Zach Dubinsky

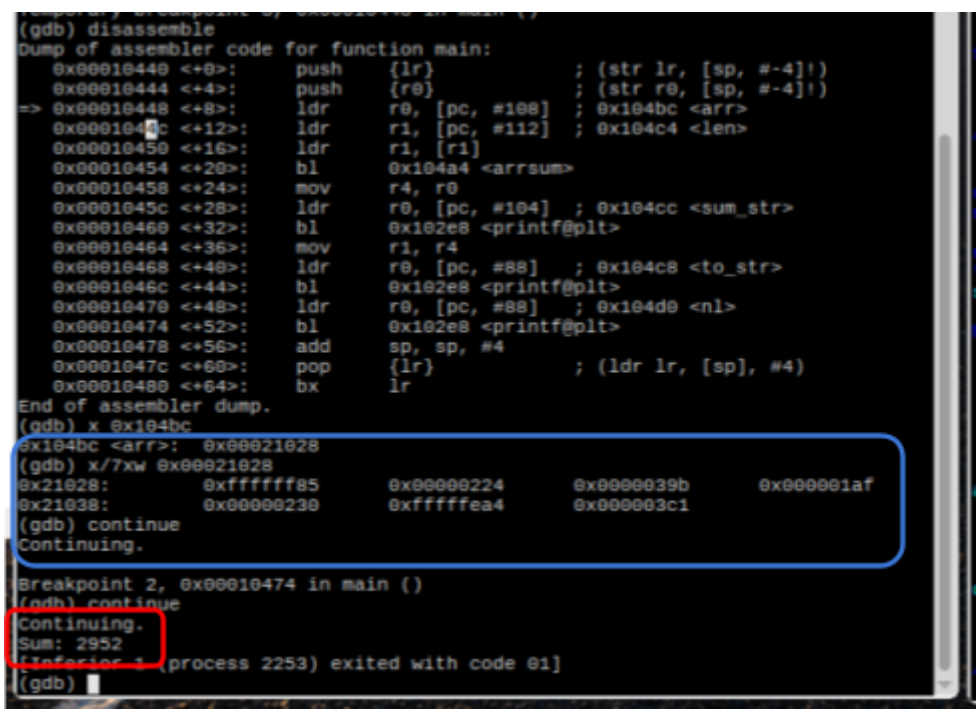
CSC-270-01

Prof. Rieffel

2/17/22

Lab 6

1. My array sum algorithm is designed in the same way as in my original Lab 6 submission. However the code outside of the algorithm is much cleaner. The one change I made was to have data memory calculate the length of the list using C-ish operations that ARM provides. ARM offers a great deal of pseudo-instructions that MIPS does not provide. I have included a screenshot of the GDB debugger outputting the contents of data memory at the address of the array (in blue). The debugger also shows the output to the console (in red).



```
(gdb) disassemble
Dump of assembler code for function main:
   0x00010440 <+0>:     push    {lr}            ; (str lr, [sp, #-4]!)
   0x00010444 <+4>:     push    {r0}            ; (str r0, [sp, #-4]!)
=> 0x00010448 <+8>:     ldr     r0, [pc, #108]  ; 0x104bc <arr>
   0x0001044c <+12>:    ldr     r1, [pc, #112]  ; 0x104c4 <len>
   0x00010450 <+16>:    ldr     r1, [r1]
   0x00010454 <+20>:    bl      0x104a4 <arrsum>
   0x00010458 <+24>:    mov     r4, r0
   0x0001045c <+28>:    ldr     r0, [pc, #104]  ; 0x104cc <sum_str>
   0x00010460 <+32>:    bl      0x102e8 <printf@plt>
   0x00010464 <+36>:    mov     r1, r4
   0x00010468 <+40>:    ldr     r0, [pc, #88]   ; 0x104c8 <to_str>
   0x0001046c <+44>:    bl      0x102e8 <printf@plt>
   0x00010470 <+48>:    ldr     r0, [pc, #88]   ; 0x104d0 <nl>
   0x00010474 <+52>:    bl      0x102e8 <printf@plt>
   0x00010478 <+56>:    add     sp, sp, #4
   0x0001047c <+60>:    pop     {lr}            ; (ldr lr, [sp], #4)
   0x00010480 <+64>:    bx      lr
End of assembler dump.
(gdb) x 0x104bc
0x104bc <arr>:  0x00021028
(gdb) x/7xw 0x00021028
0x21028:        0xffffff85      0x00000224      0x0000039b      0x000001af
0x21038:        0x00000230      0xffffffea4     0x000003c1
(gdb) continue
Continuing.

Breakpoint 2, 0x00010474 in main ()
(gdb) continue
Continuing.
Sum: 2952
[Inferior 1 (process 2253) exited with code 01]
(gdb)
```

2. My insertion sort algorithm relies on pointer arithmetic. It is designed in the same way as my MIPS version. The following image shows the GDB debugger with outputs of data memory at the address of the array before and after the sort (in red). The outputs of the insertion sort program to the console are also present (in blue).

```
Temporary breakpoint 1, 0x00010448 in main ()
(gdb) disassemble
Dump of assembler code for function main:
   0x00010440 <+0>:     push    {lr}            ; (str lr, [sp, #-4]!)
   0x00010444 <+4>:     push    {r0}            ; (str r0, [sp, #-4]!)
=> 0x00010448 <+8>:     ldr     r4, [pc, #264]  ; 0x10558 <arr>
   0x0001044c <+12>:    ldr     r5, [pc, #264]  ; 0x1055c <len>
   0x00010450 <+16>:    ldr     r5, [r5]
   0x00010454 <+20>:    ldr     r0, [pc, #276]  ; 0x10570 <unsorted>
   0x00010458 <+24>:    bl      0x102e8 <printf@plt>
   0x0001045c <+28>:    mov     r0, r4
   0x00010460 <+32>:    mov     r1, r5
   0x00010464 <+36>:    bl      0x10494 <pr_arr>
   0x00010468 <+40>:    mov     r0, r4
   0x0001046c <+44>:    mov     r1, r5
   0x00010470 <+48>:    bl      0x10500 <i_sort>
   0x00010474 <+52>:    ldr     r0, [pc, #240]  ; 0x1056c <sorted>
   0x00010478 <+56>:    bl      0x102e8 <printf@plt>
   0x0001047c <+60>:    mov     r0, r4
   0x00010480 <+64>:    mov     r1, r5
   0x00010484 <+68>:    bl      0x10494 <pr_arr>
   0x00010488 <+72>:    add     sp, sp, #4
   0x0001048c <+76>:    pop     {lr}            ; (ldr lr, [sp], #4)
   0x00010490 <+80>:    bx      lr
End of assembler dump.
(gdb) x 0x10558
0x10558 <arr>:  0x00021028
(gdb) x/10xw 0x00021028
0x21028:        0x00000002      0x00000008      0xfffffff7      0x00000004
0x21038:        0x0000000a      0x00000011      0x00000020      0x000003f1
0x21048:        0xffffff90      0x0000001e
(gdb) break 0x10474
Function "0x10474" not defined.
Make breakpoint pending on future shared library load? (y or [n]) n
(gdb) break *0x10474
Breakpoint 2 at 0x10474
(gdb) continue
Continuing.
Unsorted List: 2,8,-9,4,10,17,32,1009,-112,30

Breakpoint 2, 0x00010474 in main ()
(gdb) x/10xw 0x00021028
0x21028:        0xffffff90      0xfffffff7      0x00000002      0x00000004
0x21038:        0x00000008      0x0000000a      0x00000011      0x0000001e
0x21048:        0x00000020      0x000003f1
(gdb) continue
Continuing.
Sorted List: -112,-9,2,4,8,10,17,30,32,1009
[Inferior 1 (process 2225) exited with code 01]
(gdb)
```