Hacettepe University Computer Engineering Department
BBM234 Computer Organization 2018-2019 Spring Term MIPS Project 1

Name: Burak

Surname: Yılmaz

Student Number: 21627868

E-mail: b21627868@cs.hacettepe.edu.tr

Introduction

In this project, we learnt how to write and simulate MIPS code. We implemented two basic mips program.

Problem 1:

We are given two arrays A[n] and B[n]. Write a MIPS programs that compares each element of these two arrays one by one. If A[i] > B[i], swap the elements of each array. Otherwise, do not swap.

Our array size will be 5. I tested my code for the test cases given below.

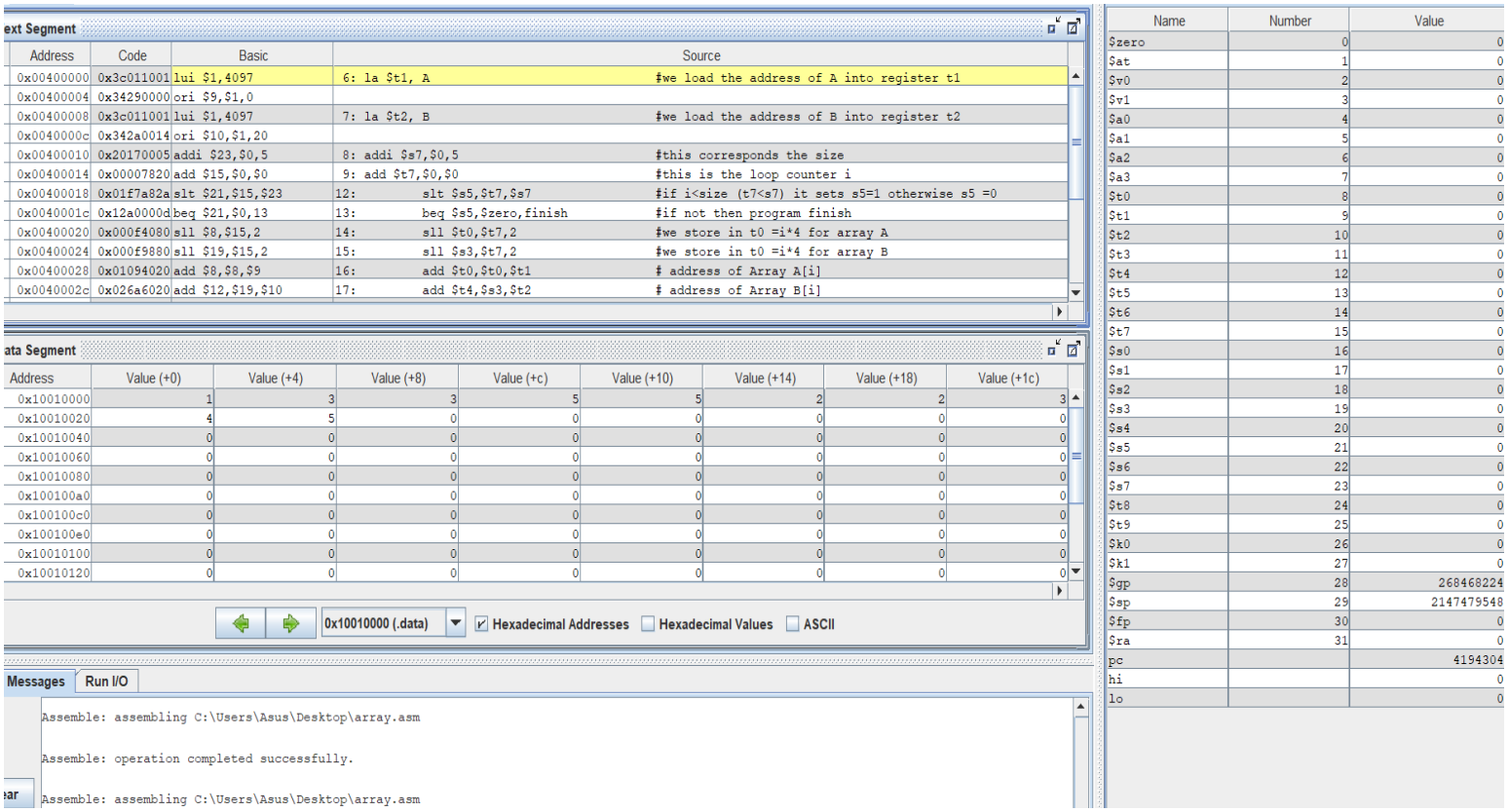
Test 1: A={6,2,8,4,10}, B={1,7,3,9,5}

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

For each test, I saved the screenshots of the memory before running the code and screenshot of the memory after running the code. You can see below.

TEST 1

Before the program starts.



Text Segment

Address	Code	Basic	Source
0x00400000	0x3c011001	lui $1,4097	6: la $t1, A #we load the address of A into register t1
0x00400004	0x34290000	ori $9,$1,0	
0x00400008	0x3c011001	lui $1,4097	7: la $t2, B #we load the address of B into register t2
0x0040000c	0x342a0014	ori $10,$1,20	
0x00400010	0x20170005	addi $23,$0,5	8: addi $s7,$0,5 #this corresponds the size
0x00400014	0x00007820	add $15,$0,$0	9: add $t7,$0,$0 #this is the loop counter i
0x00400018	0x01f7a82a	slt $21,$15,$23	12: slt $s5,$t7,$s7 #if i<size (t7<s7) it sets s5=1 otherwise s5 =0
0x0040001c	0x12a0000d	beq $21,$0,13	13: beq $s5,$zero,finish #if not then program finish
0x00400020	0x000f4080	sll $8,$15,2	14: sll $t0,$t7,2 #we store in t0 =i*4 for array A
0x00400024	0x000f9880	sll $19,$15,2	15: sll $s3,$t7,2 #we store in t0 =i*4 for array B
0x00400028	0x01094020	add $8,$8,$9	16: add $t0,$t0,$t1 # address of Array A[i]
0x0040002c	0x026a6020	add $12,$19,$10	17: add $t4,$s3,$t2 # address of Array B[i]

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	3	3	5	5	2	2	3
0x10010020	4	5	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run I/O

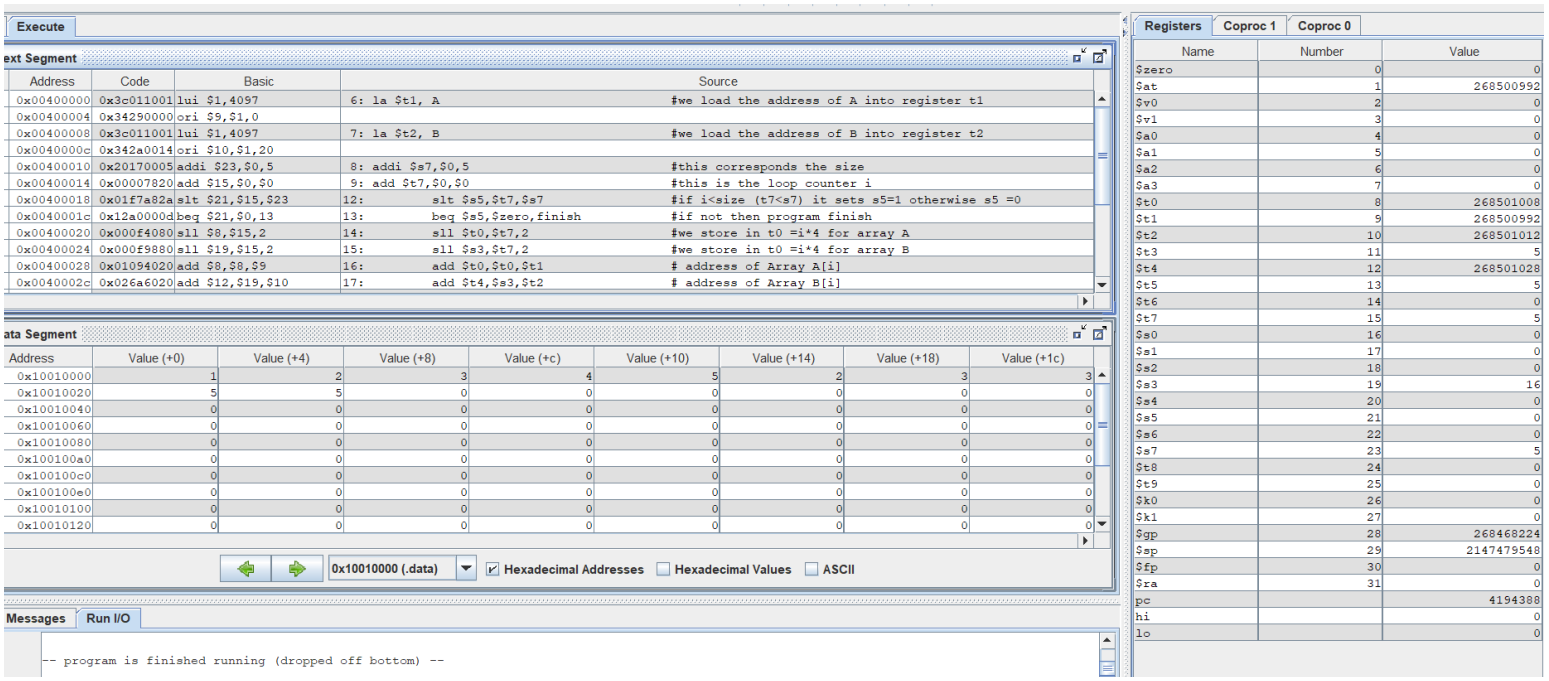
```
Assemble: assembling C:\Users\Asus\Desktop\array.asm

Assemble: operation completed successfully.

Assemble: assembling C:\Users\Asus\Desktop\array.asm
```

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
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

After the program executed.



Execute

Text Segment

Address	Code	Basic	Source
0x00400000	0x3c011001	lui $1,4097	6: la $t1, A #we load the address of A into register t1
0x00400004	0x34290000	ori $9,$1,0	
0x00400008	0x3c011001	lui $1,4097	7: la $t2, B #we load the address of B into register t2
0x0040000c	0x342a0014	ori $10,$1,20	
0x00400010	0x20170005	addi $23,$0,5	8: addi $s7,$0,5 #this corresponds the size
0x00400014	0x00007820	add $15,$0,$0	9: add $t7,$0,$0 #this is the loop counter i
0x00400018	0x01f7a82a	slt $21,$15,$23	12: slt $s5,$t7,$s7 #if i<size (t7<s7) it sets s5=1 otherwise s5 =0
0x0040001c	0x12a0000d	beq $21,$0,13	13: beq $s5,$zero,finish #if not then program finish
0x00400020	0x000f4080	sll $8,$15,2	14: sll $t0,$t7,2 #we store in t0 =i*4 for array A
0x00400024	0x000f9880	sll $19,$15,2	15: sll $s3,$t7,2 #we store in t0 =i*4 for array B
0x00400028	0x01094020	add $8,$8,$9	16: add $t0,$t0,$t1 # address of Array A[i]
0x0040002c	0x026a6020	add $12,$19,$10	17: add $t4,$s3,$t2 # address of Array B[i]

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	2	3	4	5	2	3	3
0x10010020	5	5	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run I/O

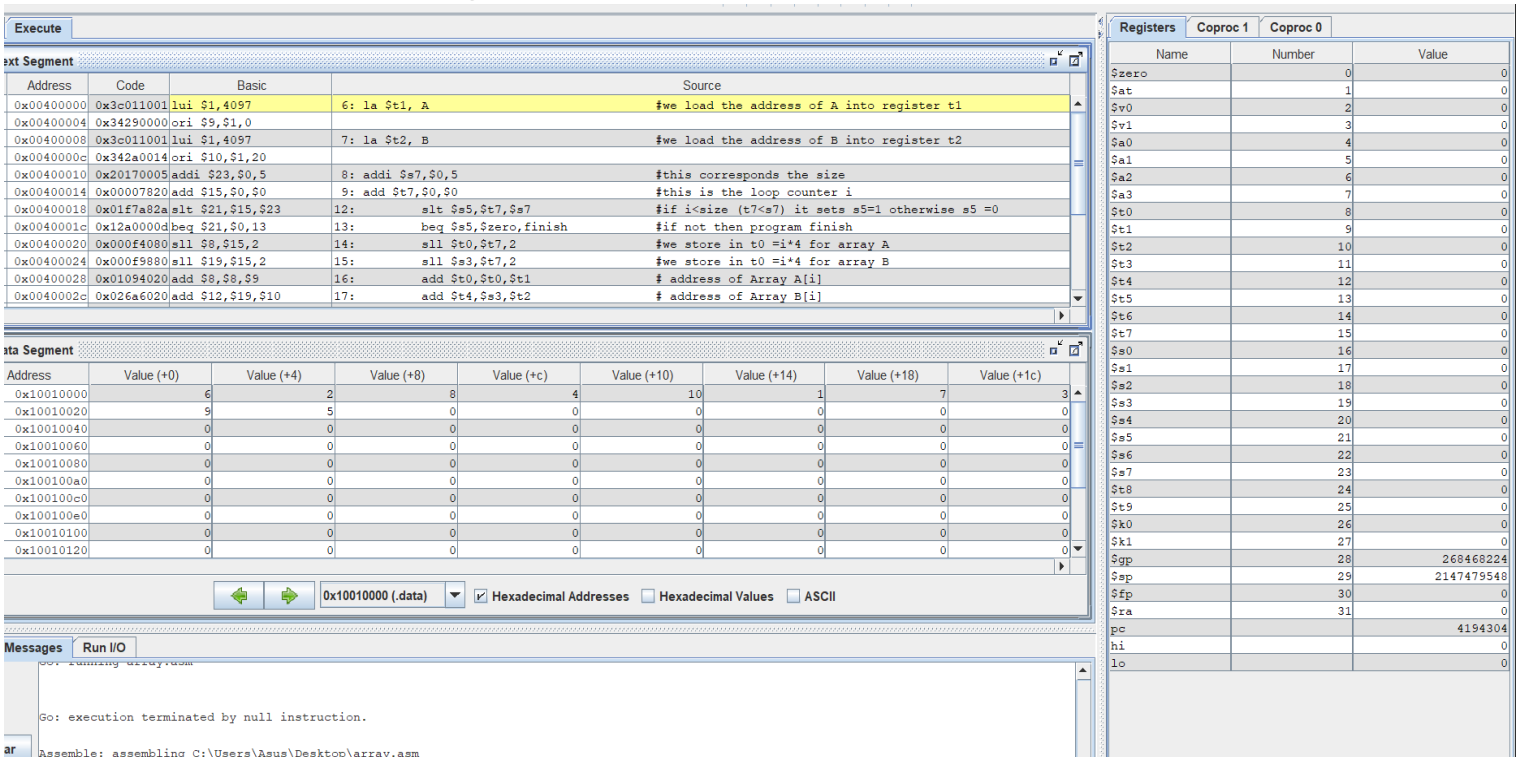
```
-- program is finished running (dropped off bottom) --
```

Registers Coproc 1 Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268501008
$t1	9	268500992
$t2	10	268501012
$t3	11	5
$t4	12	268501028
$t5	13	5
$t6	14	0
$t7	15	5
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	16
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	5
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194388
hi		0
lo		0

TEST 2

Before the program starts.



Execute

Text Segment

Address	Code	Basic	Source
0x00400000	0x3c011001	lui $1,4097	6: la $t1, A #we load the address of A into register t1
0x00400004	0x34290000	ori $9,$1,0	
0x00400008	0x3c011001	lui $1,4097	7: la $t2, B #we load the address of B into register t2
0x0040000c	0x342a0014	ori $10,$1,20	
0x00400010	0x20170005	addi $23,$0,5	8: addi $s7,$0,5 #this corresponds the size
0x00400014	0x00007820	add $15,$0,$0	9: add $t7,$0,$0 #this is the loop counter i
0x00400018	0x01f7a82a	slt $21,$15,$23	12: slt $s5,$t7,$s7 #if i<size (t7<s7) it sets s5=1 otherwise s5 =0
0x0040001c	0x12a0000d	beq $21,$0,13	13: beq $s5,$zero,finish #if not then program finish
0x00400020	0x000f4080	sll $8,$15,2	14: sll $t0,$t7,2 #we store in t0 =i*4 for array A
0x00400024	0x000f9880	sll $19,$15,2	15: sll $s3,$t7,2 #we store in t0 =i*4 for array B
0x00400028	0x01094020	add $8,$8,$9	16: add $t0,$t0,$t1 # address of Array A[i]
0x0040002c	0x026a6020	add $12,$19,$10	17: add $t4,$s3,$t2 # address of Array B[i]

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	6	2	8	4	10	1	7	3
0x10010020	9	5	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run I/O

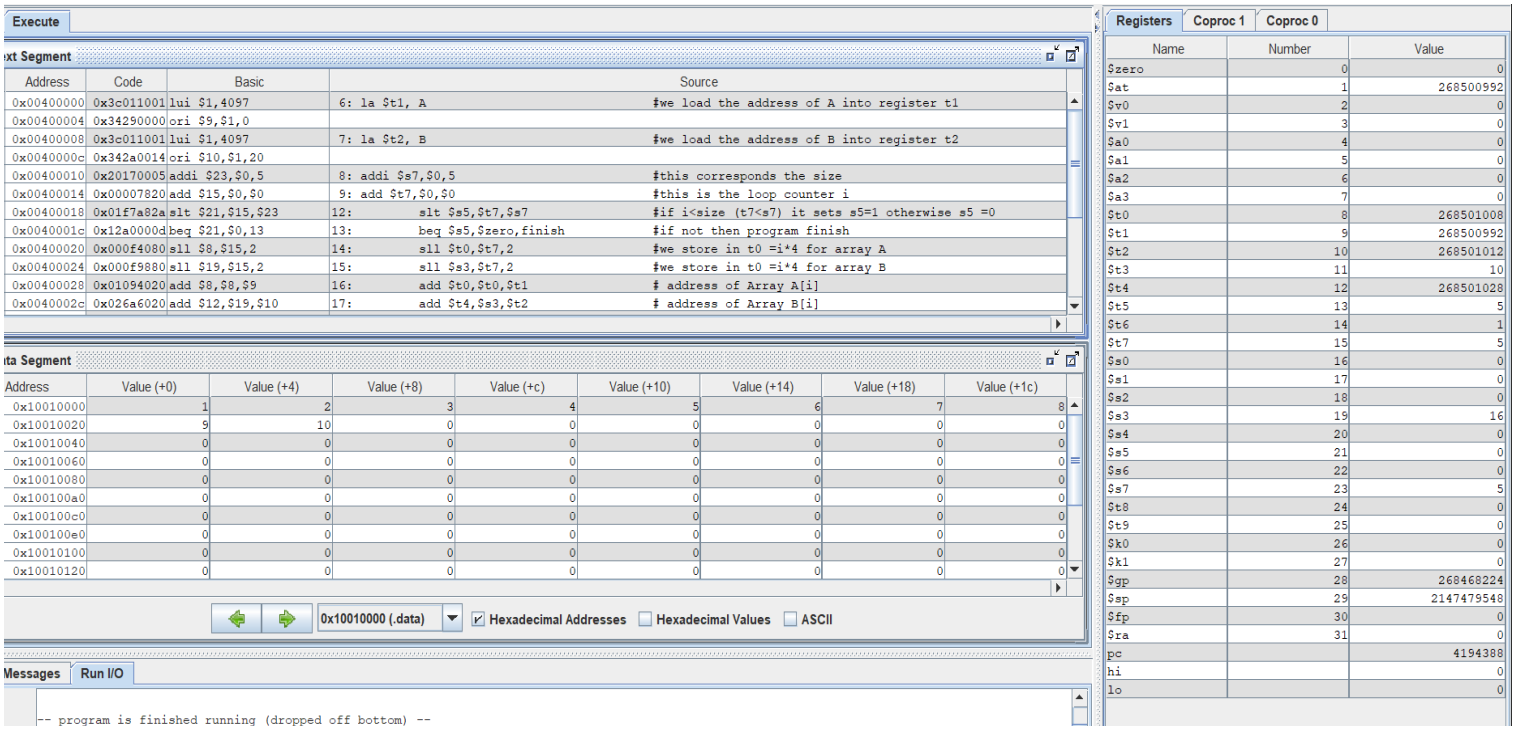
Go: execution terminated by null instruction.

Assemble: assembling C:\Users\Asus\Desktop\array.asm

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
$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
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

After the program executed.



Execute

Text Segment

Address	Code	Basic	Source
0x00400000	0x3c011001	lui $1,4097	6: la $t1, A #we load the address of A into register t1
0x00400004	0x34290000	ori $9,$1,0	
0x00400008	0x3c011001	lui $1,4097	7: la $t2, B #we load the address of B into register t2
0x0040000c	0x342a0014	ori $10,$1,20	
0x00400010	0x20170005	addi $23,$0,5	8: addi $s7,$0,5 #this corresponds the size
0x00400014	0x00007820	add $15,$0,$0	9: add $t7,$0,$0 #this is the loop counter i
0x00400018	0x01f7a82a	slt $21,$15,$23	12: slt $s5,$t7,$s7 #if i<size (t7<s7) it sets s5=1 otherwise s5 =0
0x0040001c	0x12a0000d	beq $21,$0,13	13: beq $s5,$zero,finish #if not then program finish
0x00400020	0x000f4080	sll $8,$15,2	14: sll $t0,$t7,2 #we store in t0 =i*4 for array A
0x00400024	0x000f9880	sll $19,$15,2	15: sll $s3,$t7,2 #we store in t0 =i*4 for array B
0x00400028	0x01094020	add $8,$8,$9	16: add $t0,$t0,$t1 # address of Array A[i]
0x0040002c	0x026a6020	add $12,$19,$10	17: add $t4,$s3,$t2 # address of Array B[i]

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	2	3	4	5	6	7	8
0x10010020	9	10	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Messages Run I/O

-- program is finished running (dropped off bottom) --

Registers Coproc 1 Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268501008
$t1	9	268500992
$t2	10	268501012
$t3	11	10
$t4	12	268501028
$t5	13	5
$t6	14	1
$t7	15	5
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	16
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	5
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194388
hi		0
lo		0

Problem 2:

Our second problem is about function calls. We are requested to write a MIPS code for the following C code fragment. In our code, we are not allowed to use multiplication instructions (mult or mul).

```
int main() {
int a; int b;
int result = 0;
if(a != b)
        result = compare(a, b);
else
        result = a+b;
return result;
}
int compare(int a, int b)
{
if(a<b)
        return punish(a, b);
else
        return award(a, b);
}
int punish(int a, int b)
{ return 2(a-b);}
int award(int a, int b)
{ return 2(a+b);}
```

I tested my program for the following input values:

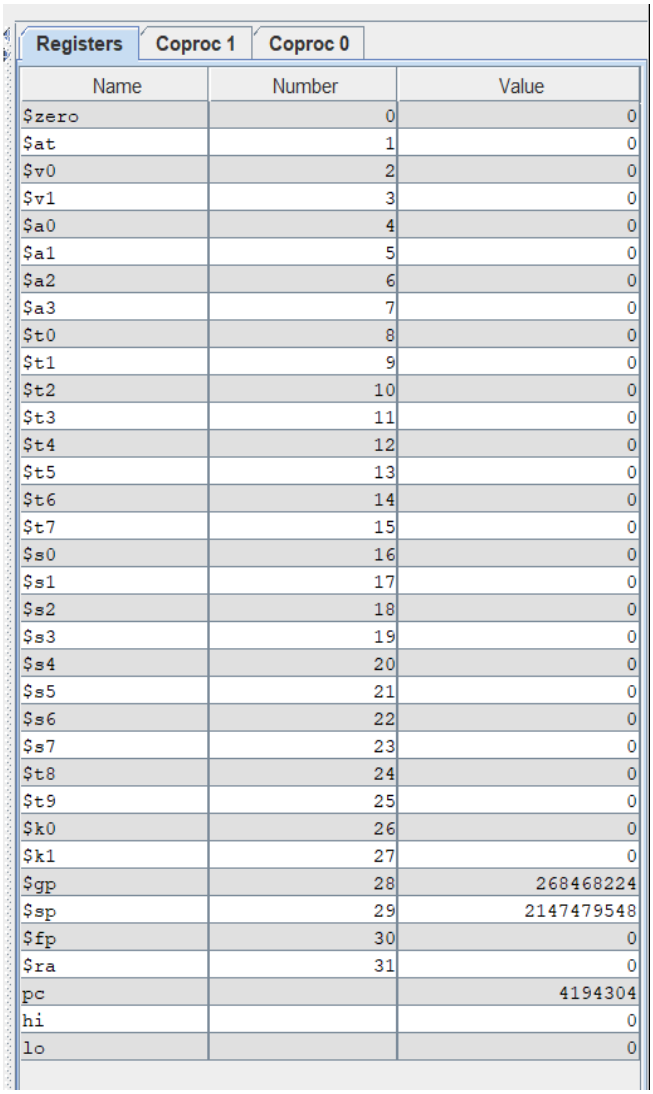
Test 1: a=3, b=3

Test 2: a=3, b=5

Test 3: a=5, b=3

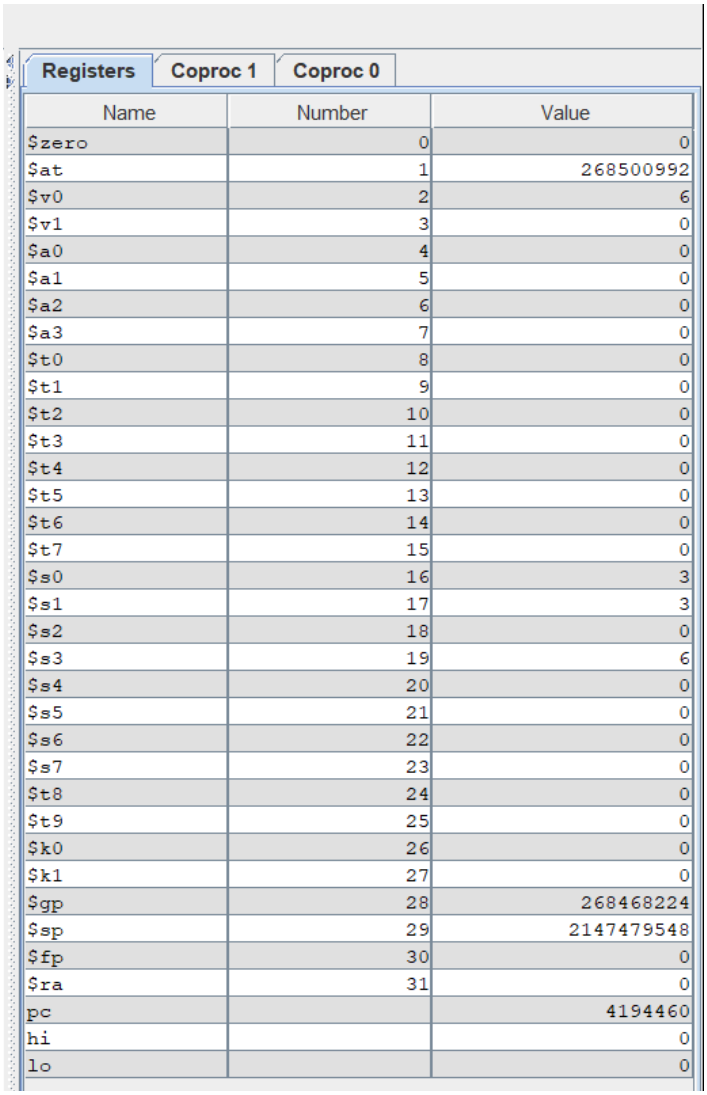
For each test, I saved the screenshots of the registers ($s0=a, $s1=b, $s3=result) before running the code and screenshots of the registers after running the code. You can see below the screenshot and also, I will explain the stack implementation below.

TEST 1



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
$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
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0



Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	6
$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	3
$s1	17	3
$s2	18	0
$s3	19	6
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194460
hi		0
lo		0

Before the program starts.

After the execution

$s0=3, $s1=3, $s3=result is 6 saved in register s3

I did not change the $s0 and $s1 registers instead of changing the values I used temporary registers while doing arithmetic operations

TEST 2

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
$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
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	-4
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	6
$t2	10	10
$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	-4
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194324
pc		4194456
hi		0
lo		0

Before the program starts.

After the execution

$s0=3, $s1=5, $s3=result is -4 saved in register s3

I did not change the s0 and s1 instead of changing the values I used temporary registers while doing arithmetic operations

TEST 3

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
$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
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194304
hi		0
lo		0

Before the program starts.

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	16
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	1
$t1	9	10
$t2	10	6
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	3
$s2	18	0
$s3	19	16
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194324
pc		4194456
hi		0
lo		0

After the execution

$s0=5, $s1=3, $s3=result is 16 saved in register s3

I did not changed the s0 and s1 instead of changing the values I used temporary registers while doing arithmetic operations

Using stack in mips program

The stack pointer is in register `$sp`.`$sp` contains the address of the top of the stack. When main function calls compare function using jal instruction it saves the return address into $ra register and when compare function calls award function or punish function, it uses jal instruction also and it overwrites the $ra register and we will lose the return address of compare function So calling the function will already destroy the $ra register. To avoid that we simply saved the $ra register into the stack. When we want to save the return address into stack we simply allocate memory and saved the $ra register into it as a result of this we will not lose the information in $ra register.