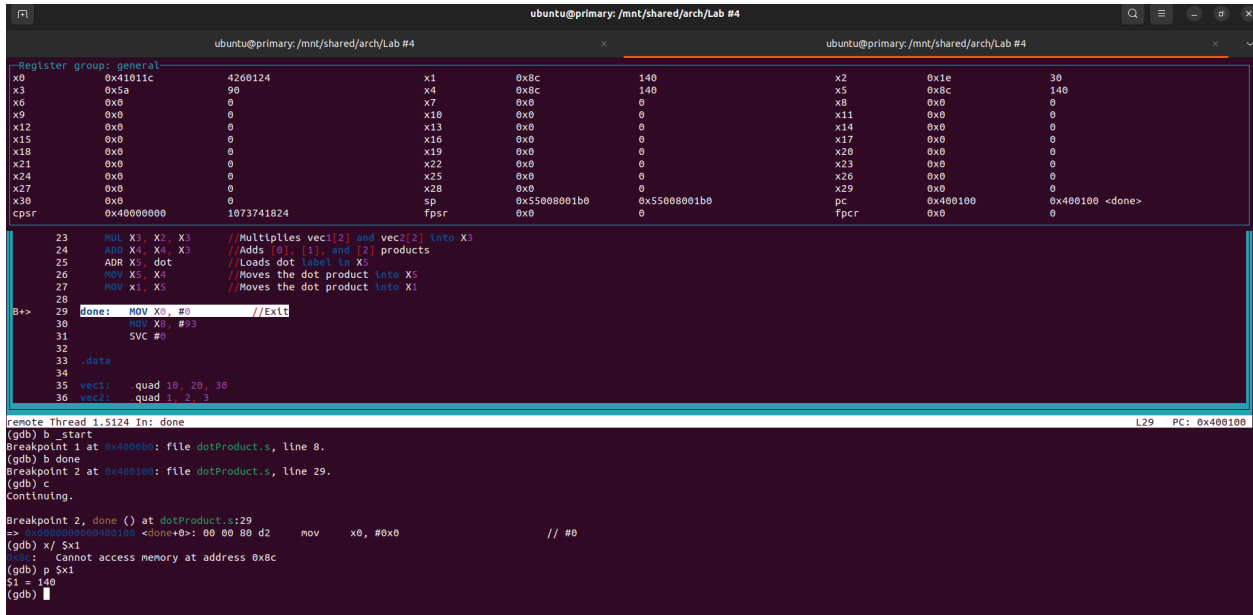


Pledge: I pledge my honor that I have abided by the Stevens Honor System.

Name: Haig Emirzian

Lab #4 With GDB



The screenshot shows a GDB session with the following components:

- Register group: general**

Register	Value	Register	Value	Register	Value	Register	Value	
x0	0x41011c	4260124	x1	0x8c	140	x2	0x1e	30
x3	0x5a	90	x4	0x8c	140	x5	0x8c	140
x6	0x0	0	x7	0x0	0	x8	0x0	0
x9	0x0	0	x10	0x0	0	x11	0x0	0
x12	0x0	0	x13	0x0	0	x14	0x0	0
x15	0x0	0	x16	0x0	0	x17	0x0	0
x18	0x0	0	x19	0x0	0	x20	0x0	0
x21	0x0	0	x22	0x0	0	x23	0x0	0
x24	0x0	0	x25	0x0	0	x26	0x0	0
x27	0x0	0	x28	0x0	0	x29	0x0	0
x30	0x0	0	sp	0x55008001b0	0x55008001b0	pc	0x400100	0x400100 <done>
cpsr	0x40000000	1073741824	fpsr	0x0	0	fpcr	0x0	0
- Assembly code:**

```
23 MUL X3, X2, X3 //Multiplies vec1[2] and vec2[2] into X3
24 ADD X4, X4, X3 //Adds 0, [1], and [2] products
25 ADR X5, dot //Loads dot label in X5
26 MOV X5, X4 //Moves the dot product into X5
27 MOV X1, X5 //Moves the dot product into X1
28
29 done: MOV X0, #0 //Exit
30 MOV X3, #93
31 SVC #0
32
33 .data
34
35 vec1: quad 10, 20, 30
36 vec2: quad 1, 2, 3
```
- GDB Output:**

```
remote Thread 1.S124 In: done
(gdb) b _start
Breakpoint 1 at 0x400000: file dotProduct.s, line 8.
(gdb) b done
Breakpoint 2 at 0x400100: file dotProduct.s, line 29.
(gdb) c
Continuing.
Breakpoint 2, done () at dotProduct.s:29
=> 0x0000000000400100: <done+0>: 00 00 00 d2 mov x0, #0x0 // #0
(gdb) x/ $x1
Cannot access memory at address 0x8c
(gdb) p $x1
$1 = 140
(gdb)
```

After setting up my terminals, my first commands were two breakpoints. The first breakpoint is at the start of the program: **b _start**. My second breakpoint is at the label where my program finishes: **b _done**. I used the continue command to jump straight to the done label to exit the code: **c**. I then used **x/ \$x1** to find the value of X1, but I couldn't access it. I went back to look at my code and I forgot to store the value back into memory. Therefore, I went back to my code and stored the dot product into register X5

```

Register group: general
x0      0x410118      4260120      x1      0x410130      4260144      x2      0x1e        30
x3      0x5a         90          x4      0x8c         140          x5      0x410138      4260152
x6      0x0          0           x7      0x0          0           x8      0x0          0
x9      0x0          0           x10     0x0          0           x11     0x0          0
x12     0x0          0           x13     0x0          0           x14     0x0          0
x15     0x0          0           x16     0x0          0           x17     0x0          0
x18     0x0          0           x19     0x0          0           x20     0x0          0
x21     0x0          0           x22     0x0          0           x23     0x0          0
x24     0x0          0           x25     0x0          0           x26     0x0          0
x27     0x0          0           x28     0x0          0           x29     0x0          0
x30     0x0          0           x29     0x0          0           x29     0x0          0
x30     0x0          0           sp      0x55008001b0    0x55008001b0    pc      0x4000fc      0x4000fc <done>
cpsr    0x40000000    1073741824    fpsr    0x0          0           fpcr    0x0          0

22      LDUR X3, [X1]      //Loads vec2[2] into X3
23      MUL X3, X2, X3     //Multiplies vec1[2] and vec2[2] into X3
24      ADD X4, X3, X3     //Adds [0], [1], and [2] products
25      ADR X5, dot        //Loads dot label to X5
26      STUR X4, [X2]      //Loads the summation into the dot register
27
B+> 28      done: MOV X0, #0      //Exit
29      MOV X0, #99
30      SVC #0
31
32      .data
33
34      vec1: quad 10, 20, 30
35      vec2: quad 1, 2, 3

remote Thread 1.5053 In: done
(gdb) b _start
Breakpoint 1 at 0x4000b0: file dotProduct.s, line 8.
(gdb) c
Breakpoint 2 at 0x4000fc: file dotProduct.s, line 28.
(gdb) c
Continuing.
Breakpoint 2, done () at dotProduct.s:28
=> 0x4000fc: 0x4000fc <done+0>: 00 00 80 d2      mov     x0, #0x0      // #0
(gdb) x/ $x5
0x410138: 140
(gdb)

```

I then used the command `x/ $x5` to dereference register 5 and print out its value, which happens to be the dot product of **140**.

Lab #5 With Default Points (0,0) (0,2) (2,0)

```

[ Register Values Unavailable ]

conditionals.s
B+> 11      ADR X0, p1      //Loads p1 label into X0
12      ADR X1, p2      //Loads p2 label into X1
13      ADR X2, p3      //Loads p3 label into X2
14
15      LDUR X3, [X0]     //Gets x and y from p1
16      LDUR X4, [X0]     //Loads p1[0] into X3
17      ADD X4, X0, #0     //Adds 0 to X0 to get p1[1]
18      LDUR X5, [X0]     //Loads p1[1] into X5
19
20      LDUR X6, [X1]     //Gets x and y from p2
21      LDUR X7, [X1]     //Loads p2[0] into X6
22      ADD X7, X1, #0     //Adds 0 to X1 to get p2[1]
23      LDUR X8, [X1]     //Loads p2[1] into X8
24

remote Thread 1.4210 In: start
(gdb) b _start
Breakpoint 1 at 0x4000b0: file conditionals.s, line 11.
(gdb) c
Breakpoint 2 at 0x40014c: file conditionals.s, line 76.
(gdb)

```

After setting up my terminals, my first commands were two breakpoints. The first breakpoint is at the start of the program: `b _start`. My second breakpoint is at the label where my program finishes: `b _done`.

```

Register group: general
x0 0x410158 4260184 x1 0x410188 4260232 x2 0x410178 4260216
x3 0x0 0 x4 0x410160 4260192 x5 0x0 0
x6 0x0 0 x7 0x410170 4260208 x8 0x0 0
x9 0x2 2 x10 0x410180 4260224 x11 0x2 2
x12 0x0 0 x13 0x0 0 x14 0x0 0
x15 0x0 0 x16 0x0 0 x17 0x0 0
x18 0x0 0 x19 0x0 0 x20 0x0 0
x21 0x0 0 x22 0x0 0 x23 0x0 0
x24 0x0 0 x25 0x0 0 x26 0x0 0
x27 0x0 0 x28 0x0 0 x29 0x0 0
x30 0x0 0 sp 0x55008001b0 0x55008001b0 pc 0x40014c <done>
cpsr 0x40000000 1073741824 fpsr 0x0 0 fpcr 0x0 0

70
71 //Code is executed if legs are equal to hypotenuse
72 equal: ADR X1, yes //Outputs yes label
73
74
75
B+> 76 done: MOV X0, #0 //Exits the program
77 mov X0, #93
78 SVC #0
79
80
81 .data
82 p1: quad 0, 0
83 p2: quad 0, 2

remote Thread 1.4210 In: done
(gdb) b _start
Breakpoint 1 at 0x4000b0: file conditionals.s, line 11.
(gdb) b done
Breakpoint 2 at 0x40014c: file conditionals.s, line 76.
(gdb) c
Continuing.

Breakpoint 2, done () at conditionals.s:76
=> 0x000000000040014c <done+0>: 00 00 80 d2 mov x0, #0x0 // #0
(gdb)

```

I used the continue command to jump straight to the done label to exit the code: `c`.

```

Register group: general
x0 0x410158 4260184 x1 0x410188 4260232 x2 0x410178 4260216
x3 0x0 0 x4 0x410160 4260192 x5 0x0 0
x6 0x0 0 x7 0x410170 4260208 x8 0x0 0
x9 0x2 2 x10 0x410180 4260224 x11 0x2 2
x12 0x0 0 x13 0x0 0 x14 0x0 0
x15 0x0 0 x16 0x0 0 x17 0x0 0
x18 0x0 0 x19 0x0 0 x20 0x0 0
x21 0x0 0 x22 0x0 0 x23 0x0 0
x24 0x0 0 x25 0x0 0 x26 0x0 0
x27 0x0 0 x28 0x0 0 x29 0x0 0
x30 0x0 0 sp 0x55008001b0 0x55008001b0 pc 0x40014c <done>
cpsr 0x40000000 1073741824 fpsr 0x0 0 fpcr 0x0 0

70
71 //Code is executed if legs are equal to hypotenuse
72 equal: ADR X1, yes //Outputs yes label
73
74
75
B+> 76 done: MOV X0, #0 //Exits the program
77 mov X0, #93
78 SVC #0
79
80
81 .data
82 p1: quad 0, 0
83 p2: quad 0, 2

remote Thread 1.4446 In: done
(gdb) b _start
Breakpoint 1 at 0x4000b0: file conditionals.s, line 11.
(gdb) b done
Breakpoint 2 at 0x40014c: file conditionals.s, line 76.
(gdb) c
Continuing.

Breakpoint 2, done () at conditionals.s:76
=> 0x000000000040014c <done+0>: 00 00 80 d2 mov x0, #0x0 // #0
(gdb) x/s 4260232
0x410188: "It is a right triangle."
(gdb)

```

After continuing, I tried printing the string with the `x/s <address value>` command. The address value for X1 is 4260232; therefore, I typed `x/s 4260232`. As expected, the default coordinates do form a triangle.

```

Register group: general
x0 0x410158 4260184 x1 0x410188 4260232 x2 0x410178 4260216
x3 0x0 0 x4 0x410160 4260192 x5 0x0 0
x6 0x0 0 x7 0x410170 4260208 x8 0x0 0
x9 0x2 2 x10 0x410180 4260224 x11 0x2 2
x12 0x0 0 x13 0x0 0 x14 0x0 0
x15 0x0 0 x16 0x0 0 x17 0x0 0
x18 0x0 0 x19 0x0 0 x20 0x0 0
x21 0x0 0 x22 0x0 0 x23 0x0 0
x24 0x0 0 x25 0x0 0 x26 0x0 0
x27 0x0 0 x28 0x0 0 x29 0x0 0
x30 0x0 0 sp 0x55008001b0 0x55008001b0 pc 0x40014c <done>
cpsr 0x40000000 1073741824 fpsr 0x0 0 fpcr 0x0 0

70
71 //Code is executed if legs are equal to hypotenuse
72 equal: ADR X1, yes //Outputs yes label
73
74
75
B+> 76 Done: MOV X0, #0 //Exits the program
77 MOV X8, #93
78 SVC #0
79
80
81 .data
82 p1: quad 0, 0
83 p2: quad 0, 2

remote Thread 1.4722 In: done
(gdb) b _start
Breakpoint 1 at 0x4000b0: file conditionals.s, line 11.
(gdb) b done
Breakpoint 2 at 0x40014c: file conditionals.s, line 76.
(gdb) c
Continuing.
Breakpoint 2, done () at conditionals.s:76
=> 0x000000000000014c <done+0>: 00 00 80 d2 mov x0, #0x0 // #0
(gdb) x/s $x1
x1: "It is a right triangle."
(gdb)

```

To make sure the output was correct, I used the `x/s $x1` command to check the value of the `x1` register, which prints out **“It is a right triangle.”**

Lab #5 With Custom Points (0,4) (1,2) (3,2)

```

Register group: general
x0 0x410158 4260184 x1 0x4101a0 4260256 x2 0x410178 4260216
x3 0x0 0 x4 0x410160 4260192 x5 0x0 0
x6 0x1 1 x7 0x410170 4260208 x8 0x0 0
x9 0x3 3 x10 0x410180 4260224 x11 0x3 3
x12 0x0 0 x13 0x1 1 x14 0x0 0
x15 0x1 1 x16 0x1 1 x17 0x1 1
x18 0x0 0 x19 0x1 1 x20 0x2 2
x21 0x2 2 x22 0x2 2 x23 0x1 1
x24 0x1 1 x25 0xffffffffffffff -1 x26 0x0 0
x27 0x0 0 x28 0x0 0 x29 0x0 0
x30 0x0 0 sp 0x55008001b0 0x55008001b0 pc 0x40014c <done>
cpsr 0x40000000 1073741824 fpsr 0x0 0 fpcr 0x0 0

70
71 //Code is executed if legs are equal to hypotenuse
72 equal: ADR X1, yes //Outputs yes label
73
74
75
B+> 76 Done: MOV X0, #0 //Exits the program
77 MOV X8, #93
78 SVC #0
79
80
81 .data
82 p1: quad 0, 4
83 p2: quad 1, 2

remote Thread 1.4495 In: done
(gdb) b _start
Breakpoint 1 at 0x4000b0: file conditionals.s, line 11.
(gdb) b done
Breakpoint 2 at 0x40014c: file conditionals.s, line 76.
(gdb) c
Continuing.
Breakpoint 2, done () at conditionals.s:76
=> 0x000000000000014c <done+0>: 00 00 80 d2 mov x0, #0x0 // #0
(gdb) x/s 4260256
x1: "It is not a right triangle."
(gdb)

```

I ran the same commands as with the first potential triangle coordinates. First, I set the breakpoints at the start and at the label that exits my program: **b _start** and **b done**. I then used the continue command to jump to the end of my program: **c**. I then used the same printing string

command, `x/s`, with the address value presented in the terminal, **4260256**. As expected, my triangle is not a triangle.

```
Register group: general
x0      0x410158      4260184      x1      0x4101a0      4260256      x2      0x410178      4260216
x3      0x0          0          x4      0x410160      4260192      x5      0x0          0
x6      0x1          1          x7      0x410170      4260208      x8      0x0          0
x9      0x3          3          x10     0x410180      4260224      x11     0x3          3
x12     0x0          0          x13     0x1          1          x14     0x0          0
x15     0x1          1          x16     0x1          1          x17     0x1          1
x18     0x0          0          x19     0x1          1          x20     0x2          2
x21     0x2          2          x22     0x2          2          x23     0x1          1
x24     0x1          1          x25     0xffffffff    -1         x26     0x0          0
x27     0x0          0          x28     0x0          0          x29     0x0          0
x30     0x0          0          sp      0x55008001b0    0x55008001b0 pc      0x40014c    0x40014c <done>
cpsr    0x40000000    1073741824    fpsr    0x0          0          fpcr    0x0          0

conditional.s
72 equal: ADR X1, yes //Outputs yes label
73
74
75
B>> 76 done: MOV X0, #0 //Exits the program
77
78      MOV X8, #33
79      SVC #0
80
81 .data
82 pl: quad 0, 4
83 x1: quad 1, 2

remote Thread 1.4495 In: done
Breakpoint 1 at 0x400000: file conditional.s, line 11.
(gdb) b done
Breakpoint 2 at 0x40014c: file conditional.s, line 76.
(gdb) c
Continuing.

Breakpoint 2, done () at conditional.s:76
=> 0x0000000000000000: <done=0>: 00 00 80 d2 mov x0, #0x0 // #0
(gdb) x/s 4260256
0x0000000000000000: "It is not a right triangle."
(gdb) x $X1
Value can't be converted to integer.
(gdb) x/s $X1
Value can't be converted to integer.
(gdb) x/s $X1
0x0000000000000000: "It is not a right triangle."
(gdb)
```

To make sure the output was correct, I used the `x/s $x1` command to check the value of the `x1` register, which prints out **“It is not a right triangle.”**