

Angel Velazquez
CST-307
Dr. Citro

Introducing QtSPIM

<https://github.com/angel-vlzqz/Computer-Architecture> - github link to CST-307 repo

Installing QtSPIM was a simple procedure. The directions were straightforward, I did not have any difficulties downloading QtSPIM. However, I did have to override Apple's default security parameters because I was downloading an application from the web. My OS read it as "unverified". I assisted other classmates running MacOS to help them override the same security issues.

The test programs needed to be clearer to run. I wasn't sure how to load or execute a test program, but I simply gave the application the path to find the test files on my system, and understanding that made the learning curve easier. When I would run the programs, I would notice the registers on the left-hand side turn red, which meant that their states were altered during the execution of the Assembly instructions.

I experimented with single stepping and the breakpoint feature within QtSPIM and found them incredibly helpful in looking at what the registers are filled with at every step.

Successful Installation

The screenshot shows the QtSpim simulator interface with the title "IntroducingQTSPIM". The assembly code has been successfully executed, and the results are displayed in the registers and memory sections. The assembly code includes various instructions such as addu, mul, and syscall, along with comments explaining their functions. The registers section shows the state of the processor after execution, and the memory section shows the contents of memory locations. The status bar at the bottom indicates the current git commit: "git:(main) Type '#' for AI command suggestions".

```

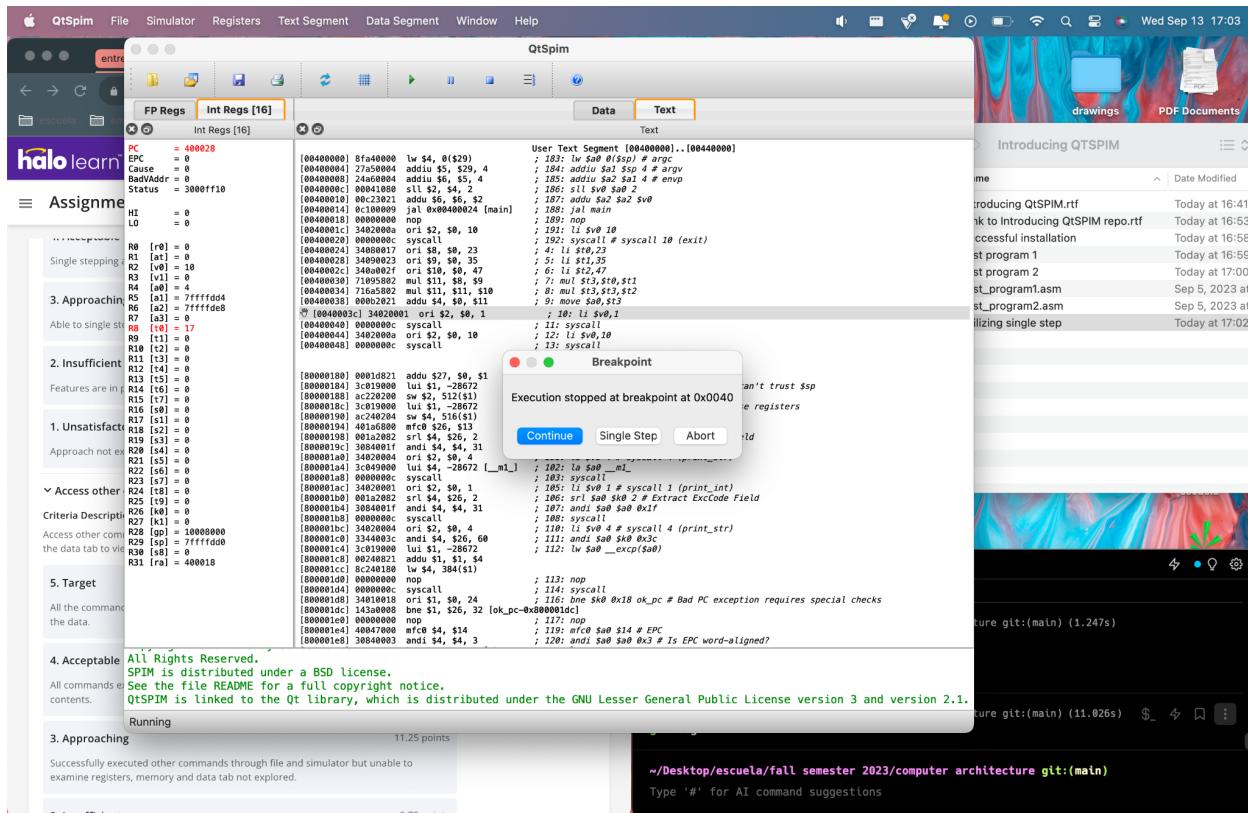
    User Text Segment [0x04000000]..[0x04400000]
    ; 183: lv $v0 $t0 $sp # argc
    ; 184: addiu $a1 $sp 4 # argv
    ; 185: addiu $a2 $sp 8 # envp
    ; 186: sll $v0 $a2 2
    ; 187: addu $a2 $a2 $v0 2
    ; 188: jal main
    ; 189: li $v0 10
    ; 190: li $v0 10
    ; 192: syscall # syscall 10 (exit)
    ; 21: li $v0 10
    ; 5: li $t2,35
    ; 6: li $t2,47
    ; 7: mul $t3,$t2,$t2
    ; 8: addu $t3,$t3,$t2
    ; 9: move $a0,$t3
    ; 10: li $v0,1
    ; 11: la $a0,_m1_
    ; 12: li $v0,10
    ; 13: syscall

    Kernel Text Segment [0x00000000]..[0x00100000]
    ; 90: move $k1,$at # Save $at
    ; 92: sw $v0 $t1 4 # Not re-entrant and we can't trust $sp
    ; 93: sw $a2 2 # But we need to use these registers
    ; 95: mfc0 $t0 $t0 # Cause register
    ; 96: mfc0 $t0 $t0 # Extract ExcCode Field
    ; 97: andi $a0 $a0 0x1f
    ; 101: li $v0 4 # syscall 4 (print_int)
    ; 102: la $a0,_m1_
    ; 103: li $v0 4 # syscall 4 (print_int)
    ; 105: li $v0 1 # syscall 1 (print_int)
    ; 106: srl $a0 $a0 2 # Extract ExcCode Field
    ; 107: andi $a0 $a0 0x1f
    ; 108: mfc0 $t0 $t0
    ; 110: li $v0 4 # syscall 4 (print_int)
    ; 111: andi $a0 $a0 0x3c
    ; 122: lw $a0 _exc($a0)

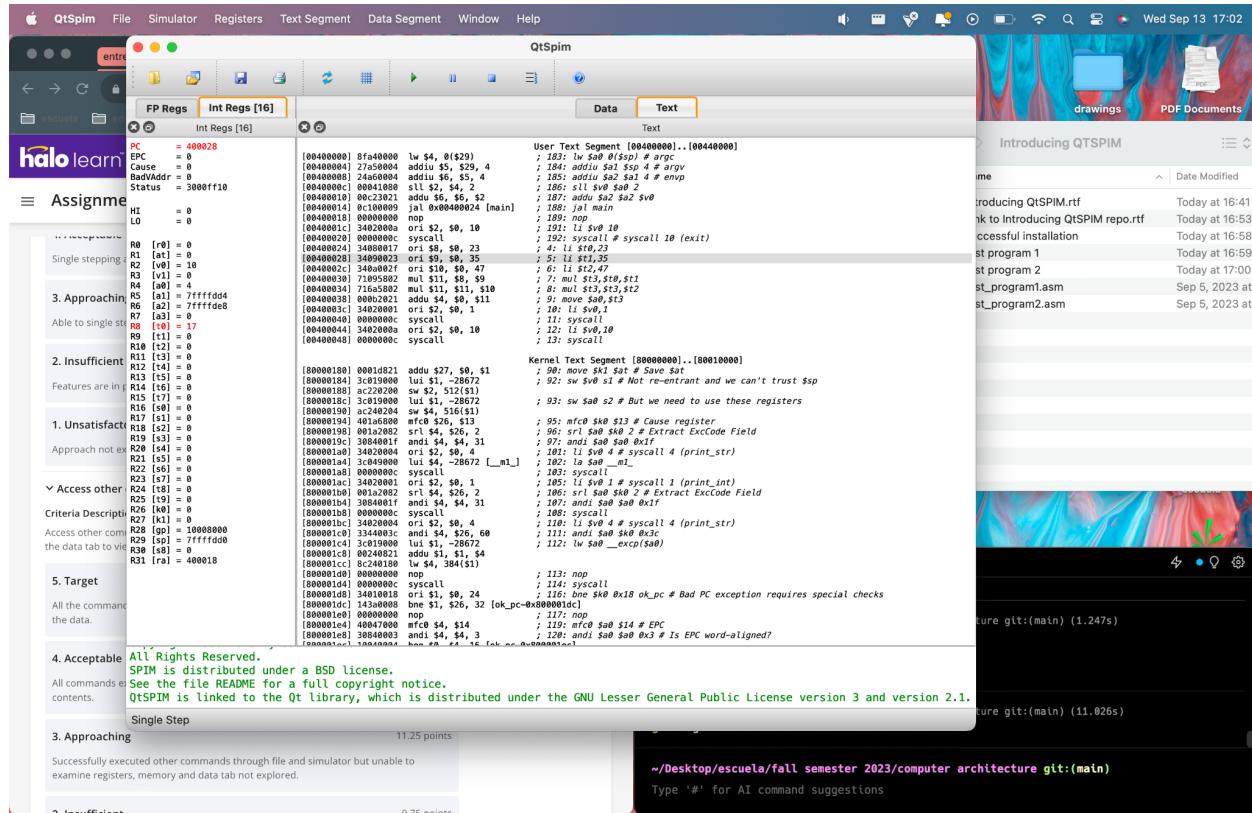
    Memory and registers cleared

```

Utilizing Breakpoints



Utilizing Single Step



Angel Velazquez

CST-307

Dr. Citro

Test Program 1

The screenshot shows the QtSpim interface with the assembly code for Test Program 1. The assembly code is displayed in the Text Segment window, showing instructions from memory addresses 0x00000000 to 0x00440000. The code includes various arithmetic operations (addiu, add, mul, and), jumps (jal), and system calls (syscall). Registers are monitored at the bottom left, and a file browser is visible on the right.

```
00000000 00000000 lw $4, 0($20) ; 182: lw $4, 0($20)
00000004 27a50004 addiu $5, $29, 4 ; 184: addiu $5, $29, 4
00000008 24a60004 addiu $6, $5, 4 ; 185: addiu $6, $5, 4
0000000c 00041000 sll $2, $4, 2 ; 186: sll $2, $4, 2
00000010 00041000 sll $3, $6, 2 ; 187: sll $3, $6, 2
00000014 0c100009 jal 0x00400024 [main] ; 188: jal main
00000018 00000000 nop ; 189: nop
00000020 00000000 ori $2, $0, 10 ; 190: ori $2, $0, 10
00000024 0000000c syscall ; 191: syscall
00000028 34000007 ori $8, $0, 23 ; 4: li $8, 23
0000002c 34090023 ori $9, $0, 35 ; 5: li $9, 35
00000030 34090024 ori $10, $0, 47 ; 6: li $10, 47
00000034 71095002 mul $11, $8, $9 ; 7: mul $11, $8, $9
00000038 71095002 mul $11, $11, $10 ; 8: mul $11, $11, $10
0000003c 00041002 add $14, $8, $11 ; 9: add $14, $8, $11
00000040 00041002 add $15, $9, 1 ; 10: add $15, $9, 1
00000044 0000000c syscall ; 11: syscall
00000048 0000000c syscall ; 12: syscall
00000100 0000000c syscall ; 13: syscall
00000120 00010021 addu $27, $9, $1 ; 90: move $27, $9
00000124 3c010000 lui $1, -28672 ; 92: sw $v0, $1
00000128 ac220000 sw $2, $12($1) ; 93: sw $v0, $1
00000132 00010020 lui $1, -28672 ; 93: sw $v0, $2
00000136 3c010000 lui $1, -28672 ; 93: sw $v0, $2
0000013a 0001000f and $4, $26, $13 ; 95: mfc0 $v0, $4
0000013e 0001000f and $4, $26, $13 ; 96: srl $v0, $4
00000142 001a2002 srl $4, $26, 2 ; 97: andi $v0, $4
00000146 30840010 andi $4, $4, 31 ; 97: andi $v0, $4
0000014a 30840011 andi $4, $4, 31 ; 98: andi $v0, $4
00000154 0000000c syscall ; 99: syscall
00000158 3c045000 lui $1, -28672 [._mtl_] ; 102: la $a0, _mtl_
00000162 0000000c syscall ; 103: syscall
00000166 34020000 ori $2, $0, 1 ; 104: li $2, 1
00000170 34020000 ori $2, $0, 2 ; 106: li $2, 2
00000174 3084001f andi $4, $4, 31 ; 107: andi $v0, $4
00000178 0000000c syscall ; 108: syscall
00000182 0000000c syscall ; 109: syscall
00000186 33440003 andi $4, $26, 60 ; 110: andi $v0, $4
00000190 3c010000 lui $1, -28672 ; 111: andi $v0, $4
00000194 00240021 addu $1, $1, $4 ; 112: lw $v0, _-
00000198 0000000c syscall ; 113: syscall
0000019c 0000000c syscall ; 114: syscall
000001dc 34020000 ori $1, $0, 24 ; 116: mfc0 $v0, $0
000001e0 3c010000 lui $1, $26, 32 [ok_pc=0x80000000] ; 117: mfc0 $v0, $0
000001e4 00000000 nop ; 117: nop
000001e8 40047000 mfc0 $4, $14 ; 119: mfc0 $v0, $4
000001f2 30840003 andi $4, $4, 3 ; 120: andi $v0, $4
000001f6 30840003 andi $4, $4, 3 ; 120: andi $v0, $4
000001f0 3c010000 lui $1, -28672 [ok_nok=0x00000000] ; 121: la $a0, ok_nok
```

All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 or later.

File: QtSpim - Test Program 1

File Path: /Desktop/estudia/Fall Semester 2023/Computer Architecture/gctt/match

Type '#' for AI command suggestions

Test Program 2

The screenshot shows the QtSpim debugger interface with the following details:

- Registers:**
 - PC = 400854
 - EPC = 0
 - Cause = 0
 - BadVaddr = 0
 - Status = 3000ff10
 - HE = 0
 - LO = 60d8
 - R0 [r0] = 0
 - R1 [r1] = 10010000
 - R2 [r2] = 0
 - R3 [r3] = 0
 - R4 [r4] = 60d8
 - R5 [r5] = 7fffffe00
 - R6 [r6] = 7fffffe00
 - R7 [r7] = 0
 - R8 [r8] = 11
 - R9 [r9] = 10
 - R10 [r10] = 38
 - R11 [r11] = 60d8
 - R12 [r12] = 0
 - R13 [r13] = 0
 - R14 [r14] = 0
 - R15 [r15] = 0
 - R16 [r16] = 0
 - R17 [r17] = 0
 - R18 [r18] = 0
 - R19 [r19] = 0
 - R20 [r20] = 0
 - R21 [r21] = 0
 - R22 [r22] = 0
 - R23 [r23] = 0
 - R24 [r24] = 0
 - R25 [r25] = 0
 - R26 [r26] = 0
 - R27 [r27] = 0
 - R28 [r28] = 10000000
 - R29 [r29] = 7fffffdc
 - R30 [r30] = 0
 - R31 [r31] = 400018
- Data Segment:** User Text Segment [00400000]..[00440000]


```
00400000 8f4d0000 lw $4, 0($20) ; 182: lw $v0 $t3($sp) # argc
[...]
00400004 27a50004 addiu $5, $29, 4 ; 184: addiu $a2 $a1 4 # envp
00400008 24a60004 addiu $6, $29, 4 ; 185: addiu $a2 $a1 4 # envp
0040000c 00041080 sll $2, $4, $2 ; 186: sll $v0 $a0 2
00400010 00041080 sll $3, $4, $2 ; 187: sll $a0 $a2 $a2 $v0
00400014 00041080 jal 0x00400024 {main} ; 188: jal main
00400018 00000000 nop ; 189: nop
[...]
```
- Text Segment:** Kernel Text Segment [00000000]..[00000000]


```
00000180 00010821 addu $27, $0, $1 ; 90: move $k1 $s1
[...]
00000184 3c010800 lui $1, -28672 ; 92: sw $v0 $s1
00000188 3c010800 lui $1, -28672 ; 93: sw $s0 $s2
0000018c 3c010800 lui $1, -28672 ; 93: sw $s0 $s2
[...]
00000198 ac240204 sw $4, 516($1) ; 95: mfc0 $v0 $s
0000019c 401a0800 mfc0 $26, $13 ; 96: srl $v0 $s
000001a0 00020201 addu $4, $0, $11 ; 97: andi $s0 $s
000001a4 34020001 ori $2, $0, 1 ; 101: li $v0 1
000001a8 34020001 ori $2, $0, 1 ; 101: li $v0 1
000001ac 34020001 ori $2, $0, 1 ; 101: li $v0 1
000001e0 00000000 lui $1, -28672 {__m1} ; 102: andi $s0 $s
000001e4 00000000 lui $1, -28672 {__m1} ; 103: syscall
[...]
000001f8 34020001 ori $2, $0, 1 ; 105: li $v0 1
000001fc 001a2082 srl $4, $26, 2 ; 106: srl $v0 $s
00000200 00000000 andi $4, $4, $1 ; 107: andi $s0 $s
00000204 00000000 syscall ; 108: syscall
[...]
0000020c 34020001 ori $2, $0, 4 ; 110: li $v0 4
00000210 34020001 ori $2, $0, 4 ; 110: li $v0 4
00000214 33000000 and $4, $4, $60 ; 111: andi $s0 $s
00000218 00000000 lui $1, -28672 ; 112: lw $a0 -4
0000021c 00000000 lui $1, $1, $4 ; 113: nop
[...]
00000240 34020001 lw $4, 384($1) ; 113: syscall
00000244 00000000 syscall ; 114: syscall
[...]
00000248 34010018 ori $1, $0, 24 ; 116: bne $k0 $s
0000024c 143a0000 bne $1, $26, 32 {ok_pc-0x0000001d}
[...]
```
- File Browser:** Shows files like boot.s, boot.o, boot.out, and boot.rtf.
- Console:** Displays assembly code and comments.