Clay Ramey

Dr. Tu

COMP-3350

04/01/24

Lab 2

Task 1:

Initializing registers.

```
1   .data
2   A: .word 21, 50, 63, 72, 0, 95, 11, 28, 4, 5, 16, 7
3
4   .text
5   .globl main
6
7   main:
8           la $s0,A    # array A
9           li $s1,12   # Length of array A
10          li $s2,1    # var i (initialized at 1)
11          li $s3,0    # var j
12          li $s4,0    # var v
13          li $t0,0    # address of A[i]
14          li $t1,0    # addres of A[j]
15          li $t2,0    # value of A[i]
16          li $t3,0    # value of A[j]
17
```

Main initializing description:

- $s0 – set to address of arr A
- $s1 – set to length of arr A = 12
- $s2 – Set to 1 to initializing var i

Next, the use of branches:
Loop 1 & Loop 2 branch:

```
Loop1:
        sll $t0, $s2, 2    # shift left obj in $s2 left by 2, puts result into $t0
        add $t0, $t0, $s0  # adds $t0 to $s0
        lw $s4, 0($t0)     # loads A[i] into v
        addi $s3, $s2, -1  # j = i-1
        sll $t1, $s3, 2    # shifts object in $s3 left by 2, puts result into $t1
        add $t1, $t1, $s0  # Adds $t1 to $s0 to get A[j] in $t1

Loop2:
        lw $t3, 0($t1)          # Adds 0 to $t1 to get an addy, puts val of A[j] -> $t3
        blt $t3, $s4, Break     # Branches to break if $t3 < $s4
        sw $t3, 4($t1)          # Adds 4 to $t1 to get addy of A[j+1] and stores in $t3
        addi $s3, $s3, -1       # --j
        addi $t1, $t1, -4       # keeps mem access consistent to j
        bge $s3, $zero, Loop2   # conditional branch to loop2 if $s3 >= 0
```

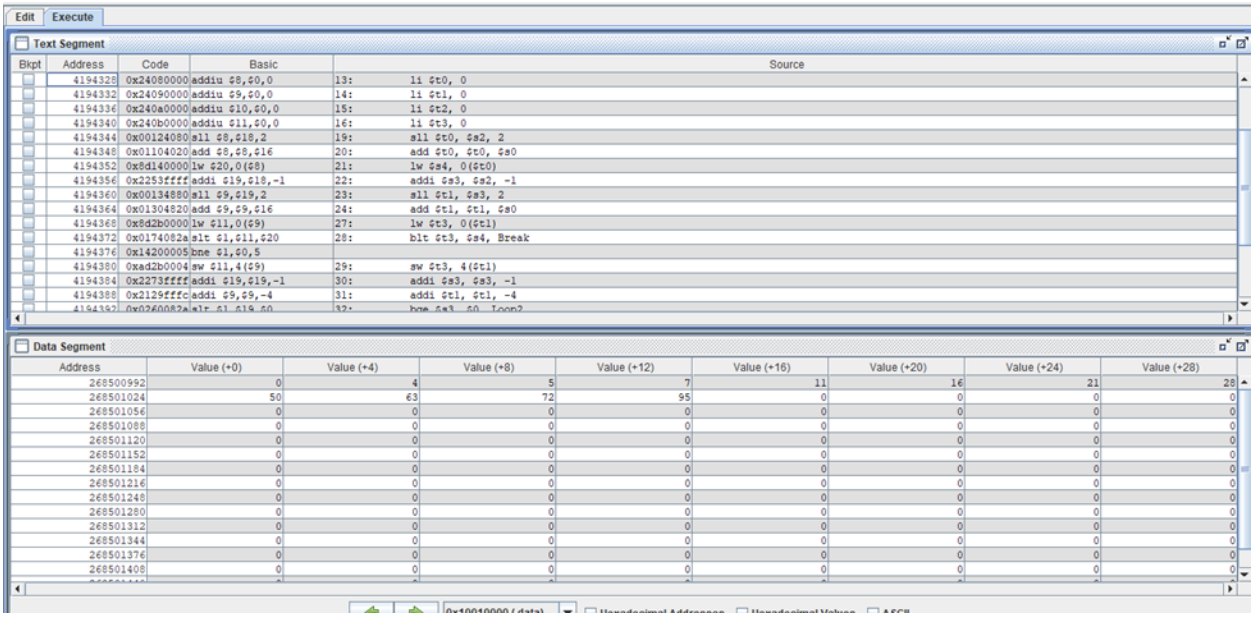
Break branch:

```
Break:
        sw $s4, 4($t1)        # A[j+1] = v
        addi $s2, $s2, 1      # ++i
        blt $s2, $s1, Loop1   # Branches to loop1 if $s2 < $s1


Exit:
        li $v0,10   # load exit op
        syscall     # exit

return 0;
```

Results after executing task 1:



Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268501036
$t1	9	268501000
$t2	10	0
$t3	11	5
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	268500992
$s1	17	12
$s2	18	12
$s3	19	2
$s4	20	7
$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		4194424
hi		0
lo		0

Task 2:

Logic for main:

```
  .data
  A:              .word 7, 42, 0 , 27, 16, 8, 4, 15, 31, 45


  .text
  .globl main


  main:
                  subu $sp, $sp, 4                # Make room for 1 register
                  sw $ra, 4($sp)                  # sets the stack pointer

                  la $a0, A
                  li $a1, 10

                  jal Sort                  # call the sort function

                  lw $ra, 4($sp)
                  addu $sp, $sp, 4


                  li $v0, 10                # Load exit op
                          syscall           # Exit
```

Sort & Swap procedure logic:

```
Sort:
        addi $sp, $sp, -20              # Make room for 5 reg
        sw $ra, 16($sp)                 # save (& sws from below)
        sw $s3, 12($sp)
        sw $s2, 8($sp)
        sw $s1, 4($sp)
        sw $s0, 0($sp)

        move $s2, $a0                   # Copy parameter $a0 in $s2 (saves $a0)
        move $s3, $a1                   # Copy the parameter $a1 in $s3 (saves $a1)

        move $s0, $zero         # i = 0
        addi $s0, $zero, 1      # i = 1

Loop1:  slt  $t0, $s0, $s3              # $t0 = 1 if $s0 < $s3 (i < n)
        beq $t0, $zero, Exit1   # Exit1 if $s0 >= $s3 (i >= n) ($t0 = 0)
        addi $s1, $s0, -1       # j = i - 1

Loop2:  slti $t0, $s1, 0                        # $t0 = 1 if $s1 < 0 (j < 0)
        bne $t0, $zero, Exit2           # Exit2 if $s1 < 0 (j < 0) ($t0 notequalto 0)
        sll $t1, $s1, 2                 # $t1 = j * 4
        add $t2, $s2, $t1               # $t2 = v + (j * 4)
        lw $t3, 0($t2)                  # $t3 = v[j]
        lw $t4, 4($t2)                  # $t4 = v[j +1]
        slt $t0, $t4, $t3               # $t0 = 0 if $t4 >= $t3
        beq $t0, $zero, Exit2           # Exit2 if $t4 >= $t3 ($t0 = 0)

        move $a0, $s2                   # 1st paramter of Swap is v
        move $a1, $s1                   # 2nd paramtere of Swap is j
        jal Swap                        # go to Swap code

        addi $s1, $s1, -1               # j -= 1
        j Loop2                         # jump to inner loop test

Exit2:  addi $s0, $s0, 1                # i += 1
        j Loop1                         # jump to outer test
```

```
Exit1:  lw $s0, 0 ($sp)         # restore $s0 from stack
        lw $s1, 4($sp)
        lw $s2, 8($sp)
        lw $s3, 12($sp)
        lw $ra, 16($sp)
        addi $sp, $sp, 20

        jr $ra  # return to calling routine
```

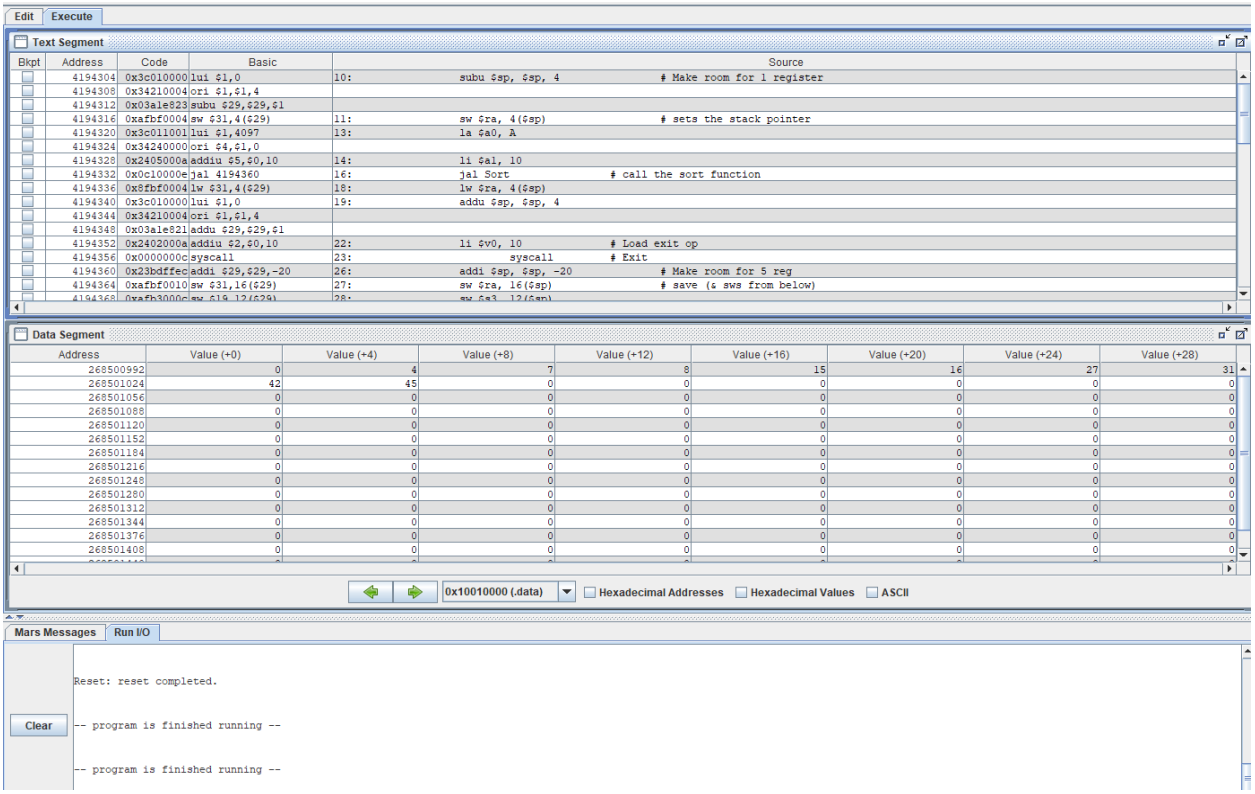
```
Swap:   sll $t1, $a1, 2         # $t1 = k *4
        add $t1, $a0, $t1       # $t1 = v + (K *4) (address of v[k])

        lw $t0, 0($t1)          # $t0 = v[k]
        lw $t2, 4($t1)          # $t2 = v[k + 1]

        sw $t2, 0($t1)          # v[k] = $t2
        sw $t0, 4($t1)          # v[k+1] = $t0

        jr $ra                  # return to calling routine
```

Results after executing task2:



Edit | Execute

Text Segment

Bkpt	Address	Code	Basic		Source	
	4194304	0x3c010000	lui $1,0	10:	subu $sp, $sp, 4	# Make room for 1 register
	4194308	0x34210004	ori $1,$1,4			
	4194312	0x03a1e823	subu $29,$29,$1			
	4194316	0xafbf0004	sw $31,4($29)	11:	sw $ra, 4($sp)	# sets the stack pointer
	4194320	0x3c011001	lui $1,4097	13:	la $a0, A	
	4194324	0x34240000	ori $4,$1,0			
	4194328	0x2405000a	addiu $5,$0,10	14:	li $a1, 10	
	4194332	0x0c10000e	jal 4194360	16:	jal Sort	# call the sort function
	4194336	0x8fbf0004	lw $31,4($29)	18:	lw $ra, 4($sp)	
	4194340	0x3c010000	lui $1,0	19:	addu $sp, $sp, 4	
	4194344	0x34210004	ori $1,$1,4			
	4194348	0x03a1e821	addu $29,$29,$1			
	4194352	0x2402000a	addiu $2,$0,10	22:	li $v0, 10	# Load exit op
	4194356	0x0000000c	syscall	23:	syscall	# Exit
	4194360	0x23bdffec	addi $29,$29,-20	26:	addi $sp, $sp, -20	# Make room for 5 reg
	4194364	0xafbf0010	sw $31,16($29)	27:	sw $ra, 16($sp)	# save (& sws from below)
	4194368	0xafb3000c	sw $19,12($29)	28:	sw $s3, 12($sp)	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	0	4	7	8	15	16	27	31
268501024	42	45	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

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Mars Messages | Run I/O

```
Reset: reset completed.

-- program is finished running --

-- program is finished running --
```

Clear

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	4
$v0	2	10
$v1	3	0
$a0	4	268500992
$a1	5	7
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	32
$t2	10	268501024
$t3	11	42
$t4	12	45
$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		4194360
hi		0
lo		0