HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING BBM234

Name : Mehmet Taha

Surname : Usta

Number : 21527472

E~mail : b21527472@cs.hacettepe.edu.tr

Subject : Learning how to write and simulate MIPS code

Programming Language : MIPS ASSEMBLY

1.Explanation of the problem

Define the problem array and reach the elements in the array in the first question. Then create a for loop to access the elements of the array. Obtaining results using if-else statement. Multiplication instructions are prohibited, so the multiplication instructions must define with the codes

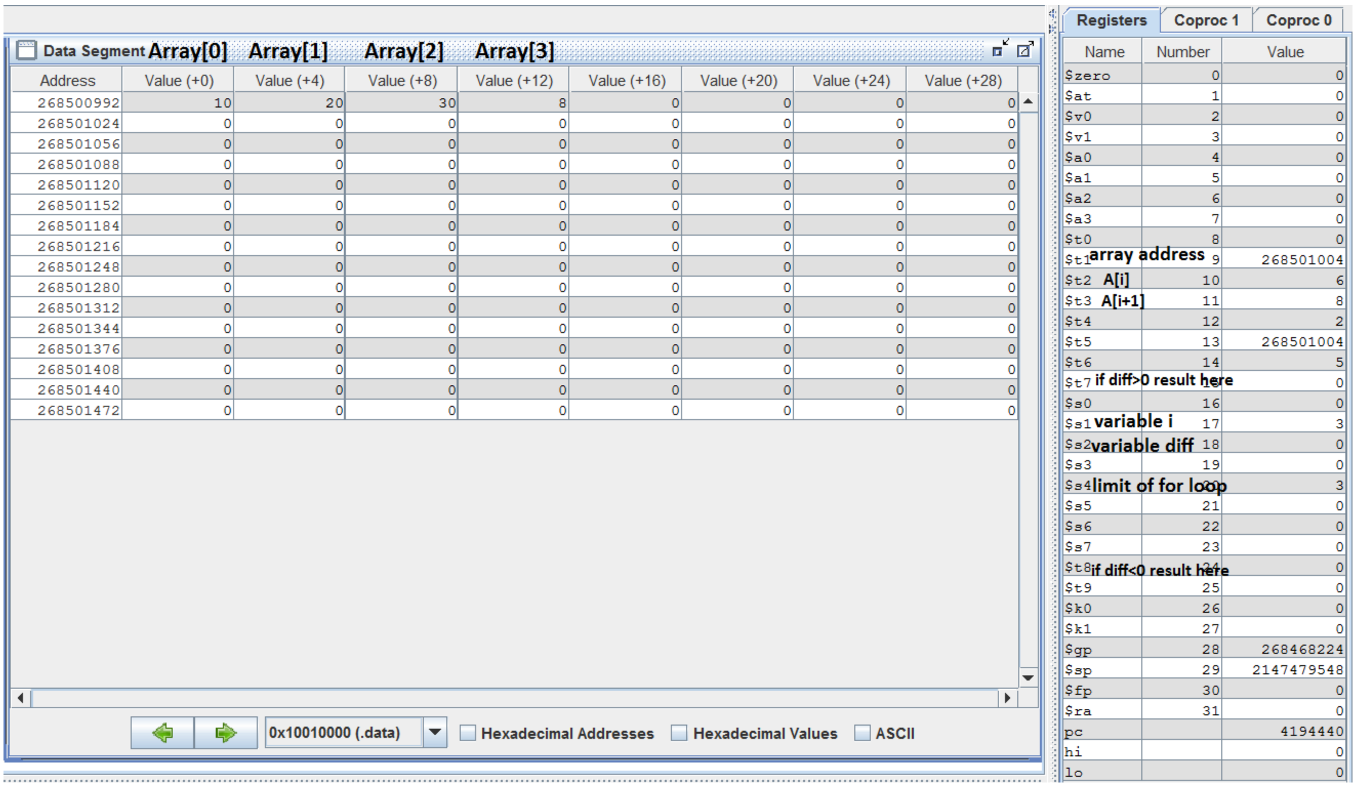
The second problem is that the defined functions are mutually connected

create and use stack structure to prevent loss of $ra between functions

2.Results

2.1)Array results

Test 1: A={2,4,6,8}



Data Segment Array[0] Array[1] Array[2] Array[3]

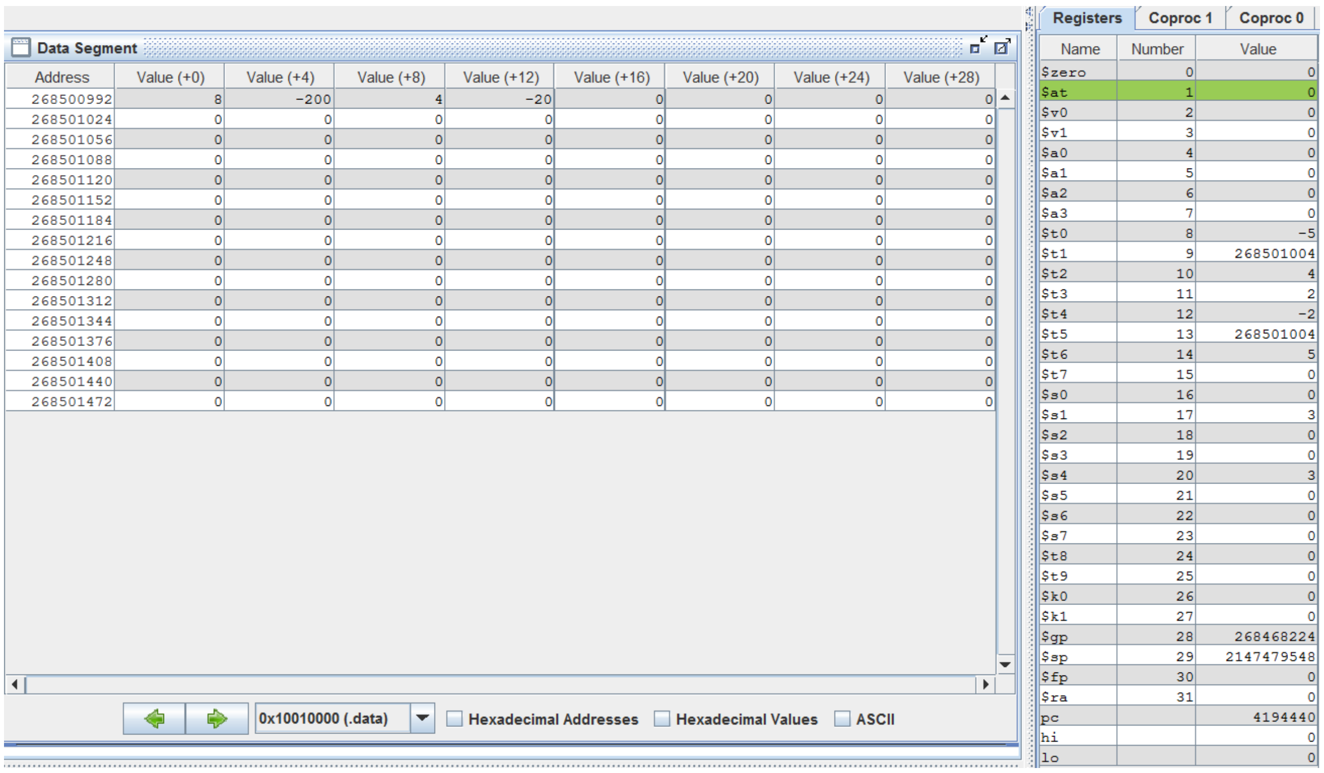
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	10	20	30	8	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	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 array address	9	268501004
$t2 A[i]	10	6
$t3 A[i+1]	11	8
$t4	12	2
$t5	13	268501004
$t6	14	5
$t7 if diff>0 result here	15	0
$s0	16	0
$s1 variable i	17	3
$s2 variable diff	18	0
$s3	19	0
$s4 limit of for loop	20	3
$s5	21	0
$s6	22	0
$s7	23	0
$t8 if diff<0 result here	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194440
hi		0
lo		0

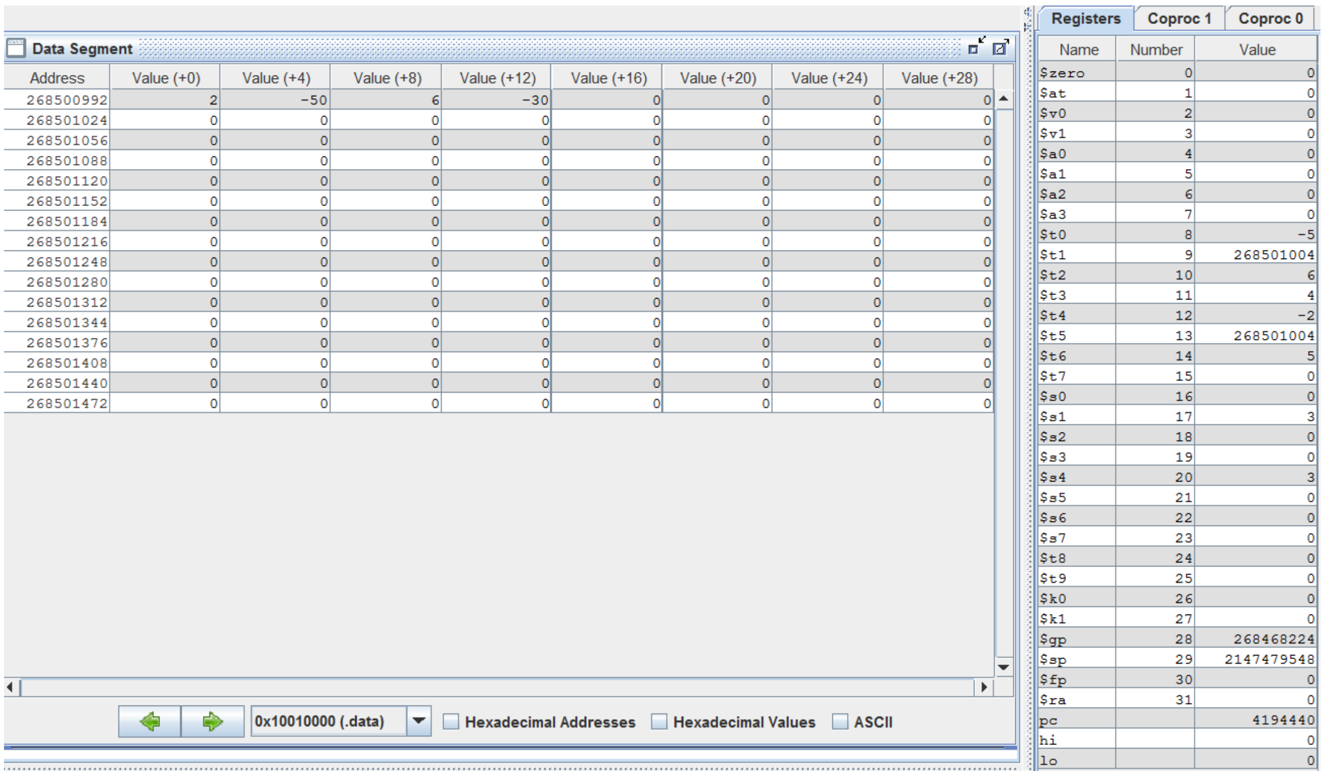
Test 2: A={8,6,4,2}



Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	8	-200	4	-20	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
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	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	-5
$t1	9	268501004
$t2	10	4
$t3	11	2
$t4	12	-2
$t5	13	268501004
$t6	14	5
$t7	15	0
$s0	16	0
$s1	17	3
$s2	18	0
$s3	19	0
$s4	20	3
$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		4194440
hi		0
lo		0

Test 3: A={2,2,6,4}

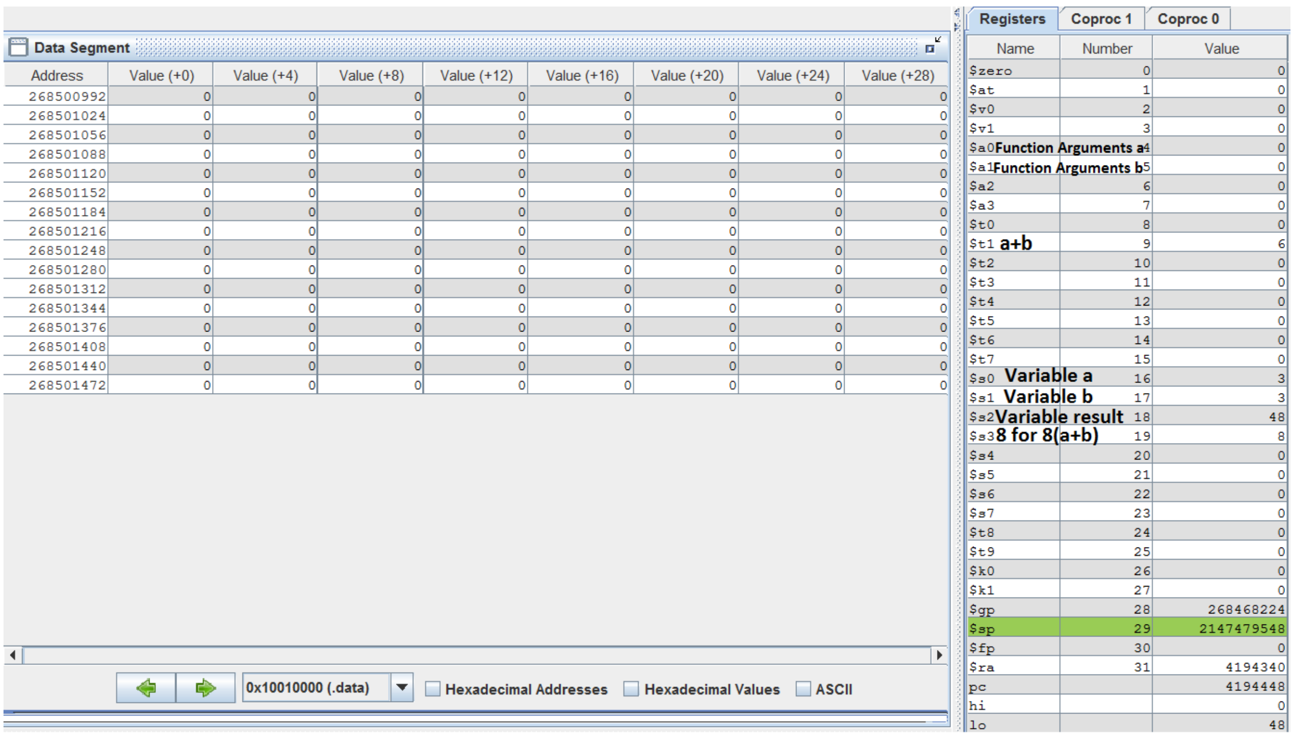


Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	2	-50	6	-30	0	0	0	0
268501024	0	0	0	0	0	0	0	0
268501056	0	0	0	0	0	0	0	0
268501088	0	0	0	0	0	0	0	0
268501120	0	0	0	0	0	0	0	0
268501152	0	0	0	0	0	0	0	0
268501184	0	0	0	0	0	0	0	0
268501216	0	0	0	0	0	0	0	0
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	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	-5
$t1	9	268501004
$t2	10	6
$t3	11	4
$t4	12	-2
$t5	13	268501004
$t6	14	5
$t7	15	0
$s0	16	0
$s1	17	3
$s2	18	0
$s3	19	0
$s4	20	3
$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		4194440
hi		0
lo		0

2.2)Function calls results

Test 1: a=3, b=3



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
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

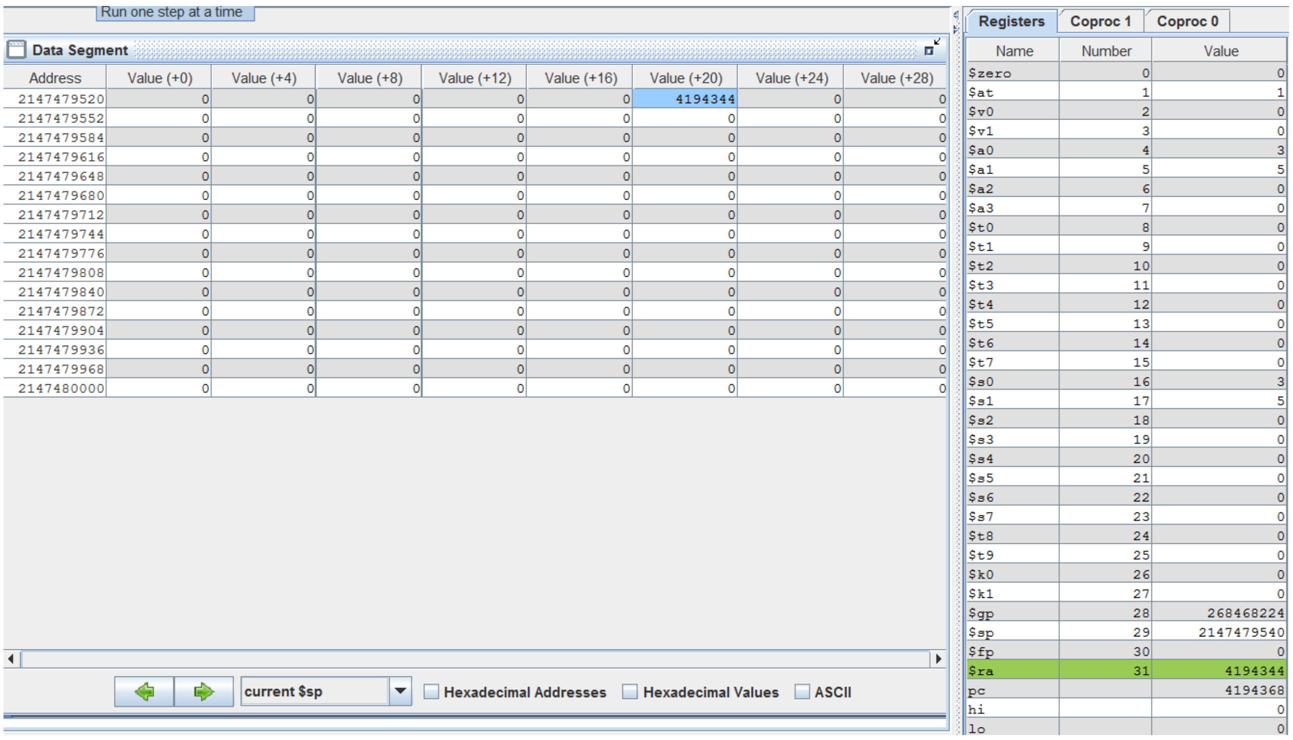
0x10010000 (.data) ☐ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0 **Function Arguments a**	4	0
$a1 **Function Arguments b**	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1 **a+b**	9	6
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0 **Variable a**	16	3
$s1 **Variable b**	17	3
$s2 **Variable result**	18	48
$s3 **8 for 8(a+b)**	19	8
$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	4194340
pc		4194448
hi		0
lo		48

Test 2: a=3, b=5

When the return address($ra) stack pointer is stored



Run one step at a time

Data Segment

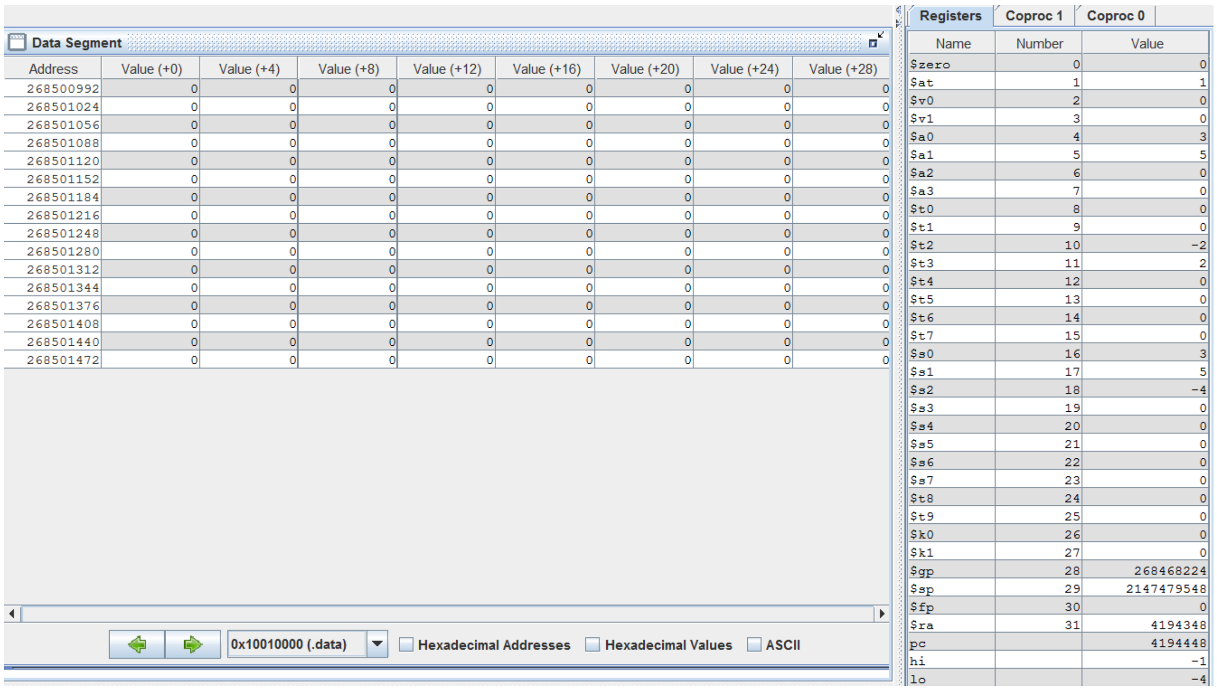
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
2147479520	0	0	0	0	0	4194344	0	0
2147479552	0	0	0	0	0	0	0	0
2147479584	0	0	0	0	0	0	0	0
2147479616	0	0	0	0	0	0	0	0
2147479648	0	0	0	0	0	0	0	0
2147479680	0	0	0	0	0	0	0	0
2147479712	0	0	0	0	0	0	0	0
2147479744	0	0	0	0	0	0	0	0
2147479776	0	0	0	0	0	0	0	0
2147479808	0	0	0	0	0	0	0	0
2147479840	0	0	0	0	0	0	0	0
2147479872	0	0	0	0	0	0	0	0
2147479904	0	0	0	0	0	0	0	0
2147479936	0	0	0	0	0	0	0	0
2147479968	0	0	0	0	0	0	0	0
2147480000	0	0	0	0	0	0	0	0

current $sp ☐ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	1
$v0	2	0
$v1	3	0
$a0	4	3
$a1	5	5
$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	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
$gp	28	268468224
$sp	29	2147479540
$fp	30	0
$ra	31	4194344
pc		4194368
hi		0
lo		0

when the program is finished



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
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	0	0

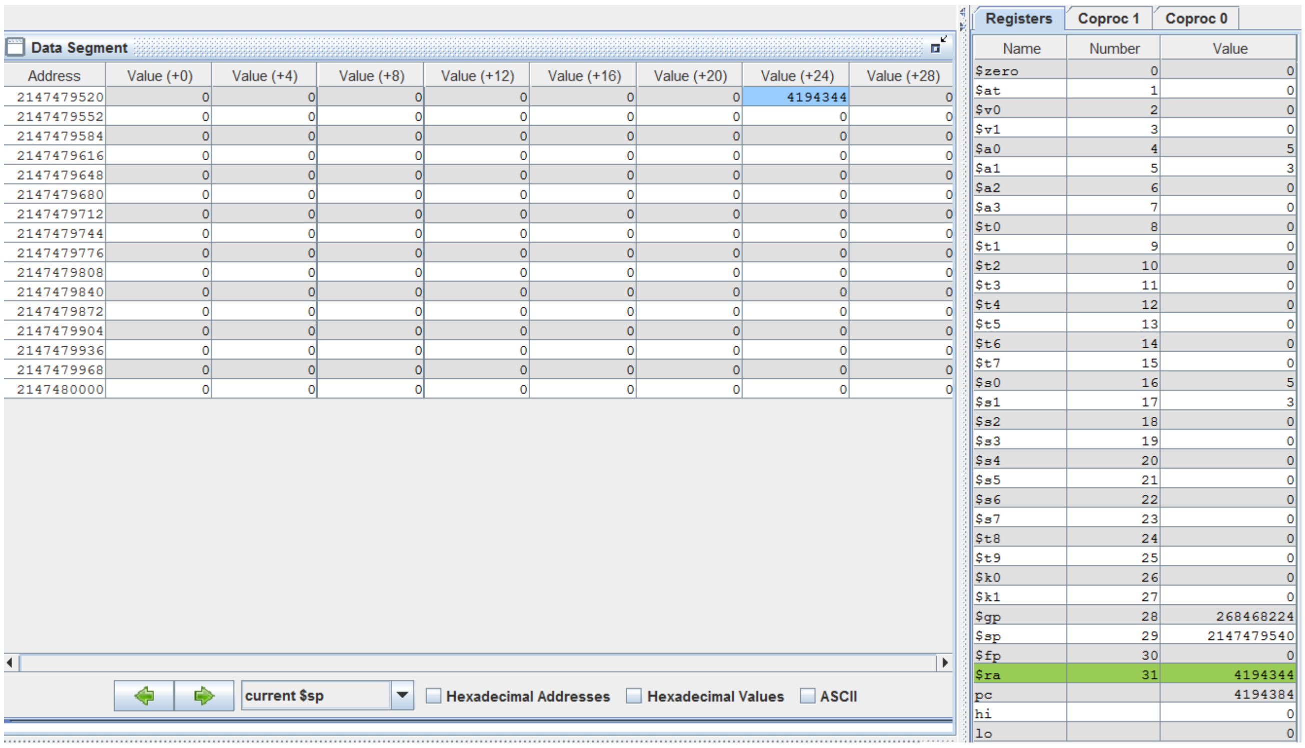
0x10010000 (.data) ☐ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	1
$v0	2	0
$v1	3	0
$a0	4	3
$a1	5	5
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	-2
$t3	11	2
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	3
$s1	17	5
$s2	18	-4
$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	4194348
pc		4194448
hi		-1
lo		-4

Test 3: a=5, b=3

When the return address($ra) stack pointer is stored



Data Segment

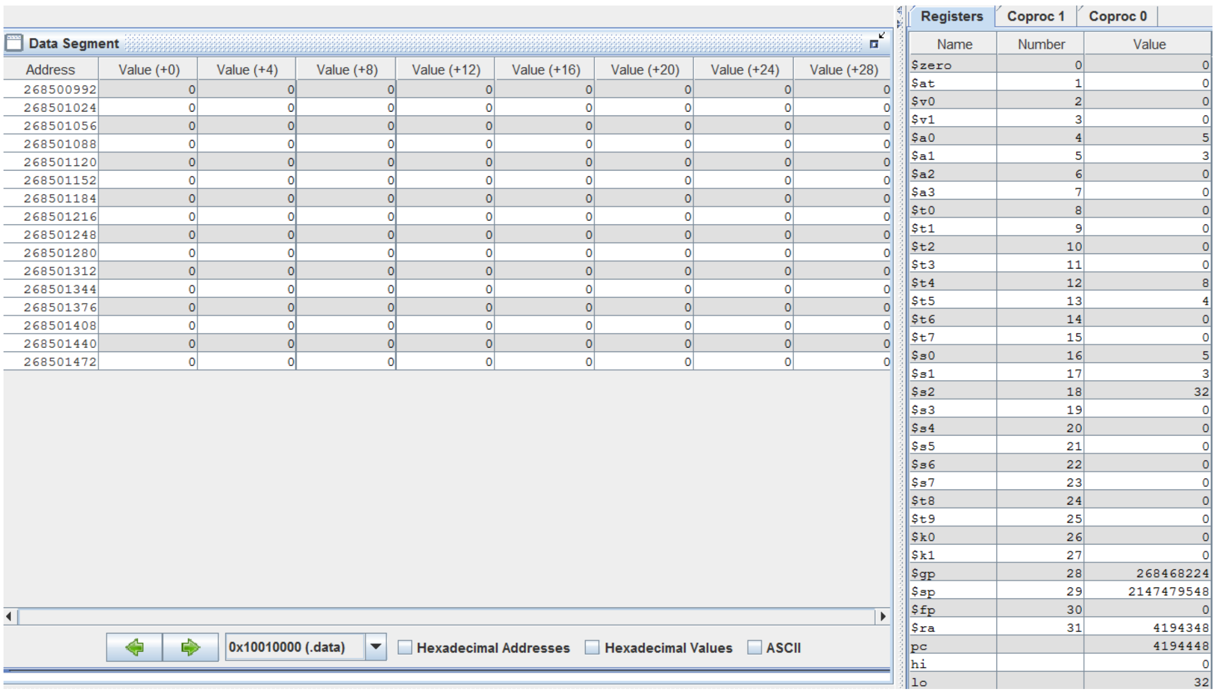
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
2147479520	0	0	0	0	0	0	4194344	0
2147479552	0	0	0	0	0	0	0	0
2147479584	0	0	0	0	0	0	0	0
2147479616	0	0	0	0	0	0	0	0
2147479648	0	0	0	0	0	0	0	0
2147479680	0	0	0	0	0	0	0	0
2147479712	0	0	0	0	0	0	0	0
2147479744	0	0	0	0	0	0	0	0
2147479776	0	0	0	0	0	0	0	0
2147479808	0	0	0	0	0	0	0	0
2147479840	0	0	0	0	0	0	0	0
2147479872	0	0	0	0	0	0	0	0
2147479904	0	0	0	0	0	0	0	0
2147479936	0	0	0	0	0	0	0	0
2147479968	0	0	0	0	0	0	0	0
2147480000	0	0	0	0	0	0	0	0

current $sp ☐ 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	5
$a1	5	3
$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	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
$gp	28	268468224
$sp	29	2147479540
$fp	30	0
$ra	31	4194344
pc		4194384
hi		0
lo		0

when the program is finished



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
268501248	0	0	0	0	0	0	0	0
268501280	0	0	0	0	0	0	0	0
268501312	0	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	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	5
$a1	5	3
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	8
$t5	13	4
$t6	14	0
$t7	15	0
$s0	16	5
$s1	17	3
$s2	18	32
$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	4194348
pc		4194448
hi		0
lo		32

3)Codes

3.1)Array codes

Codes first define array then define i and diff. for loop starts. A [i + 1] is subtracted from A [i] and diff is equal to this result. if diff is greater than 0 then A[i] = 5*A[i]. if diff is less than 0 then A[i+1] =- 5*A[i]. Multiplication is prohibited. defined 2 for loops in 2 different conditions(if-else) to describe multiplication

3.2)Function Calls codes

firstly I defined variables in main function. Then stack is allocated. I created if-else condition. if a and b are equal, the result is equal to 8 * (a + b). if not equal, else will call compare function. The compare function directs incoming function arguments as punish and award. if a is smaller than b , punish function will work. if a is not smaller than b, award function will work. The punish and award functions work after the $ra(return address) is stored in the stack. the stack is used to remember the return address in interconnected functions .After mathematical operations are done,the result will be saved to Mehmet