Lab 3

Preparation

```
Alex Griffiths
18001525
```

Question 2

a

The first instruction of a user's code is stored in 0x00400024. You can recognize the user's first instruction as it starts after the `main:` label. This label must be a global label.

b

```
       "H"             "a"               "v"
HEX    48              61                76
BINARY 0100 1000       0110 0001         0111 0110
```

Workshop Tasks

Question 2

After altering `simplecalc.s` to include the formula `x = (g + h) - (i - j + k)`, the expected result of the formula should be 4. I have set the values of each value as follows:

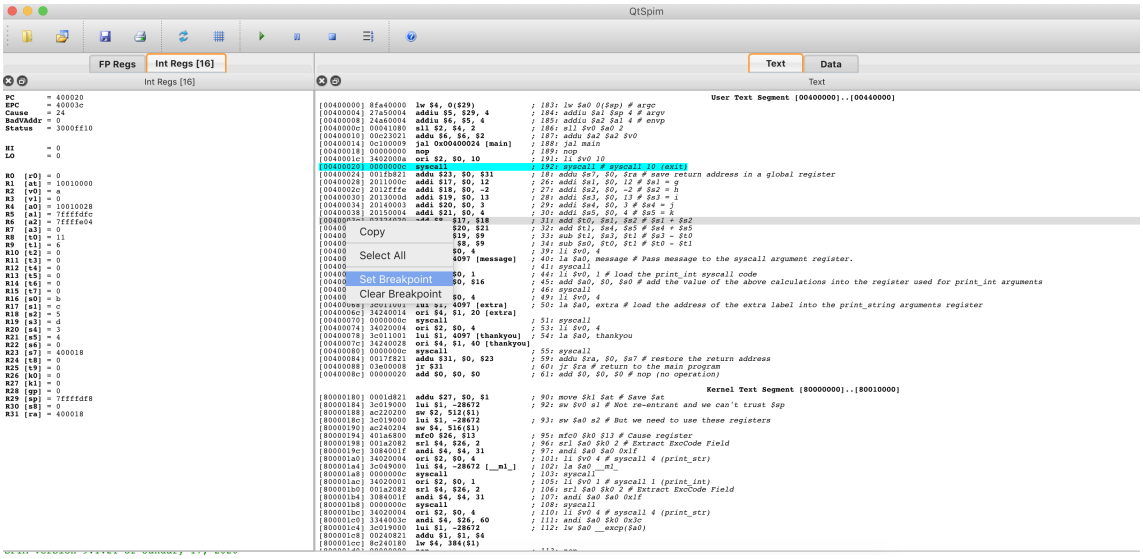
```
g = 12
h = -2
i = 13
j = 3
k = 4
```

This resolves to:

```
x = (12 + (-2)) - (13 - 3 + 4)
  = 10 - (13 - 7)
  = 10 - 6
  = 4
```

Breakpoint after initial values are set

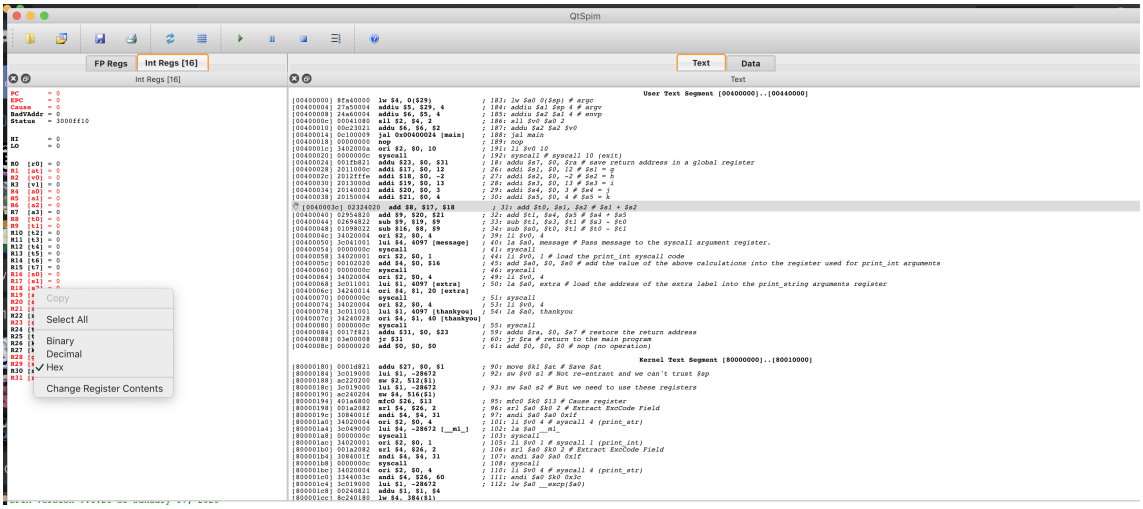
You can see the line selected comes after the intial values are set (Please ignore the light blue line. I do not know why it is highlighted, but the breakpoint is not inserted on this line)



Choosing & setting a new value in the register

Here I'm selecting the `$s2` register to alter by clicking `Change Register Contents`

You can also see a hand icon next to the line where the breakpoint is inserted.



Here I am assigning a new value (`5`) to the register. I have the `Hexadecimal` radio button selected, which for this value is fine because 5 in Decimal is represented the same way as in Hexadecimal.

QtSpim

FP Regs | Int Regs [16]

Int Regs [16]

```
PC      = 0
EPC     = 0
Cause   = 0
BadVAddr = 0
Status  = 3000ff10

HI      = 0
LO      = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 0
R3  [v1] = 0
R4  [a0] = 0
R5  [a1] = 0
R6  [a2] = 0
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 0
R29 [sp] = 7fffeffc
R30 [s8] = 0
R31 [ra] = 0
```

Text | Data

Text

```
                                                User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (exit)
[00400024] 001fb821  addu $23, $0, $31          ; 18: addu $s7, $0, $ra # save return address in a global register
[00400028] 2011000c  addi $17, $0, 12           ; 26: addi $s1, $0, 12 # $s1 = g
[0040002c] 2012fffe  addi $18, $0, -2           ; 27: addi $s2, $0, -2 # $s2 = h
[00400030] 2013000d  addi $19, $0, 13           ; 28: addi $s3, $0, 13 # $s3 = i
[00400034] 20140003  addi $20, $0, 3            ; 29: addi $s4, $0, 3 # $s4 = j
[00400038] 20150004  addi $21, $0, 4            ; 30: addi $s5, $0, 4 # $s5 = k
[0040003c] 02324020  add $8, $17, $18           ; 31: add $t0, $s1, $s2 # $s1 + $s2
[00400040] 02954820  add $9, $20, $21           ; 32: add $t1, $s4, $s5 # $s4 + $s5
[00400044] 02694822  sub $9, $19, $9            ; 33: sub $t1, $s3, $t1 # $s3 - $t0
[00400048] 01098022  sub $16, $8, $9            ; 34: sub $s0, $t0, $t1 # $t0 - $t1
[0040004c] 34020004  ori $2, $0, 4              ; 39: li $v0, 4
[00400050] 3c041001  lui $4, 4097 [message]     ; 40: la $a0, message # Pass message to the syscall argument register.
[00400054] 0000000c  syscall                    ...syscall code
[00400058] 34020001  ori $2, $0, 1              ...of the above calculations into the register used for print_int arguments
[0040005c] 00102020  add $4, $0, $16
[00400060] 0000000c  syscall
[00400064] 34020004  ori $2, $0, 4              ...of the extra label into the print_string arguments register
[00400068] 3c011001  lui $1, 4097 [extra]
[0040006c] 34240014  ori $4, $1, 20 [extra]
[00400070] 0000000c  syscall
[00400074] 34020004  ori $2, $0, 4
[00400078] 3c011001  lui $1, 4097 [thankyou]
[0040007c] 34240028  ori $4, $1, 40 [thankyo
[00400080] 0000000c  syscall
[00400084] 001f8821  addu $31, $0, $23          ...return address
[00400088] 03e00008  jr $31                     ...gram
[0040008c] 00000020  add $0, $0, $0             ...on)

                                                Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not re-entrant and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0 s2 # But we need to use these registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13              ; 95: mfc0 $k0 $13 # Cause register
[80000198] 001a2082  srl $4, $26, 2             ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[8000019c] 3084001f  andi $4, $4, 31            ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4              ; 101: li $v0 4 # syscall 4 (print_str)
[800001a4] 3c049000  lui $4, -28672 [__m1_]     ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                    ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1              ; 105: li $v0 1 # syscall 1 (print_int)
[800001b0] 001a2082  srl $4, $26, 2             ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001b4] 3084001f  andi $4, $4, 31            ; 107: andi $a0 $a0 0x1f
[800001b8] 0000000c  syscall                    ; 108: syscall
[800001bc] 34020004  ori $2, $0, 4              ; 110: li $v0 4 # syscall 4 (print_str)
[800001c0] 3344003c  andi $4, $26, 60           ; 111: andi $a0 $k0 0x3c
[800001c4] 3c019000  lui $1, -28672             ; 112: lw $a0 __excp($a0)
[800001c8] 00240821  addu $1, $1, $4
[800001cc] 8c240180  lw $4, 384($1)
```

Change Value

New value for R18

5

Hexadecimal Decimal

Cancel OK

Altered Registers

You can see in register 18 (the register for `$s2`) that the value 5 is set.

```
R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 4
R3  [v1] = 0
R4  [a0] = 1
R5  [a1] = 7ffffdfc
R6  [a2] = 7ffffe04
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = c
R18 [s2] = 5
R19 [s3] = d
R20 [s4] = 3
R21 [s5] = 4
R22 [s6] = 0
R23 [s7] = 400018
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 0
R29 [sp] = 7ffffdf8
R30 [s8] = 0
R31 [ra] = 400018
```

The result after changing the value in the register

The above image shows the output of the program, and specifically, the first line contains the calculated value of the new formula, which written out would look like:

```
x = (12 + 5) - (13 - 3 + 4)
  = 17 - (13 - 7)
  = 17 - 6
  = 11
```

Question 3

Memory Segments holding the message "Have a nice day". The highlighted sections are the words containing the hex codes of the string.

The way this data is loaded in to memory address looks like this:

```
 ADDRESS     HEX
|1001002F|y |79
|1001002E|a |61
|1001002D|d |64
|1001002C|  |20
|1001002B|e |65
|1001002A|c |63
|10010029|i |69
|10010028|n |6E
|10010027|  |20
|10010026|a |61
|10010025|  |20
|10010024|e |65
|10010023|v |76
|10010022|a |61
|10010021|H |48
|10010020|LF|0A
|  ...   |  |
|10010000|  |
 ^^^^^^^^
 Base Addr
```

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    20656854  756c6176  666f2065  69206620    T h e   v a l u e   o f   f   i
[10010010]    00203a73  7661480a  20612065  6563696e    s :   . . H a v e   a   n i c e
[10010020]    79616420  00293a20  200a0a0a  202e2e2e      d a y   : ) . . . .   . . .
[10010030]    6e616854  6f79206b  293a2075  00000000    T h a n k   y o u   : ) . . . .
[10010040]..[1003ffff]  00000000
```

76	61	48	0a	20	61	20	65	65	63	69	6e
v	a	H	L>		a		e	e	c	i	n

79	61	64	20
y	a	d	

It might not be clear in the image above, but each word is 4 characters long, as each character is a byte, and a word is 32 bits long. As a byte is 4 characters, we can only have 4 characters per word.

Each word is packed with the hex code for first character in the least significant bit. When the word is full, the next hex code for the ASCII character is stored in the least significant bit of the next word. This leads to a strange memory layout which when blocked out looks like the values in the table I have drawn.

After altering the data segment that is loaded by QtSpim

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    20656854  756c6176  666f2065  69206620    T h e   v a l u e   o f   f   i
[10010010]    00203a73  6f6e2d0a  20612074  6563696e    s :   . . - n o t   a   n i c e
[10010020]    79616420  00293a20  200a0a0a  202e2e2e      d a y   : ) . . . .   . . .
[10010030]    6e616854  6f79206b  293a2075  00000000    T h a n k   y o u   : ) . . . .
[10010040]..[1003ffff]  00000000
```

Output to console after altering the data segment

```
The value of f is: 4
-not a nice day :)


 ... Thank you :)
```

Question 4

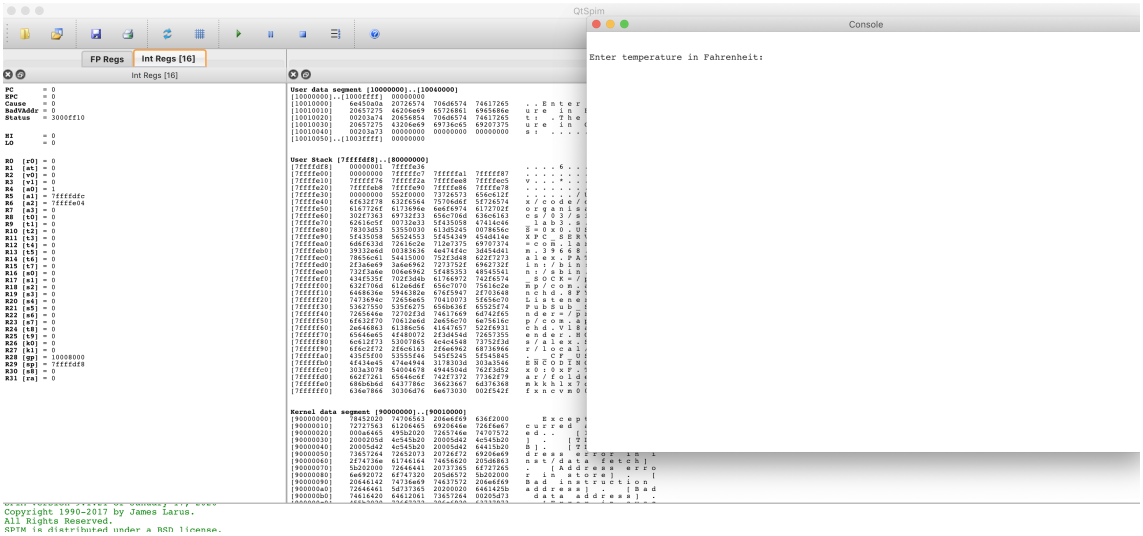
We load data from the user input by loading the syscall code `5` into `$v0`. Calling the `syscall` instruction will then wait for the user's input. When it receives the user's input, it will store the value in `$v0`, which we then move to `$t0` to maintain appropriate use of registers following convention.

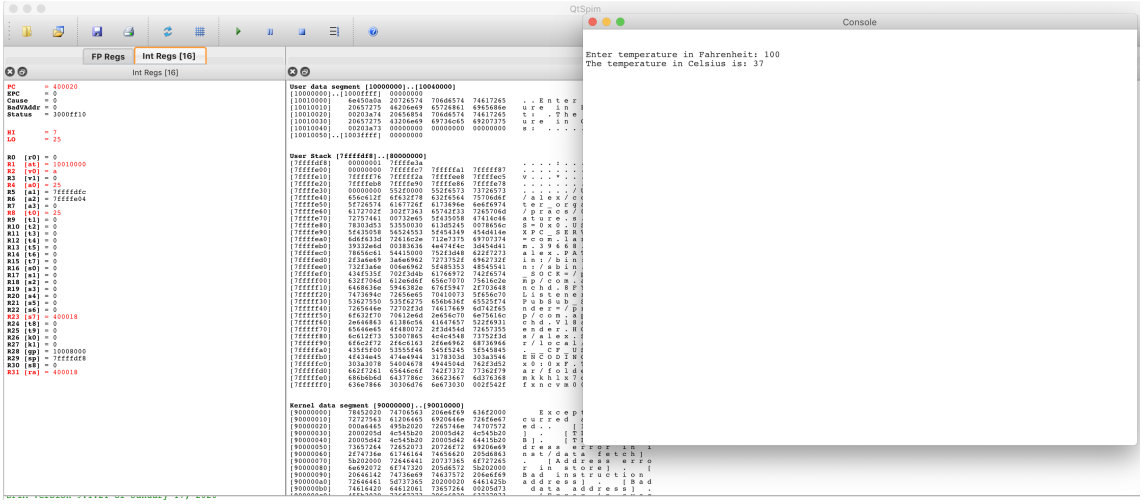
The formula that was used to convert temperature from Farenheit to Celsius is

```
C = (5 * (x - 32)) / 9

x = user input
```

Screenshots of the program running




QtSpim
FP Regs
Int Regs [16]
Int Regs [16]
PC = 400020
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 7
LO = 25
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = a
R3 [v1] = 0
R4 [a0] = 25
R5 [a1] = 7effffdfc
R6 [a2] = 7effffe64
R7 [a3] = 0
R8 [t0] = 25
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 400018
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7effffdf8
R30 [s8] = 0
R31 [ra] = 400018
User data segment [10000000]..[10040000]
User Stack [7effffdf8]..[80000000]
Kernel data segment [90000000]..[90010000]
Console
Enter temperature in Fahrenheit: 100
The temperature in Celsius is: 37