



Department Of Computer Engineering

Project1

Computer Organization

**Name** : İsmail

**Surname** : Ateş

**Student Number** : 21626953

**Programming Language** : MIPS

**Advisor** : Hüseyin Temuçin

# Q.1 ARRAYS USING FOR LOOP

## 1.INTRODUCTION / AIM

In this experiment, , we will learn how to write and simulate MIPS code.

## 2.PROBLEM

We have 2 arrays which are A[n] and B[n]. If  $A[i] > B[i]$  we must swap the elements of each array.

## 3.SUMMARY OF CODE(Labels)

### 3.1 main:

```
main:
    addi $sp, $sp, -4 #open stack
    addi $t3, $0, 0   #define i for forloop
    la $t1, A
    la $t2, B
    addi $t0, $0, 4   #constant for byteset (use for adding 4 byte on array address)
    addi $a1, $t1, 0  #argument of Address A
    addi $a2, $t2, 0  #argument of Address B
    addi $t5, $0, 5   #constant for forloop
    j forloop
```

On main label firstly I open the stack frame and I define \$t3 variable for forloop. I get addresses of A[0] and B[0] with variables which are \$t1, \$t2. I define \$a1, \$a2 for using in function which equals \$t1, \$t2. I define \$t5 for using like a constant=5 for forloop. On main label I define this variables and jump to forloop label.

### 3.2 forloop:

```

foorloop:
    slt $t4, $t3, $t5    #check foorloop
    beq $t4, 1, compare #if i<5 it goes compare
    j finish

```

On foorlop label it compares \$t3 (i) and \$t5(constant) if \$t3 less than \$t5 it define \$t4=1 otherwise 0. On second line if \$t4 equals 1 it jump to compare label otherwise goes to finish label. Arrays lengths are 5 so I used static code below.

### 3.3 Compare:

```

compare:
    addi $t3, $t3, 1    #i=i+1
    lw $s1, 0($a1)      #get s1=A[i]
    lw $s2, 0($a2)      #get s2= B[i]
    slt $s0, $s1, $s2   #check A[i]<B[i]
    beq $s0, 0, swap     #if A[i]< B[i] go swap
    add $a1, $a1, $t0    #get adrees A[i]+4
    add $a2, $a2, $t0    #get address B[i]+4
    j foorloop

```

On compare label I increase i then define variables \$s1, \$s2 for values of A[i], B[i]. On 4. line I checked A[i] is less than B[i]. If A[i] < B[i] define \$s0 =1 . Then checking if \$s0 equals 0 It goings to swap label otherwise it won't be swap the elements of each array and it increase the addresses of each arrays 4 byte and jump to foorloop.

### 3.4 Swap:

```

swap:
    sw $s1, 0($sp)      #stack[0]= A[i]
    sw $s2, 0($a1)      #A[i]= B[i]
    lw $s1, 0($sp)      #A[i]= stack[0]
    sw $s1, 0($a2)      #B[i]=A[i]
    add $a1, $a1, $t0    #get adrees A[i]+4
    add $a2, $a2, $t0    #get adress B[i]+4
    j foorloop

```

I store the value of A[i] on stack frame. Than Update A[i] value equals B[i]. Then I get value of A[i] on stack. After that update B[i] value equals A[i]. And increase addresses of each array 4 byte.

### 3.5 Finish:

I closed stack frame and the code below was executed and program was finished.

# 4.TEST CASES

## 4.1 Test Case:

A={6,2,8,4,10}, B={1,7,3,9,5}

## Before Executed

Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	4194304	0x200b0000	addi \$t1,\$0,0	11: addi \$t3, \$0, 0 #define 1 for forloop
	4194308	0x3c011001	lui \$t1,4097	12: la \$t1, A
	4194312	0x34290000	ori \$9,\$1,0	
	4194316	0x3c011001	lui \$t1,4097	13: la \$t2, B
	4194320	0x342a0014	ori \$t0,\$1,20	
	4194324	0x20080004	addi \$8,\$0,4	14: addi \$t0, \$0, 4 #constant for byteset(use for adding 4 byte on array...
	4194328	0x21250000	addi \$5,\$9,0	15: addi \$a1, \$t1, 0 #argument of Adrees A
	4194332	0x21460000	addi \$6,\$t0,0	16: addi \$a2, \$t2, 0 #argumen of Adrees B
	4194336	0x200d0005	addi \$t3,\$0,5	17: addi \$t5, \$0, 5 #constant for forloop
	4194340	0x0810000a	j 4194344	18: j foorloop

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	6	2	8	4	10	1	7	3
268501024	9	5	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0

0x10010000 (.data)

Hexadecimal Addresses

Hexadecimal Values

ASCII

Registers

Coproc 1

Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$a0	16	0
\$a1	17	0
\$a2	18	0
\$a3	19	0
\$a4	20	0
\$a5	21	0
\$a6	22	0
\$a7	23	0
\$t5	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0

## After Executed

## 4.2 Test Case:

A={1,3,3,5,5}, B={2,2,3,4,5}

## Before Executed

**Text Segment**

Bkpt	Address	Code	Basic	Source
	4194304	0x200b0000	addi \$t1,\$0,0	11: addi \$t3, \$0, 0 #define 1 for foorloop
	4194308	0x3c011001	lui \$t1,4097	12: la \$t1, A
	4194312	0x34290000	ori \$9,\$t1,0	
	4194316	0x3c011001	lui \$t1,4097	13: la \$t2, B
	4194320	0x342a0014	ori \$t0,\$t1,20	
	4194324	0x20080004	addi \$8,\$0,4	14: addi \$t0, \$0, 4 #constant for byteset(use for adding 4 byte on array...
	4194328	0x21250000	addi \$5,\$9,0	15: addi \$a1, \$t1, 0 #argument of Adrees A
	4194332	0x21460000	addi \$6,\$t0,0	16: addi \$a2, \$t2, 0 #argümen of Adrees B
	4194336	0x200d0005	addi \$t3,\$0,5	17: addi \$t5, \$0, 5 #constant for foorloop
	4194340	0x0810000a	j 4194344	18: j foorloop

**Data Segment**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1	3	3	5	5	2	2	3
268501024	4	5	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0

**Registers**

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0

## After Executed

**Text Segment**

Bkpt	Address	Code	Basic	Source
	4194304	0x200b0000	addi \$t1,\$0,0	11: addi \$t3, \$0, 0 #define 1 for foorloop
	4194308	0x3c011001	lui \$t1,4097	12: la \$t1, A
	4194312	0x34290000	ori \$9,\$t1,0	
	4194316	0x3c011001	lui \$t1,4097	13: la \$t2, B
	4194320	0x342a0014	ori \$t0,\$t1,20	
	4194324	0x20080004	addi \$8,\$0,4	14: addi \$t0, \$0, 4 #constant for byteset(use for adding 4 byte on array...
	4194328	0x21250000	addi \$5,\$9,0	15: addi \$a1, \$t1, 0 #argument of Adrees A
	4194332	0x21460000	addi \$6,\$t0,0	16: addi \$a2, \$t2, 0 #argümen of Adrees B
	4194336	0x200d0005	addi \$t3,\$0,5	17: addi \$t5, \$0, 5 #constant for foorloop
	4194340	0x0810000a	j 4194344	18: j foorloop

**Data Segment**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	1	2	3	4	5	2	3	3
268501024	5	5	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0

**Registers**

Name	Number	Value
\$zero	0	0
\$at	1	1
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	268501012
\$a2	6	268501032
\$a3	7	0
\$t0	8	4
\$t1	9	268500992
\$t2	10	268501012
\$t3	11	5
\$t4	12	0
\$t5	13	5
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	5
\$s2	18	5
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0

# Q.2 FUNCTION CALLS

## 1.INTRODUCTION / AIM

In this experiment, , we will learn how to write and simulate MIPS code.

## 2.PROBLEM

Compare two integer and return values with functions. In code, we are not allowed to use multiplication instructions .

## 3.SUMMARY OF CODE(Labels)

### 3.1 Main:

```
main:    addi $s0, $0, 3      #a=3
         addi $s1, $0, 3      #b=5
         addi $s2, $0, 0      #result=0
         addi $a0, $s0, 0     #argumentA=a
         addi $a1, $s1, 0     #argumentB=b
         bne $s0, $s1, compare #if a!=b go compare
         add $s2, $s0,$s1     #else result=a+b
         addi $v0, $s2, 0     #returnValue=result
```

On this label I defined integers which are \$s0,\$s1 equals to a and b. Then defined \$s2 equals result. I defined arguments for using on functions which are \$a0,\$a1 equals to a and b. Then if a is not equal to b it goes to compare label. Otherwise adding \$s0 and \$s1 and define \$s2. Then return the value of \$s2 for \$v0.

### 3.2 Compare:

```
compare: slt $t1, $a0, $a1    #if a<b go punish else go award
         beq $t1, 1, punish
         beq $t1, 0, award
```

On this label if \$a0 less than \$a1 it defines \$t1 1 otherwise it defines \$t1 0. If \$t1 equals 1 it goes to punish label, if \$t1 equals 0 it goes to award label.

### 3.3 Award:

```
award:   add $t0, $a0, $a1     #t0=a+b
         sll $t0, $t0, 1       #t0=t0*2
         addi $v0, $t0, 0      #returnValue=t0
         j finish
```

On this label it's add \$a0 and \$a1 and define \$t0 the result of adding. Then I used shift left logical for multiply the \$t0 with 2. Then define the result value \$v0 equals to \$t0. Then jump the finish label.

### 3.4 Punish:

```
punish:  sub $t0, $a0, $a1     #t0=a-b
         sll $t0, $t0, 1       #t0=t0*2
         addi $v0, $t0, 0      #returnValue=t0
         j finish
```



On this label it's subtract \$a0 and \$a1 and define \$t0 the result of subtraction. Then I used shift left logical for multiply the \$t0 with 2. Then define the result value \$v0 equals to \$t0. Then jump the finish label.

### 3.5 Finish:

The code below was executed and program was finished.

## 4.TEST CASES

### 4.1 Test Case:

a=3, b=3

### Before Executed

Text Segment					Registers		
Bkpt	Address	Code	Basic	Source	Name	Number	Value
	4194304	0x20100003	addi \$t6,\$0,3	6: main: addi \$a0, \$0, 3 #a=3	\$zero	0	0
	4194308	0x20110003	addi \$t7,\$0,3	7: addi \$a1, \$0, 3 #b=3	\$a0	1	0
	4194312	0x20120000	addi \$t8,\$0,0	8: addi \$a2, \$0, 0 #result=0	\$a1	2	0
	4194316	0x22040000	addi \$t4,\$t6,0	9: addi \$a0, \$a0, 0 #argumentA=a	\$a2	3	0
	4194320	0x22250000	addi \$t5,\$t7,0	10: addi \$a1, \$a1, 0 #argumentB=b	\$a3	4	0
	4194324	0x16110003	bne \$t6,\$t7,3	11: bne \$a0, \$a1, compare #if a!=b go compare	\$a4	5	0
	4194328	0x02119020	add \$t8,\$t6,\$t7	12: add \$a2, \$a0,\$a1 #else result=a+b	\$a5	6	0
	4194332	0x22420000	addi \$t2,\$t8,0	13: addi \$v0, \$a2, 0 #returnValue=result	\$a6	7	0
	4194336	0x08100016	j 4194392	14: j finish	\$a7	8	0
	4194340	0x005492a	slt \$t5,\$t4,\$t5	15: compare: slt \$t1, \$a0, \$a1 #if a<b go punish else go award	\$a8	9	0
					\$a9	10	0
					\$a10	11	0
					\$a11	12	0
					\$a12	13	0
					\$a13	14	0
					\$a14	15	0
					\$a15	16	0
					\$a16	17	0
					\$a17	18	0
					\$a18	19	0
					\$a19	20	0
					\$a20	21	0
					\$a21	22	0
					\$a22	23	0
					\$a23	24	0
					\$a24	25	0
					\$a25	26	0
					\$a26	27	0

### After Executed

Text Segment					Registers		
Bkpt	Address	Code	Basic	Source	Name	Number	Value
	4194304	0x20100003	addi \$t6,\$0,3	6: main: addi \$a0, \$0, 3 #a=3	\$zero	0	0
	4194308	0x20110003	addi \$t7,\$0,3	7: addi \$a1, \$0, 3 #b=3	\$a0	1	0
	4194312	0x20120000	addi \$t8,\$0,0	8: addi \$a2, \$0, 0 #result=0	\$a1	2	3
	4194316	0x22040000	addi \$t4,\$t6,0	9: addi \$a0, \$a0, 0 #argumentA=a	\$a2	3	3
	4194320	0x22250000	addi \$t5,\$t7,0	10: addi \$a1, \$a1, 0 #argumentB=b	\$a3	4	3
	4194324	0x16110003	bne \$t6,\$t7,3	11: bne \$a0, \$a1, compare #if a!=b go compare	\$a4	5	0
	4194328	0x02119020	add \$t8,\$t6,\$t7	12: add \$a2, \$a0,\$a1 #else result=a+b	\$a5	6	0
	4194332	0x22420000	addi \$t2,\$t8,0	13: addi \$v0, \$a2, 0 #returnValue=result	\$a6	7	0
	4194336	0x08100016	j 4194392	14: j finish	\$a7	8	0
	4194340	0x005492a	slt \$t5,\$t4,\$t5	15: compare: slt \$t1, \$a0, \$a1 #if a<b go punish else go award	\$a8	9	0
					\$a9	10	0
					\$a10	11	0
					\$a11	12	0
					\$a12	13	0
					\$a13	14	0
					\$a14	15	0
					\$a15	16	3
					\$a16	17	3
					\$a17	18	6
					\$a18	19	0
					\$a19	20	0
					\$a20	21	0
					\$a21	22	0
					\$a22	23	0
					\$a23	24	0
					\$a24	25	0
					\$a25	26	0
					\$a26	27	0

### 4.2 Test Case:

a=3, b=5

# Before Executed

Text Segment

Bkpt	Address	Code	Basic	Source
4194304	0x20100003	addi \$t6,\$0,3	6: main: addi \$s0,\$0,3	#a=3
4194308	0x20110005	addi \$t7,\$0,5	7: addi \$s1,\$0,5	#b=5
4194312	0x20120000	addi \$t8,\$0,0	8: addi \$s2,\$0,0	#result=0
4194316	0x22040000	addi \$t4,\$t6,0	9: addi \$a0,\$s0,0	#argumentA=a
4194320	0x22250000	addi \$t5,\$t7,0	10: addi \$a1,\$s1,0	#argumentB=b
4194324	0x1e110003	bne \$t6,\$t7,3	11: bne \$s0,\$s1,compare	#if a!=b go compare
4194328	0x02119020	add \$t8,\$t6,\$t7	12: add \$s2,\$s0,\$s1	#else result=a+b
4194332	0x22420000	addi \$t2,\$t8,0	13: addi \$v0,\$s2,0	#returnValue=result
4194336	0x08100016	j 4194392	14: j finish	
4194340	0x0085482a	slt \$t9,\$t4,\$t5	15: compare: slt \$t1,\$a0,\$a1	#if a<b go punish else go award

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	0	0	0	0	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0

# After Executed

Text Segment

Bkpt	Address	Code	Basic	Source
4194304	0x20100003	addi \$t6,\$0,3	6: main: addi \$s0,\$0,3	#a=5
4194308	0x20110005	addi \$t7,\$0,5	7: addi \$s1,\$0,5	#b=3
4194312	0x20120000	addi \$t8,\$0,0	8: addi \$s2,\$0,0	#result=0
4194316	0x22040000	addi \$t4,\$t6,0	9: addi \$a0,\$s0,0	#argumentA=a
4194320	0x22250000	addi \$t5,\$t7,0	10: addi \$a1,\$s1,0	#argumentB=b
4194324	0x1e110003	bne \$t6,\$t7,3	11: bne \$s0,\$s1,compare	#if a!=b go compare
4194328	0x02119020	add \$t8,\$t6,\$t7	12: add \$s2,\$s0,\$s1	#else result=a+b
4194332	0x22420000	addi \$t2,\$t8,0	13: addi \$v0,\$s2,0	#returnValue=result
4194336	0x08100016	j 4194392	14: j finish	
4194340	0x0085482a	slt \$t9,\$t4,\$t5	15: compare: slt \$t1,\$a0,\$a1	#if a<b go punish else go award

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	0	0	0	0	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0

Name	Number	Value
\$zero	0	0
\$at	1	1
\$v0	2	-4
\$v1	3	0
\$a0	4	3
\$a1	5	5
\$a2	6	0
\$a3	7	0
\$t0	8	-4
\$t1	9	1
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	3
\$s1	17	5
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0

# 4.3 Test Case:

a=5, b=3

# Before Executed

Text Segment

Bkpt	Address	Code	Basic	Source
4194304	0x20100005	addi \$t6,\$0,5	6: main: addi \$s0,\$0,5	#a=5
4194308	0x20110003	addi \$t7,\$0,3	7: addi \$s1,\$0,3	#b=3
4194312	0x20120000	addi \$t8,\$0,0	8: addi \$s2,\$0,0	#result=0
4194316	0x22040000	addi \$t4,\$t6,0	9: addi \$a0,\$s0,0	#argumentA=a
4194320	0x22250000	addi \$t5,\$t7,0	10: addi \$a1,\$s1,0	#argumentB=b
4194324	0x1e110003	bne \$t6,\$t7,3	11: bne \$s0,\$s1,compare	#if a!=b go compare
4194328	0x02119020	add \$t8,\$t6,\$t7	12: add \$s2,\$s0,\$s1	#else result=a+b
4194332	0x22420000	addi \$t2,\$t8,0	13: addi \$v0,\$s2,0	#returnValue=result
4194336	0x08100016	j 4194392	14: j finish	
4194340	0x0085482a	slt \$t9,\$t4,\$t5	15: compare: slt \$t1,\$a0,\$a1	#if a<b go punish else go award

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	0	0	0	0	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0



After Executed

Text Segment

Blkpt	Address	Code	Basic	Source
	4194304	0x20100005	addi \$t6,\$0,5	6: main: addi \$a0,\$0,5 #a=5
	4194308	0x20110003	addi \$t7,\$0,3	7: addi \$s1,\$0,3 #b=3
	4194312	0x20120000	addi \$t8,\$0,0	8: addi \$s2,\$0,0 #result=0
	4194316	0x22040000	addi \$t4,\$t6,0	9: addi \$a0,\$s0,0 #argumentA=a
	4194320	0x22250000	addi \$t5,\$t7,0	10: addi \$a1,\$s1,0 #argumentB=b
	4194324	0x16110003	bne \$t6,\$t7,3	11: bne \$s0,\$s1,compare #if a!=b go compare
	4194328	0x02119020	add \$t8,\$t6,\$t7	12: add \$s2,\$s0,\$s1 #else result=a+b
	4194332	0x22420000	addi \$t2,\$t8,0	13: addi \$v0,\$s2,0 #returnValue=result
	4194336	0x08100016	j 4194392	14: j finish
	4194340	0x0055482a	slt \$t1,\$t4,\$t5	15: compare: slt \$t1,\$a0,\$a1 #if a<b go punish else go award

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	0	0	0	0	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0

0x10010000 (.data)

☐ Hexadecimal Addresses☐ Hexadecimal Values☐ ASCII

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	16
\$v1	3	0
\$a0	4	5
\$a1	5	3
\$a2	6	0
\$a3	7	0
\$t0	8	16
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$a0	16	5
\$s1	17	3
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0