Lab 1 Report

COMP3350

Adam Biggs – 8/29/2024

The purpose of this lab was to swap the position of 2 arbitrary elements in an array. Using the template provided I made modifications for readability and functionality. The following is a breakdown of my code and thought process.

Code Breakdown:

Establish Variables. I created an array and 2 integers (word). I also assigned their values. The words n, and m, are arbitrary and can be changed.

```
.data
A: .word 7, 42, 0, 27, 16, 8, 4, 15, 31, 45
n: .word 3
m: .word 6
```

Explanation:

name: .word value

Setup Main. This is just setup so the following code runs in main.

```
.text
.globl main
main:
```

Load Variables into registers. This just makes it so operations can be performed on the variables. $a0, $a1, and $a2 are all register addresses.

```
la $a0, A
lw $a1, n
lw $a2, m
```

la(load address), this is necessary because there are multiple elements to be stored

lw(load word) loads into register

Calculate offset for N. This takes the binary of word n and shifts its 1's left 2 spaces. And then it takes this new offset and adds it to the address of the array to find a specific element in the array. ($a1 = n)

```
sll $t1, $a1, 2          sll(shift left logical)
add $t1, $a0, $t1        adds a0 and t1 together and stores in t1
```

Calculate offset for M. This does the same thing for m. It just finds the address of the input in the array. ($a2 = m)

```
sll $t2, $a2, 2          sll(result, target, shift amount)
add $t2, $a0, $t2
```

Load the actual number of addresses in array. This converts and stores the address back into the actual number in it. For purposes of swapping them.

```
lw $t4, 0($t1)           lw(load word), 0(temp1) = 3
lw $t5, 0($t2)           lw(load word), 0,(temp2) = 6
```

Swap the elements. This literally takes the actual number we just set and swaps with the previous positions where the address was stored.

```
sw $t4, 0($t2)           sw(swap) from, 0(to)
sw $t5, 0($t1)
```

Result. The result of all this code is that the array will now read:

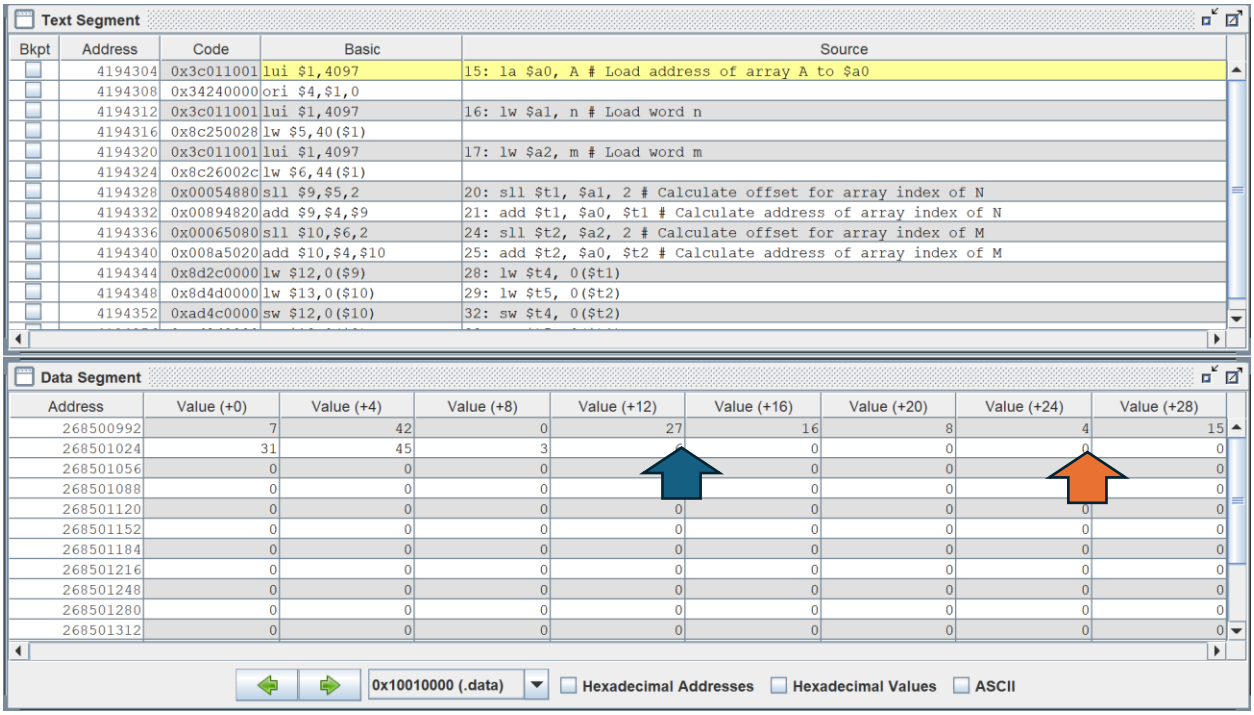
Original: 7, 42, 0, 27, 16, 8, 4, 15, 31, 45

Final: 7, 42, 0, 4, 16, 8, 27, 15, 31, 45

Screenshots Below

Screenshots Showing results:

Before Run: Notice the array values in the data segment



Text Segment

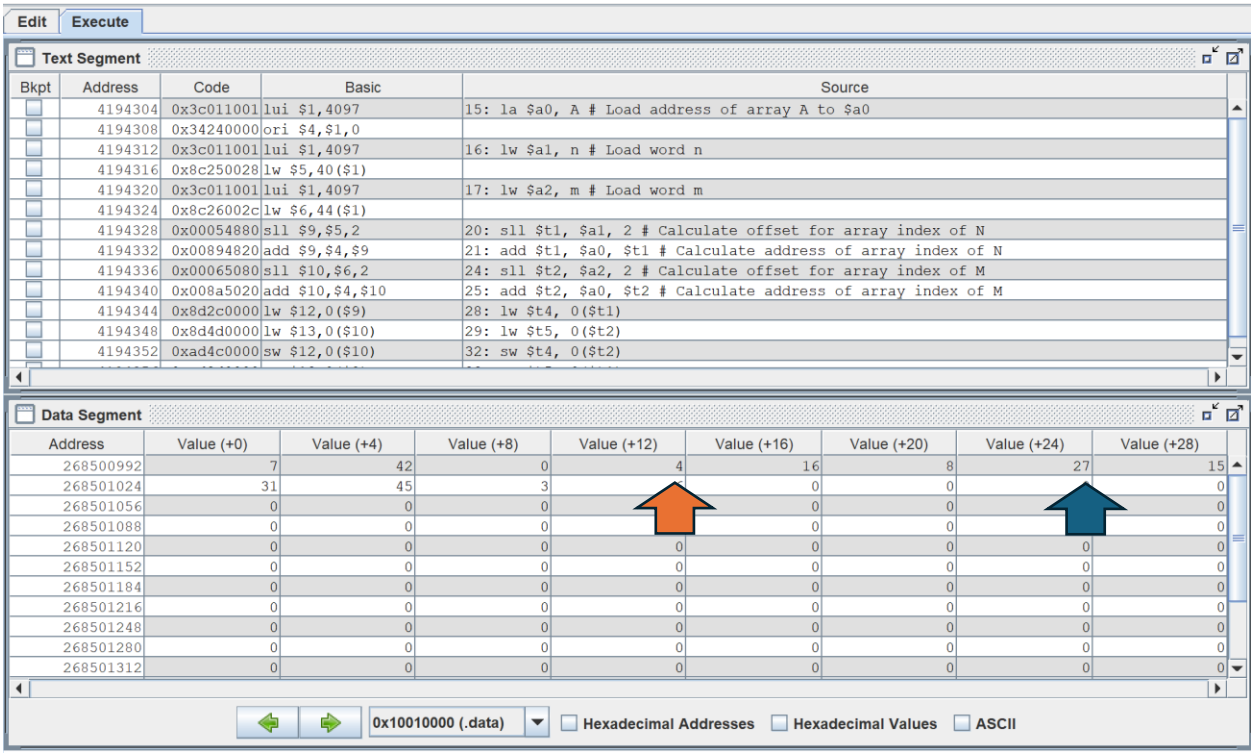
Bkpt	Address	Code	Basic	Source
	4194304	0x3c011001	lui $1,4097	15: la $a0, A # Load address of array A to $a0
	4194308	0x34240000	ori $4,$1,0	
	4194312	0x3c011001	lui $1,4097	16: lw $a1, n # Load word n
	4194316	0x8c250028	lw $5,40($1)	
	4194320	0x3c011001	lui $1,4097	17: lw $a2, m # Load word m
	4194324	0x8c26002c	lw $6,44($1)	
	4194328	0x00054880	sll $9,$5,2	20: sll $t1, $a1, 2 # Calculate offset for array index of N
	4194332	0x00894820	add $9,$4,$9	21: add $t1, $a0, $t1 # Calculate address of array index of N
	4194336	0x00065080	sll $10,$6,2	24: sll $t2, $a2, 2 # Calculate offset for array index of M
	4194340	0x008a5020	add $10,$4,$10	25: add $t2, $a0, $t2 # Calculate address of array index of M
	4194344	0x8d2c0000	lw $12,0($9)	28: lw $t4, 0($t1)
	4194348	0x8d4d0000	lw $13,0($10)	29: lw $t5, 0($t2)
	4194352	0xad4c0000	sw $12,0($10)	32: sw $t4, 0($t2)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	7	42	0	27	16	8	4	15
268501024	31	45	3	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

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

After Run: Notice the array values in the data segment changed



Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	4194304	0x3c011001	lui $1,4097	15: la $a0, A # Load address of array A to $a0
	4194308	0x34240000	ori $4,$1,0	
	4194312	0x3c011001	lui $1,4097	16: lw $a1, n # Load word n
	4194316	0x8c250028	lw $5,40($1)	
	4194320	0x3c011001	lui $1,4097	17: lw $a2, m # Load word m
	4194324	0x8c26002c	lw $6,44($1)	
	4194328	0x00054880	sll $9,$5,2	20: sll $t1, $a1, 2 # Calculate offset for array index of N
	4194332	0x00894820	add $9,$4,$9	21: add $t1, $a0, $t1 # Calculate address of array index of N
	4194336	0x00065080	sll $10,$6,2	24: sll $t2, $a2, 2 # Calculate offset for array index of M
	4194340	0x008a5020	add $10,$4,$10	25: add $t2, $a0, $t2 # Calculate address of array index of M
	4194344	0x8d2c0000	lw $12,0($9)	28: lw $t4, 0($t1)
	4194348	0x8d4d0000	lw $13,0($10)	29: lw $t5, 0($t2)
	4194352	0xad4c0000	sw $12,0($10)	32: sw $t4, 0($t2)

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	7	42	0	4	16	8	27	15
268501024	31	45	3	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

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Registers after run:

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	0
$v1	3	0
$a0	4	268500992
$a1	5	3
$a2	6	6
$a3	7	0
$t0	8	0
$t1	9	268501004
$t2	10	268501016
$t3	11	0
$t4	12	27
$t5	13	4
$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

Full Class:

```
1  # Adam Biggs 8/29/2024
2  # Lab 1 - Swap Elements
3
4  .data
5  A: .word 7, 42, 0, 27, 16, 8, 4, 15, 31, 45 #array
6  n: .word 3 #int n = 3 (can be changed to anything)
7  m: .word 6 #int m = 6
8
9  .text
10 .globl main
11
12 # Swap 2 elements, n, and m
13 main:
14
15 la $a0, A # Load address of array A to $a0
16 lw $a1, n # Load word n
17 lw $a2, m # Load word m
18
19 # Find Address in the Array for N
20 sll $t1, $a1, 2 # Calculate offset for array index of N
21 add $t1, $a0, $t1 # Calculate address of array index of N
22
23 # Find Address in the Array for M
24 sll $t2, $a2, 2 # Calculate offset for array index of M
25 add $t2, $a0, $t2 # Calculate address of array index of M
26
27 # Load actual number of address into temp 4, 5
28 lw $t4, 0($t1)
29 lw $t5, 0($t2)
30
31 # Swap Elements Position
32 sw $t4, 0($t2)
33 sw $t5, 0($t1)
```