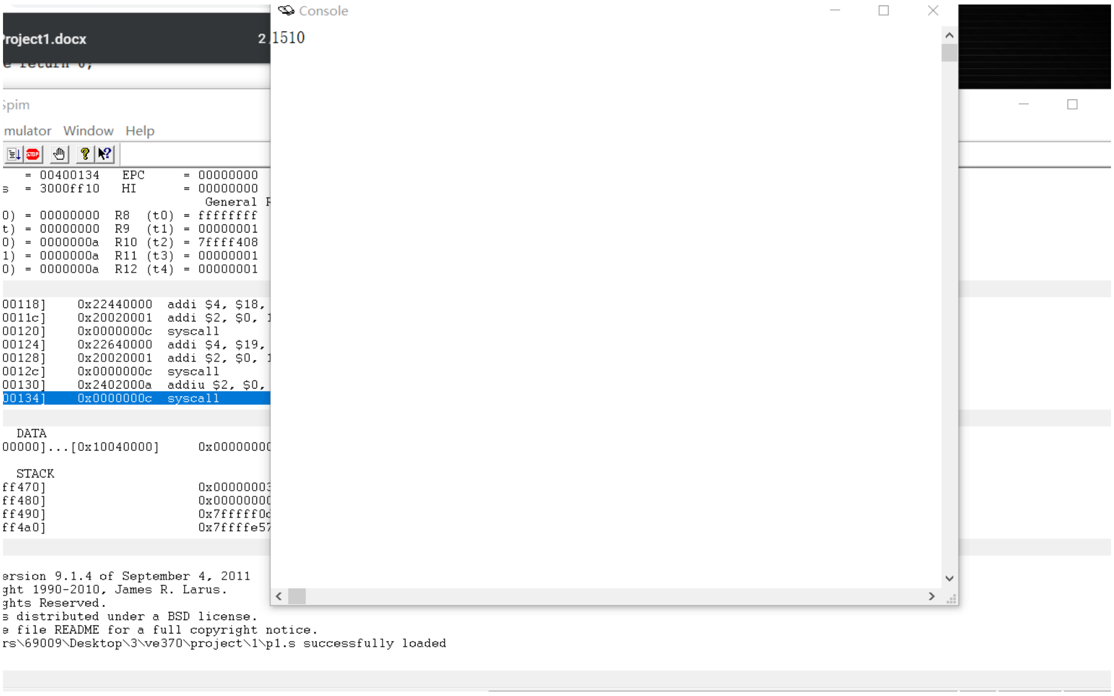
VE370 Project1



```
1510
```

```
    = 00400134   EPC      = 00000000
s   = 3000ff10   HI       = 00000000
                            General F
0)  = 00000000  R8  (t0) = ffffffff
t)  = 00000000  R9  (t1) = 00000001
0)  = 0000000a  R10 (t2) = 7ffff408
1)  = 0000000a  R11 (t3) = 00000001
0)  = 0000000a  R12 (t4) = 00000001

00118]    0x22440000  addi $4, $18,
0011c]    0x20020001  addi $2, $0, 1
00120]    0x0000000c  syscall
00124]    0x22640000  addi $4, $19,
00128]    0x20020001  addi $2, $0, 1
0012c]    0x0000000c  syscall
00130]    0x2402000a  addiu $2, $0,
00134]    0x0000000c  syscall

  DATA
00000]...[0x10040000]     0x0000000C

  STACK
ff470]                    0x00000003
ff480]                    0x0000000C
ff490]                    0x7fffff0c
ff4a0]                    0x7ffffe57

ersion 9.1.4 of September 4, 2011
ght 1990-2010, James R. Larus.
ghts Reserved.
s distributed under a BSD license.
e file README for a full copyright notice.
rs\69009\Desktop\3\ve370\project\1\p1.s successfully loaded
```

The above is the simulation results of my code. I use 25 numbers which range from 50 to 74. Therefore, there should be 10 persons who fail the exam and 15 persons who pass the exam. Since the first return value is the number of people who pass and the second return value is the number of people who fail, I think this simulation is successful.

For source code:

```
.text

.globl __start

__start:

main:



    #The array has 25 elements, ranged from 50 to 74
```

```
addi $sp, $sp, -100
addi $s0, $0, 50
sw $s0, 0($sp)
addi $s0, $0, 51
sw $s0, 4($sp)
addi $s0, $0, 52
sw $s0, 8($sp)
addi $s0, $0, 53
sw $s0, 12($sp)
addi $s0, $0, 54
sw $s0, 16($sp)
addi $s0, $0, 55
sw $s0, 20($sp)
addi $s0, $0, 56
sw $s0, 24($sp)
addi $s0, $0, 57
sw $s0, 28($sp)
addi $s0, $0, 58
sw $s0, 32($sp)
addi $s0, $0, 59
sw $s0, 36($sp)
addi $s0, $0, 60
```

```
sw $s0, 40($sp)
addi $s0, $0, 61
sw $s0, 44($sp)
addi $s0, $0, 62
sw $s0, 48($sp)
addi $s0, $0, 63
sw $s0, 52($sp)
addi $s0, $0, 64
sw $s0, 56($sp)
addi $s0, $0, 65
sw $s0, 60($sp)
addi $s0, $0, 66
sw $s0, 64($sp)
addi $s0, $0, 67
sw $s0, 68($sp)
addi $s0, $0, 68
sw $s0, 72($sp)
addi $s0, $0, 69
sw $s0, 76($sp)
addi $s0, $0, 70
sw $s0, 80($sp)
addi $s0, $0, 71
```

```
sw $s0, 84($sp)
addi $s0, $0, 72
sw $s0, 88($sp)
addi $s0, $0, 73
sw $s0, 92($sp)
addi $s0, $0, 74
sw $s0, 96($sp)
lw $t9, 96($sp)
lw $t9, 92($sp)
lw $t9, 88($sp)


#Set array size
addi $s1, $0, 25


#for PassCnt
add $a3, $0, 1       #cntType in a3
addi $a1, $s1, 0     #numElements in a1
addi $a2, $sp, 0     #int A[] in a2
jal countArray       #jump to countArray
add $t9, $0,$0
add $s2, $v1, $0     #store result in s2
```

```
#for FailCnt
addi $v1, $0, 0
addi $a3, $0, -1        #cntType
addi $a1, $s1, 0        #numElements
addi $a2, $sp, 0        #int A[]
lw $t9, 0($a2)
jal countArray          #jump to countArray
add $t9, $0,$0
add $s3, $v1, $0        #store result in s3

addi $sp, $sp, 100
#Print results
addi $a0, $s2, 0
addi $v0, $0, 1
syscall

addi $a0, $s3, 0
addi $v0, $0, 1
syscall

#EXIT
addiu $v0, $0, 10
```

```
	syscall


countArray:
	addi $t0, $a1, 0	#t0=i=numElements
For:	addi $t0, $t0, -1	#i--
	slti $t1, $t0, 0	#t0<0 t1=1, else t1=0
	sll $t2, $t0, 2		#index
	add $t2, $a2, $t2		#t2 is the address of A[i]
	lw $t6, 0($t2)		#t2=A[i]


	addi $t3, $0, 1		#for comparison
	beq $t1, $t3, EXIT		#if i<0, return to main
	add $t9, $0,$0




	beq $a3, $t3, Case1	#if cntType=1, jump to case1
	add $t9, $0,$0


	j Case2		#otherwise, jump to case2
	add $t9, $0,$0
```

```
EXIT:     jr $ra              #return to main
     add $t9, $0,$0
Case1:    slti $t4, $t6, 60        #if t6<60, t4=1
     addi $t5, $0, 1
     sub $t5, $t5, $t4        #if t4=1,t5=0
     add $v1, $v1, $t5

     j For
     add $t9, $0,$0
Case2:    slti $t4, $t6, 60        #if t6<60, t4=1
     add $v1, $v1, $t4
     j For
     add $t9, $0,$0
```