2
r5          0x15         21
r6          0xe          14
r7          0x0          0
r8          0x0          0
r9          0x0          0
r10         0x0          0
r11         0x0          0
r12         0x0          0
sp          0x7efff3b0    0x7efff3b0
lr          0x0          0
pc          0x1006c       0x1006c <_start+24>
cpsr        0x10         16
fpscr       0x0          0
(gdb)
```

Breakpoint 3 shows the multiplication of registers 3 and 4 placing result into register 6 ($7*2=14$)

A screenshot of a Raspberry Pi terminal window. The window title is 'pi@raspberrypi: ~'. The terminal output shows the GDB 'info registers' command. It lists registers r0 through r12, along with special registers sp, lr, pc, cpsr, and fpscr. The values for r0-r12 are shown in hexadecimal and decimal. Specifically, r5 has a decimal value of 21 and r6 has a decimal value of 14. The program counter (pc) is at address 0x10070, which is labeled as '<_start+28>'. The terminal also shows a breakpoint hit at line 17 of 'arithmetic1.s', where the instruction is 'mov r7, #1' and the reason for the breakpoint is '@Program Termination: exit syscall'.

```
Continuing.
Breakpoint 4, _start () at arithmetic1.s:17
17      mov r7, #1      @Program Termination: exit syscall
(gdb) info registers
r0          0x0         0
r1          0x7         7
r2          0xb        11
r3          0x7         7
r4          0x2         2
r5          0x15        21
r6          0xe        14
r7          0x0         0
r8          0x0         0
r9          0x0         0
r10         0x0         0
r11         0x0         0
r12         0x0         0
sp          0x7efff3b0   0x7efff3b0
lr          0x0         0
pc          0x10070      0x10070 <_start+28>
cpsr       0x10        16
fpscr      0x0         0
(gdb)
```

Finally, breakpoint 4 shows subtraction of register 6 and 5 to get the value of 7 placed in register 1. (21-14=7)