

Zach Dubinsky

CSC-270-01

Prof. Rieffel

2/17/22

## Lab 5

1. The following image shows the GDB debugger with my sum.s code loaded. The first block is the compiled code, and the subsequent blocks show the contents of registers *r0* and *r1* at a few breakpoints. The output of sum.s is the final line of the GDB output, with a sum of 258.

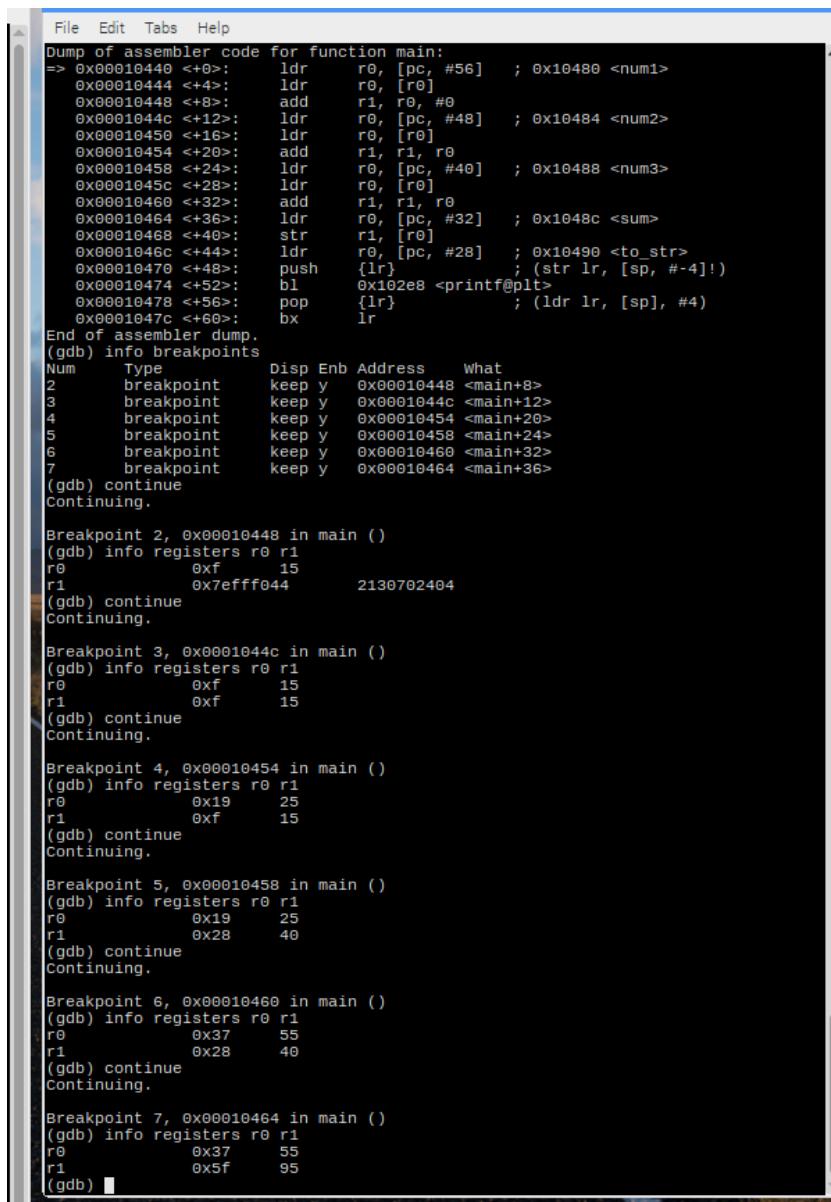
```
Temporary breakpoint 1, 0x00010440 in main ()
(gdb) disassemble
Dump of assembler code for function main:
=> 0x00010440 <+0>: ldr    r0, [pc, #56]   ; 0x10480 <num1>
 0x00010444 <+4>: ldr    r0, [r0]
 0x00010448 <+8>: add    r1, r0, #0
 0x0001044c <+12>: ldr    r0, [pc, #48]   ; 0x10484 <num2>
 0x00010450 <+16>: ldr    r0, [r0]
 0x00010454 <+20>: add    r1, r1, r0
 0x00010458 <+24>: ldr    r0, [pc, #40]   ; 0x10488 <num3>
 0x0001045c <+28>: ldr    r0, [r0]
 0x00010460 <+32>: add    r1, r1, r0
 0x00010464 <+36>: ldr    r0, [pc, #32]   ; 0x1048c <sum>
 0x00010468 <+40>: str    r1, [r0]
 0x0001046c <+44>: ldr    r0, [pc, #28]   ; 0x10490 <to_str>
 0x00010470 <+48>: push   {lr}           ; (str lr, [sp, #-4]!)
 0x00010474 <+52>: bl    0x102e8 <printf@plt>
 0x00010478 <+56>: pop    {lr}           ; (ldr lr, [sp], #4)
 0x0001047c <+60>: bx    lr
End of assembler dump.
(gdb) break *1044c
Invalid number "1044c".
(gdb) break *0x1044c
Breakpoint 2 at 0x1044c
(gdb) break *0x10458
Breakpoint 3 at 0x10458
(gdb) break *0x10464
Breakpoint 4 at 0x10464
(gdb) info registers r0 r1
r0          0x1      1
r1          0x7efff044  2130702404
(gdb) continue
Continuing.

Breakpoint 2, 0x0001044c in main ()
(gdb) info registers r0 r1
r0          0x11     17
r1          0x11     17
(gdb) continue
Continuing.

Breakpoint 3, 0x00010458 in main ()
(gdb) info registers r0 r1
r0          0xfffffffdd  4294967261
r1          0xfffffffee  4294967278
(gdb) continue
Continuing.

Breakpoint 4, 0x00010464 in main ()
(gdb) info registers r0 r1
r0          0x114    276
r1          0x102    258
(gdb) continue
Continuing.
258
[Inferior 1 (process 3225) exited with code 04]
(gdb) █
```

2. The following images show the GDB debugger with my sum2.s program loaded. The first block of output is the compiled ARM code, with memory addresses on the left-hand-side corresponding to the instruction address. I use these to set breakpoint locations which I had GDB output in the next block. The subsequent calls to *continue* and *info registers r0 r1* show the change in these registers. I have only shown these two, because they are the only ones my program uses. The output of the program is included in the next image as it did not fit in the first



```

File Edit Tabs Help
Dump of assembler code for function main:
=> 0x00010440 <+0>: ldr    r0, [pc, #56] ; 0x10480 <num1>
0x00010444 <+4>: ldr    r0, [r0]
0x00010448 <+8>: add    r1, r0, #0
0x0001044c <+12>: ldr    r0, [pc, #48] ; 0x10484 <num2>
0x00010450 <+16>: ldr    r0, [r0]
0x00010454 <+20>: add    r1, r1, r0
0x00010458 <+24>: ldr    r0, [pc, #40] ; 0x10488 <num3>
0x0001045c <+28>: ldr    r0, [r0]
0x00010460 <+32>: add    r1, r1, r0
0x00010464 <+36>: ldr    r0, [pc, #32] ; 0x1048c <sum>
0x00010468 <+40>: str    r1, [r0]
0x0001046c <+44>: ldr    r0, [pc, #28] ; 0x10490 <to_str>
0x00010470 <+48>: push   {lr} ; (str lr, [sp, #-4]!)
0x00010474 <+52>: bl    0x102e8 <printf@plt>
0x00010478 <+56>: pop    {lr} ; (ldr lr, [sp], #4)
0x0001047c <+60>: bx    lr
End of assembler dump.
(gdb) info breakpoints
Num      Type      Disp Enb Address      What
2        breakpoint keep y 0x00010448 <main+8>
3        breakpoint keep y 0x0001044c <main+12>
4        breakpoint keep y 0x00010454 <main+20>
5        breakpoint keep y 0x00010458 <main+24>
6        breakpoint keep y 0x00010460 <main+32>
7        breakpoint keep y 0x00010464 <main+36>
(gdb) continue
Continuing.

Breakpoint 2, 0x00010448 in main ()
(gdb) info registers r0 r1
r0          0xf      15
r1          0x7efff044  2130702404
(gdb) continue
Continuing.

Breakpoint 3, 0x0001044c in main ()
(gdb) info registers r0 r1
r0          0xf      15
r1          0xf      15
(gdb) continue
Continuing.

Breakpoint 4, 0x00010454 in main ()
(gdb) info registers r0 r1
r0          0x19     25
r1          0xf      15
(gdb) continue
Continuing.

Breakpoint 5, 0x00010458 in main ()
(gdb) info registers r0 r1
r0          0x19     25
r1          0x28     40
(gdb) continue
Continuing.

Breakpoint 6, 0x00010460 in main ()
(gdb) info registers r0 r1
r0          0x37     55
r1          0x28     40
(gdb) continue
Continuing.

Breakpoint 7, 0x00010464 in main ()
(gdb) info registers r0 r1
r0          0x37     55
r1          0x5f     95
(gdb) 
```

```

File Edit Tabs Help
pi@rpi5:~ $ cd Desktop/Lab5
bash: cd: Desktop/Lab5: No such file or directory
pi@rpi5:~ $ cd Desktop/Lab_5
pi@rpi5:~/Desktop/Lab_5 $ ./sum2
95
pi@rpi5:~/Desktop/Lab_5 $ 

```

3. The following image shows the swap.s program when loaded into the GDB debugger.

Following the loading prompt, the first block shows the compiled program with instruction memory addresses for reference. The second block shows the contents of data memory at the address of the array before the swap, with entries of data memory that are no in the array crossed out. The final block shows the contents of data memory after the swap has been performed, with arrows indicating the elements that have been swapped.

All data not in the array has been crossed out.

```

Temporary breakpoint 1, 0x00010440 in main ()
(gdb) disassemble
Dump of assembler code for function main:
=> 0x00010440 <+0>: ldr   r0, [pc, #80] ; 0x1049B <index1>
0x00010444 <+4>: ldr   r0, [r0]
0x00010448 <+8>: ldr   r1, [pc, #76] ; 0x1049C <index2>
0x0001044c <+12>: ldr   r1, [r1]
0x00010450 <+16>: ldr   r2, [pc, #56] ; 0x10490 <arr>
0x00010454 <+20>: ldr   r3, [r2, r0, lsl #2]
0x00010458 <+24>: ldr   r4, [r2, r1, lsl #2]
0x0001045c <+28>: str   r3, [r2, r1, lsl #2]
0x00010460 <+32>: str   r4, [r2, r0, lsl #2]
0x00010464 <+36>: push  [lr]           ; (str lr, [sp, #-4])
0x00010468 <+40>: mov    r5, r2
0x0001046c <+44>: ldr   r4, [pc, #32] ; 0x10494 <endarr>
End of assembler dump.
(gdb) x
Argument required (starting display address).
(gdb) x 0x10490
0x10490 <arr>: 0x00021028
(gdb) x/25w 0x21028
0x21028: 0x00000000 0x00000001 0x00000002 0x00000003
0x21038: 0x00000004 0x00000000 0x00000000 0x00000000
0x21048: 0x00000000 0x00000000 0x00000000 0x00000000
0x21058: 0x000206425 0x00000000 0x00000000 0x00000000
0x21068: 0x00000000 0x00000000 0x00000000 0x00000000
0x21078: 0x00000000 0x00000000 0x00000000 0x00000000
0x21088: 0x00000000 0x00000000 0x00000000 0x00000000
(gdb) break *0x10464
Breakpoint 2 at 0x10464
(gdb) continue
Continuing.

Breakpoint 2, 0x00010464 in main ()
(gdb) x/25w 0x21028
0x21028: 0x00000002 0x00000001 0x00000000 0x00000003
0x21038: 0x00000004 0x00000000 0x00000000 0x00000002
0x21048: 0x00000000 0x00000000 0x00000000 0x00000002
0x21058: 0x000206425 0x00000000 0x00000000 0x00000000
0x21068: 0x00000000 0x00000000 0x00000000 0x00000000
0x21078: 0x00000000 0x00000000 0x00000000 0x00000000
0x21088: 0x00000000 0x00000000 0x00000000 0x00000000
(gdb) 

```