BBM234: Computer Organization
MIPS Project Report

Canda Nasf, 21328232

March 25, 2018

1 Problem: Arrays Using for Loops

MIPS Assembly code for array.asm:

```
1     .data
2     A: .word 2, 4, 6, 8    # definition of array
3     .text
4     main:
5     la $t1, A          #load the array in $t1
6     addi $s0, $s0, 0   # s0 = 0 (i = 0)
7     addi $s1, $s1, 3   # s1 = 3 length of array
8     for:
9     beq $s0, $s1, done  # loop will continue if s0 != s1 if they are equal
          jump "done"
10    lw   $t2, 0($t1)    # load the element of array in t2 (tw = a[i])
11    lw $t3, 4($t1)      # load the element of array in t3 (tw = a[i+1])
12    sub $s3, $t3, $t2   # s3 will keep the difference of elements (diff = a[i
          +1] - a[i])
13    slt $s4, $0, $s3
14    beq $s4, 1, multiplication  # If difference greater than zero then jump "
          multiplication"
15    sll $s5, $t2, 2             # If difference greater than zero then s5= 4 *
           t2
16    add $t3, $t2, $s5           # t3 = t2 + s5 so t3 = 5 * t2
17    sub  $t3, $0, $t3           # take negative of t3
18    sw $t3, 4($t1)              # assign in array a[i+1] = -5*a[i]
19    addi $s0, $s0, 1            # i = i + 1
20    addi $t1, $t1, 4            # t1 = t1 + 4 for next element of array.We
          increase the address
21    j for                       # jump the for loop
22
23    multiplication:
24    sll $s5, $t2, 2                 # s5 = 4 * t2
25    add $s5, $t2, $s5               # s5 = s5 + t2 so s5 = 5 * t2
26    sw $s5, 0($t1)                  # assign in array a[i] = 5 * a[i]
27    addi $s0, $s0, 1                # i = i + 1
28    addi $t1, $t1, 4                # t1 = t1 + 4 for next element of array.We
           increase the address
```

```
29      j for                           # jump the for loop
30
31      done :
32      lw $t4, -12($t1)                #If you want to see elements of array I
            put them in registers (t4 - t7)
33      lw $t5, -8($t1)
34      lw $t6, -4($t1)
35      lw $t7, 0($t1)
36      li $v0, 10  # end of the code
37      syscall
38      .end
```

Test 1: A=2,4,6,8
Test 2: A=8,6,4,2
Test 3: A=2,2,6,4
Results for Test 1: A=2,4,6,8

before run: Fig. 1.
after run: Fig. 2.

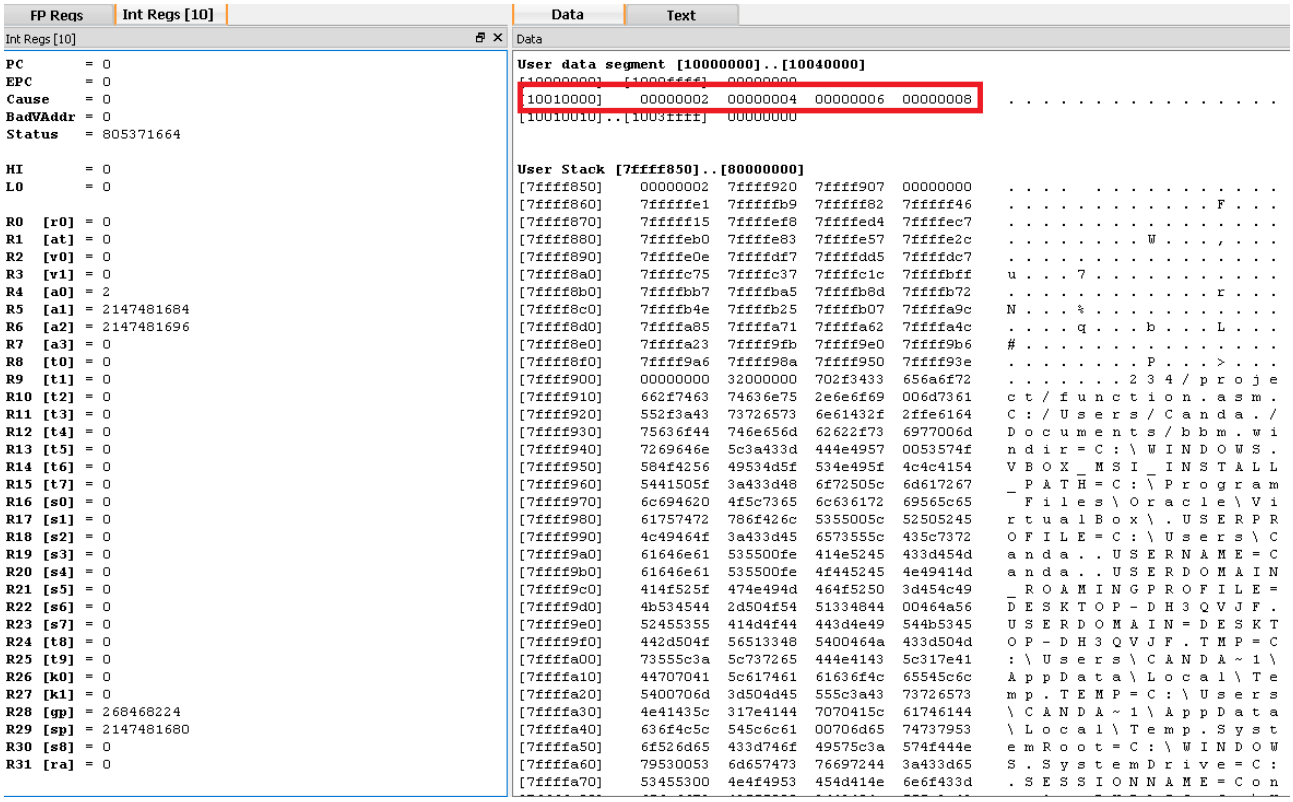

Figure 1: Test 1: before

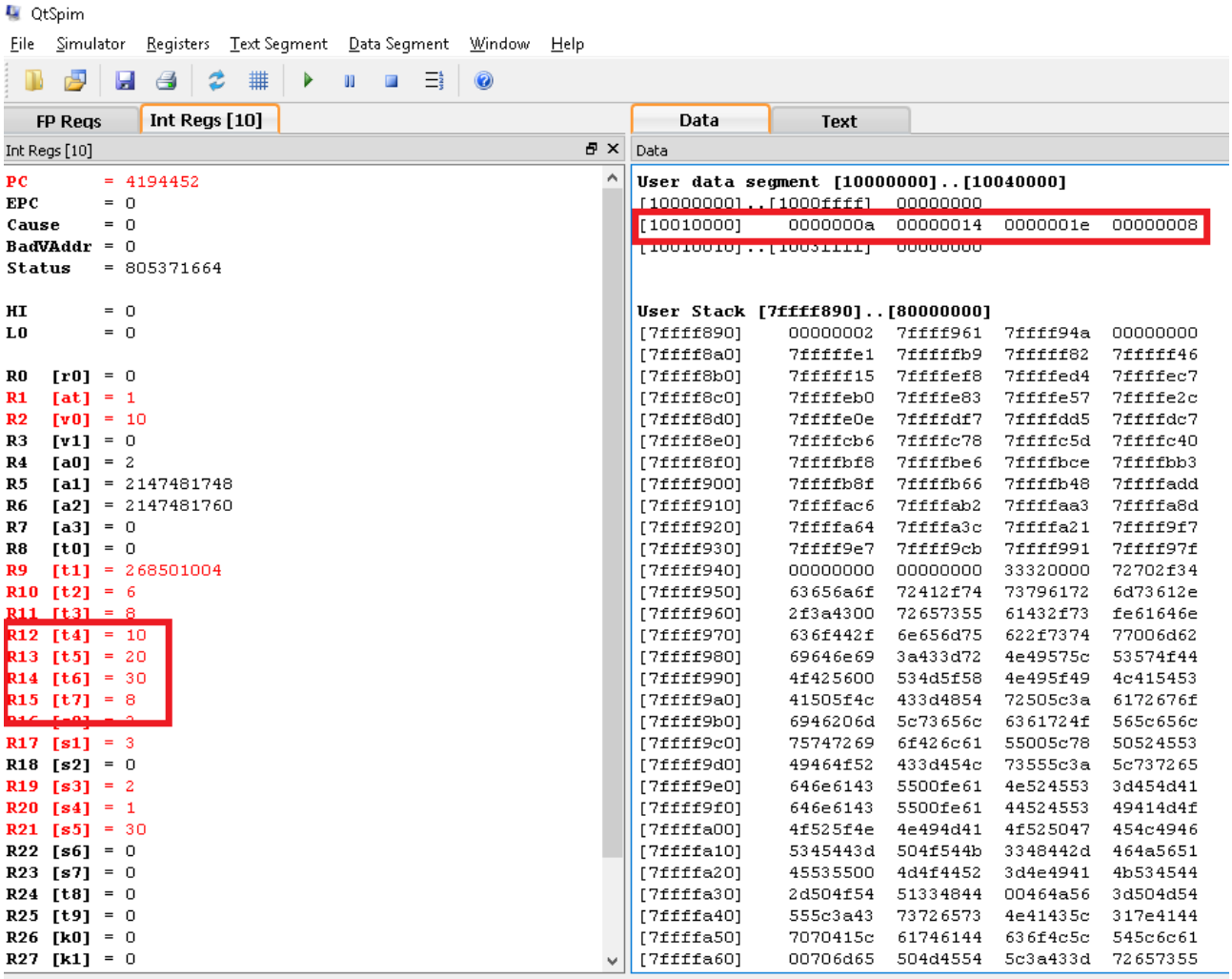

Figure 2: Test 1: after

Final state of array A is 10, 20, 30, 8 for Test1. I show results in registers too.

t4 = a[0]

t5 = a[1]

t6 = a[2]

t7 = a[3]

Results for Test 2: A=8,6,4,2

before run: Fig. 3.

after run: Fig. 4.

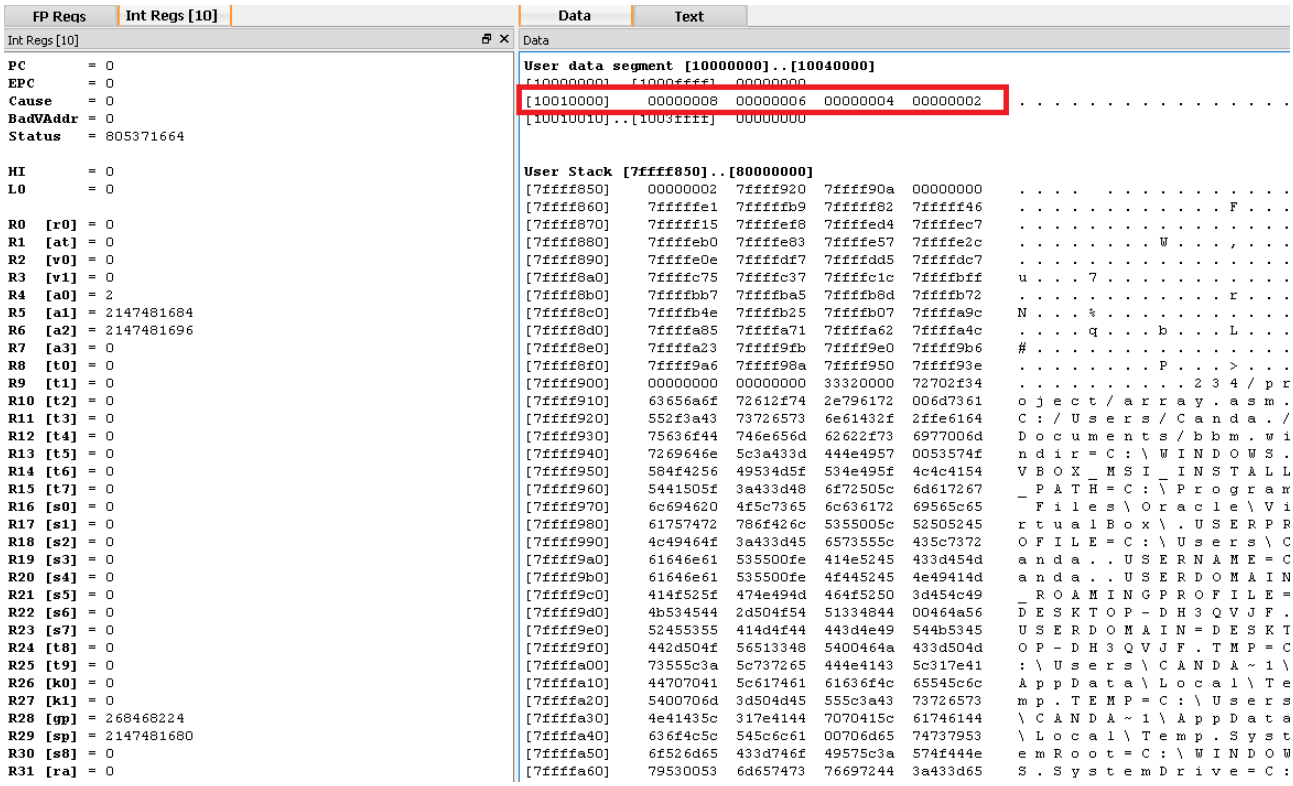

Figure 3: Test 2: before

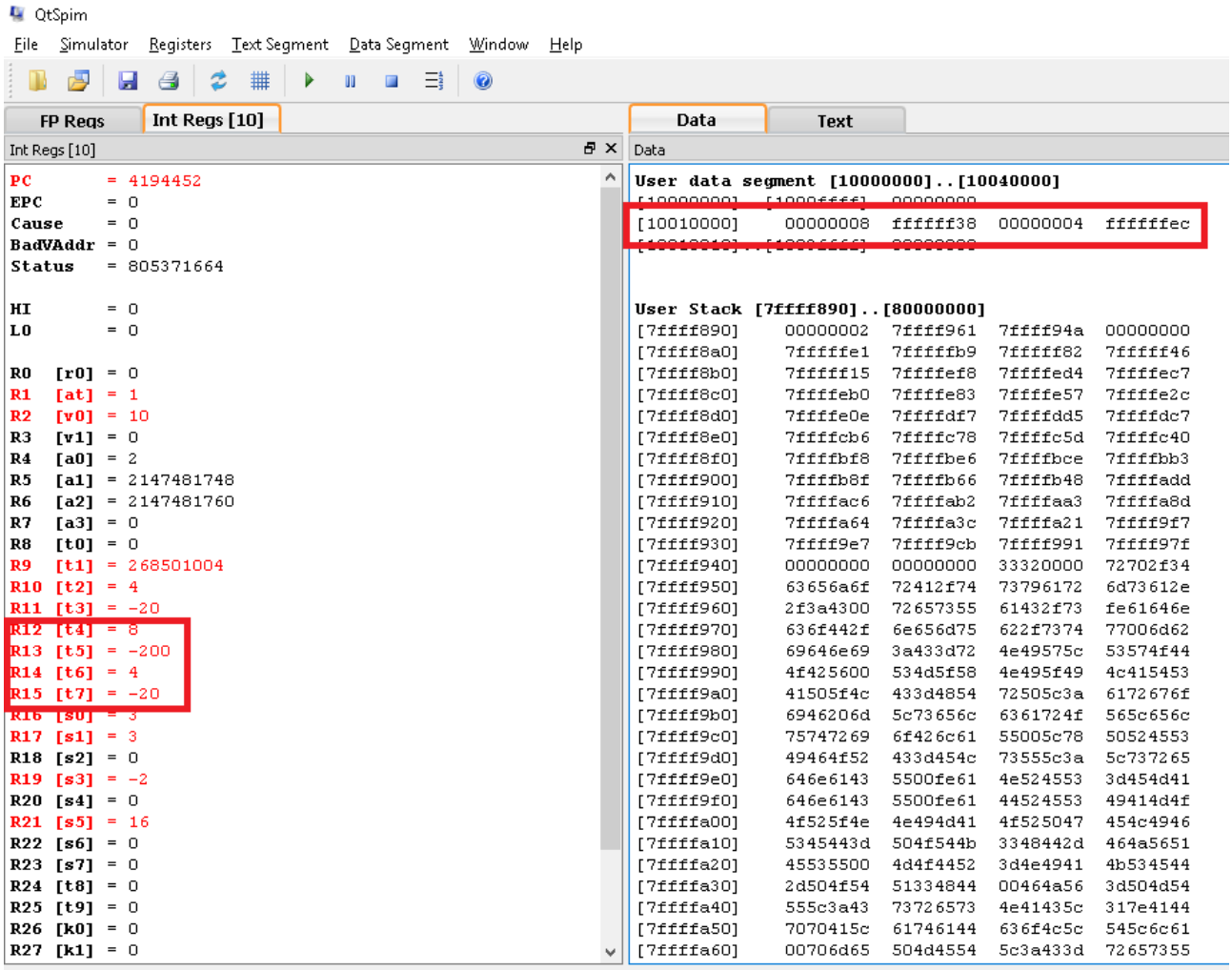

Figure 4: Test 2 : after

Final state of array A is 8, -200, 4, -20 for Test2. I show results in registers too.

t4 = a[0]
t5 = a[1]
t6 = a[2]
t7 = a[3]

Results for Test 3: A=2,2,6,4

before run: Fig. 5.
after run: Fig. 6.

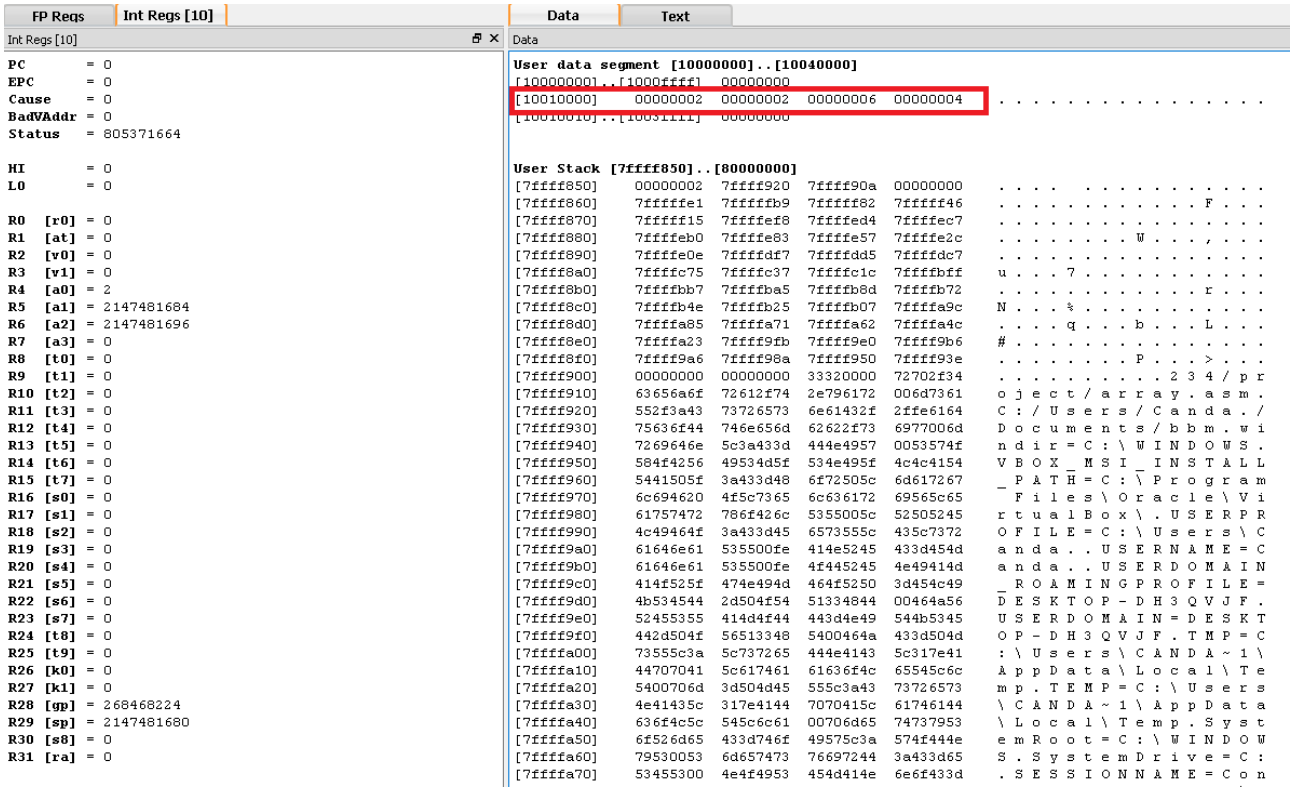

Figure 5: Test 3: before

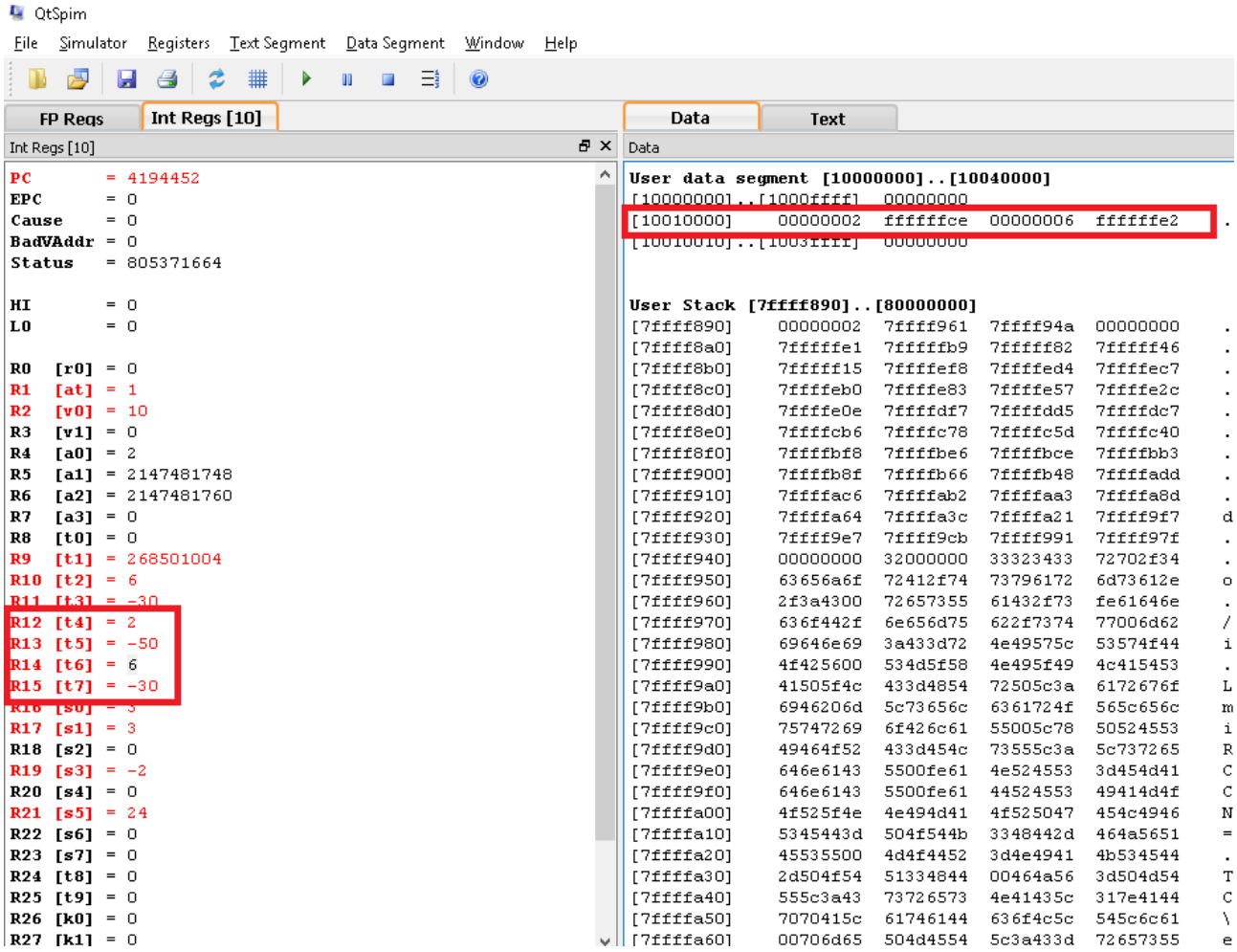

Figure 6: Test 3: after

Final state of array A is 2, -50, 6, -30 for Test3. I show results in registers too.

t4 = a[0]

t5 = a[1]

t6 = a[2]

t7 = a[3]

2 Problem: Function Calls

MIPS Assembly code for function.asm:

```
.data
x: .word 5              # input a
y: .word 3              # input b
.text

main:
lw $a0, x               # a0 = x I put it a0 beacuse i will use them for
    function arguments
lw $a1, y               # a1 = y
addi $s2, $s2, 0        # result = 0 I will show the result in register s2
beq $a0, $a1, IF        # If x = y then jump "IF"
j else                  # Otherwise jump "else"

IF:
add $v0, $a0, $a1       # v0 = x + y
sll $v0, $v0, 3         # v0 = 8 * (x + y)
add $s2, $0, $v0        # s2 = 8 * (x + y) assign result in s2
li $v0, 10              # end of code
syscall
.end

else:
jal compare             # call compare function
add $s2, $0, $v0        # assign result in s2
li $v0, 10              # end of code
syscall
.end

compare:
slt $s0, $a0, $a1
beq $s0, 1, punish      # If a1(y) greater than a0(x) jump punish
add $v0, $a0, $a1       # v0 = x + y
sll $v0, $v0, 2         # v0 = 4 * (x + y)
jr $ra                  # return result in v0

punish:
sub $v0, $a0, $a1       # v0 = x - y
sll $v0, $v0, 1         # v0 = 2 * (x -y)
jr $ra                  # return result in v0
```

Test 1: a=3, b=3
Test 2: a=3, b=5
Test 3: a=5, b=3

Results for Test 1: a=3, b=3

before run: Fig. 7.
after run: Fig. 8.

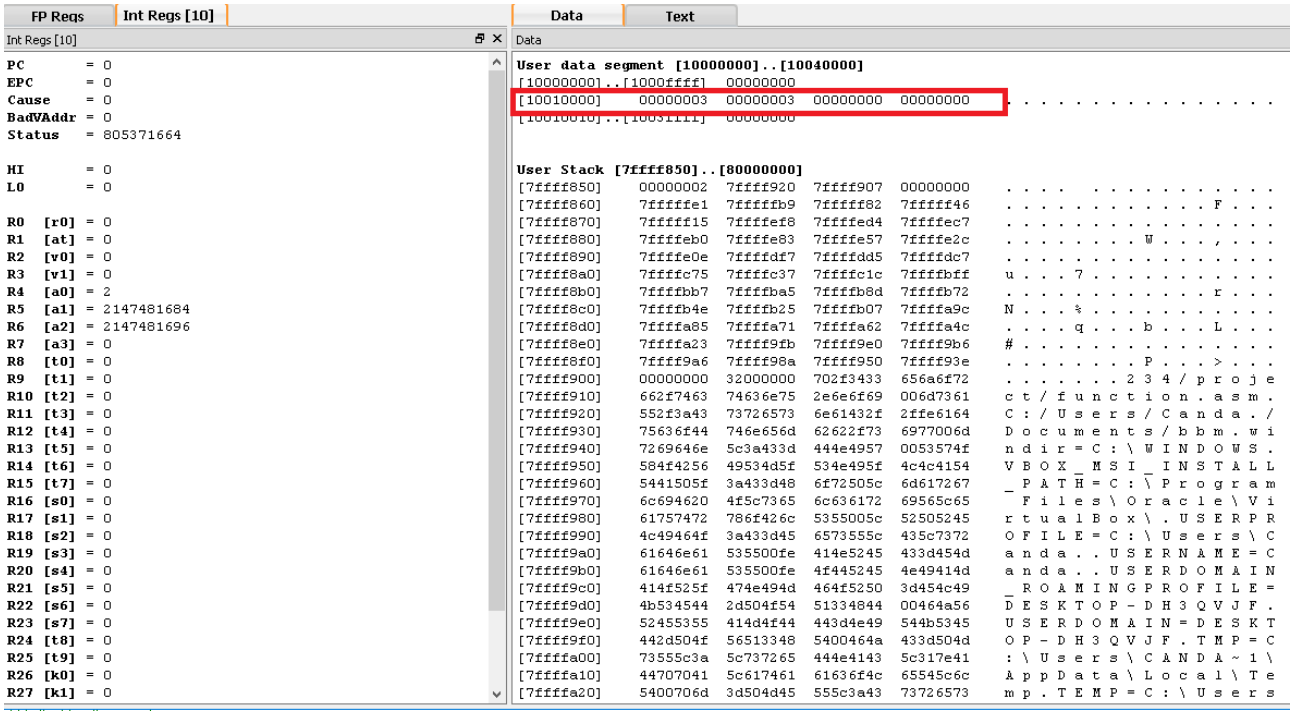

Figure 7: Test 1: before

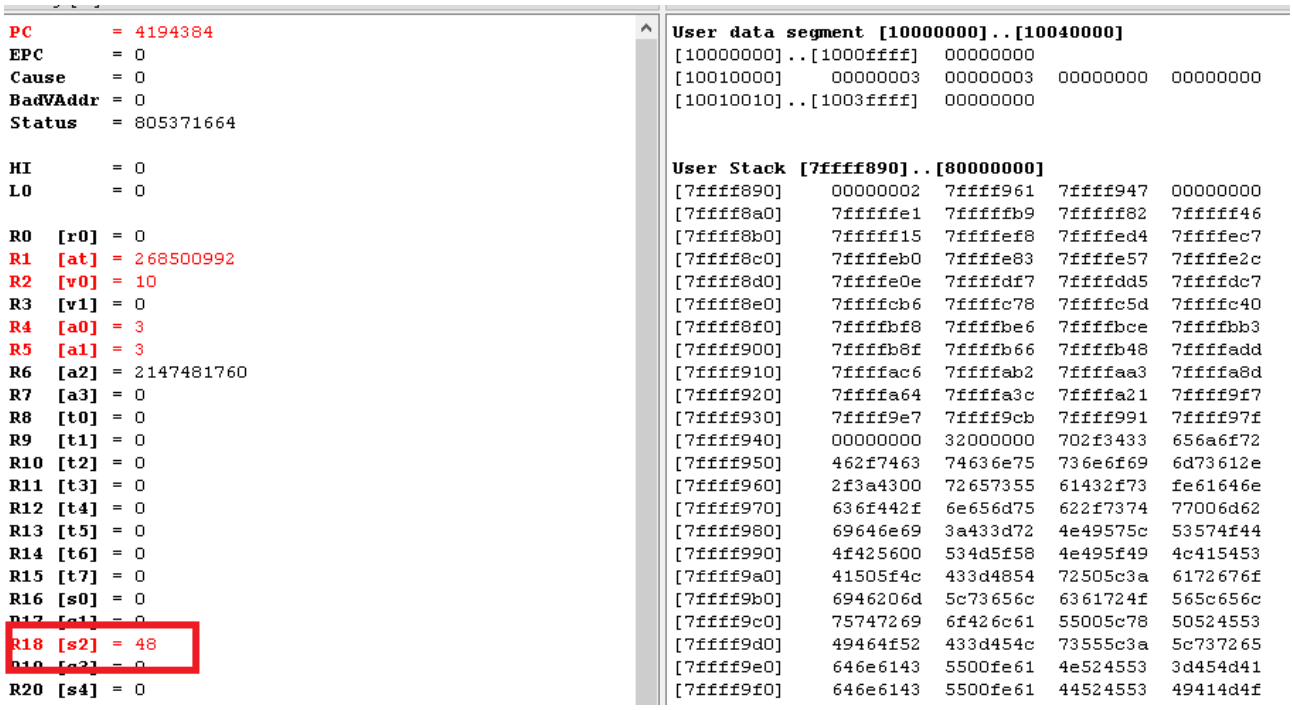

Figure 8: Test 1: after

I keep the result in s2 register. Result of test1 is 48

Results for Test 1: a=3, b=5

before run: Fig. 9.
after run: Fig. 10.

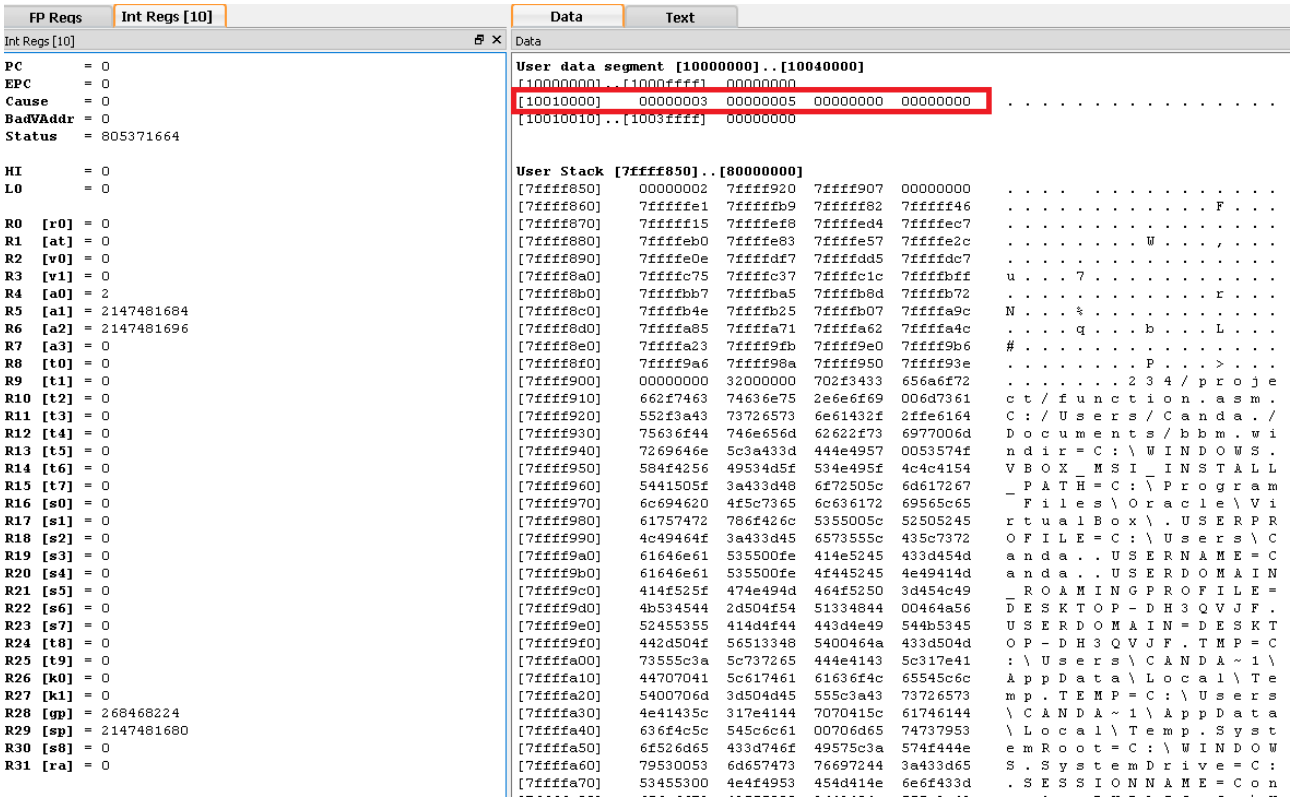

Figure 9: Test 1: before

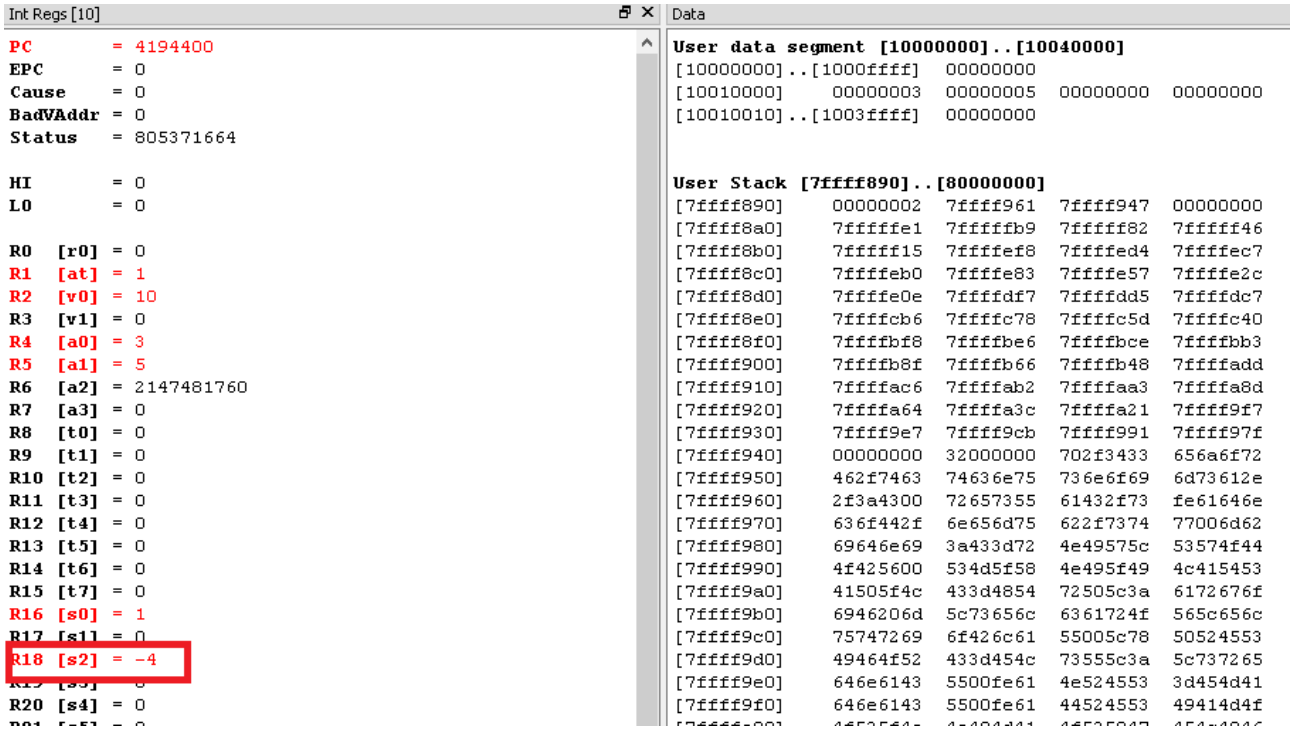

Figure 10: Test 2: after

I keep the result in s2 register. Result of test2 is -4

Results for Test 1: a=5, b=3

before run: Fig. 11.
after run: Fig. 12.

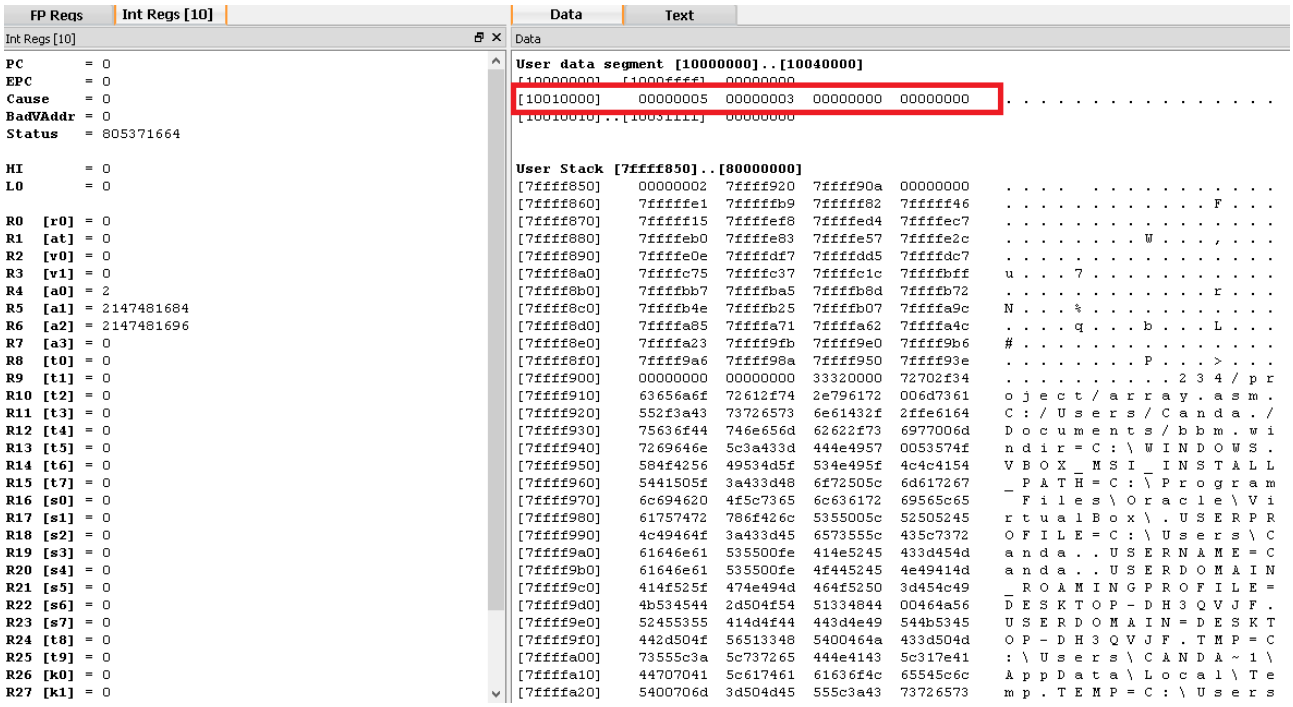

Figure 11: Test 3: before

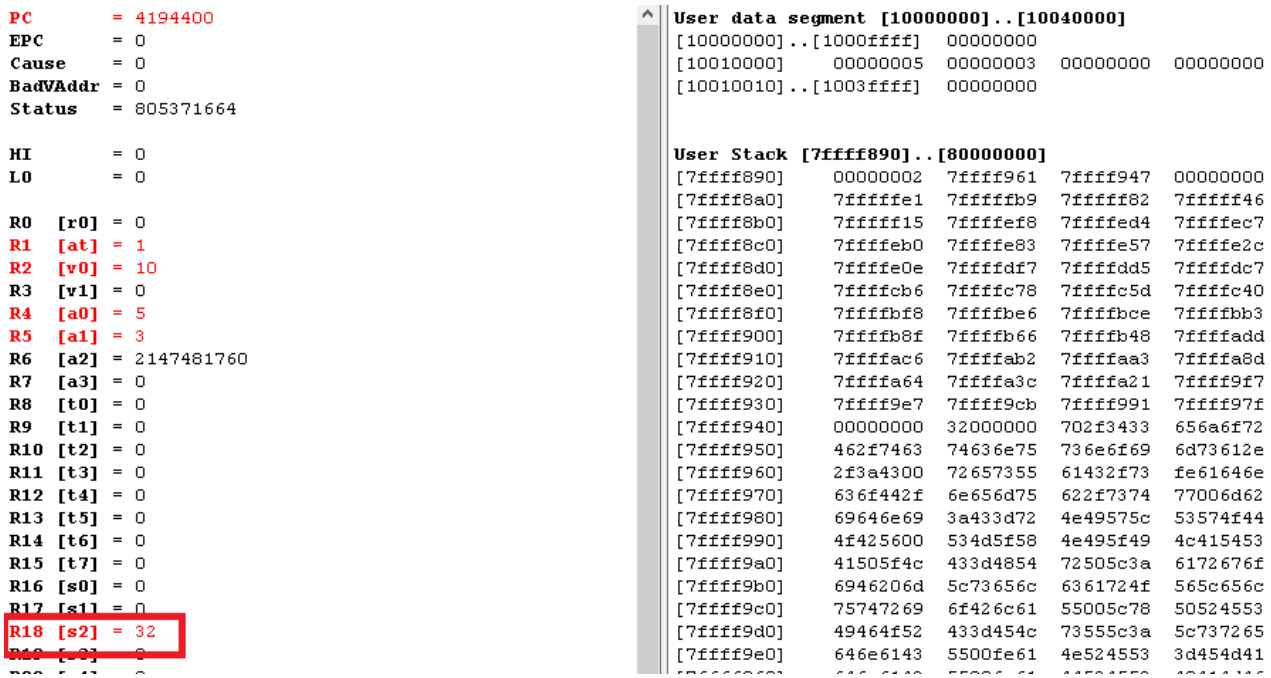

Figure 12: Test 3: after

I keep the result in s2 register. Result of test3 is 32 .