Winston Shih
WXS190012
CS 2340.003

CS 2340 Assignment 2

Q1a. B[4]=A[8+8i];
f -> $s0, g -> $s1, h-> $s2, i -> $s3, j -> $s4
A->$s6, B->$s7

```
sll $t0, $s3, 3          #$t0=i*8
addi $t1, $t0, 8         #$t1=8i+8
sll $t2, $t1, 2          #$t2=(8i+8)*2^2 (offset for A[8+8i]
add $t3, $t2, $s6        #$t3=offset for A[8i+8]+base
lw $t4, 0($t3)           #$t4=A[8i+8]
sw $t4, 16($s7)          #B[4]=A[8i+8]
```

Q1b. B[4i] = A[8];
f -> $s0, g -> $s1, h-> $s2, i -> $s3, j -> $s4
A->$s6, B->$s7

```
lw $t0, 32($s6)          #$t0=A[8]
sll $t1, $s3, 2          #$t1=4*i
sll $t2, $t1, 2          #$t2=4i*2^2 (offset for B[4i])
add $t3, $t2, $s7        #$t3=offset for B[4i]+base
lw $t4, 0($t3)           #$t4=B[4i]
sw $t0, 0($t4)           #B[4i]=A[8]
```

Q1c. B[4i+4] = A[4g+2h] +h;
f -> $s0, g -> $s1, h-> $s2, i -> $s3, j -> $s4
A->$s6, B->$s7

```
sll $t0, $s1, 2          #$t0=4*g
sll $t1, $s2, 1          #t1=2*h
add $t0, $t0, $t1        #$t0=4g+2h
sll $t0, $t0, 2          #$t0=(4g+2h)*2^2 (offset for A[4g+2h])
add $t2, $t0, $s6        #$t2=offset for A[4g+2h]+base
lw $t3, 0($t2)           #$t3=A[4g+2h]
add $t3, $t3, $s2        #$t3=A[4g+2h]+h
sll $t4, $s3, 2          #$t4=4*i
addi $t4, $t4, 4         #$t4=4i+4
sll $t4, $t4, 2          #$t4=(4i+4)*2^2 (offset for B[4i+4])
add $t5, $t4, $s7        #$t5=offset for B[4i+4]+base
lw $t6, 0($t5)           #$t6=B[4i+4]
sw $t3 , 0($t6)          #B[4i+4]=A[4g+2h]+h
```

Q2.
$t0->i
$a0->base of save array

$a1->size of save array
add $t0, $zero, $zero
loop1: sll $t1, $t0, 2
add $t2, $a0, $t1
sw $zero, 0($t2)
addi $t0, $t0, 1
slt $t3, $t0, $a1
bne $t3, $zero, loop1

MIPS Machine Code

add $t0, $zero, $zero

This instruction creates i (associated with $t0 register) and sets its value to 0.
R-format is this instruction's instruction type.
op->add->op code 0
rs->$zero->reg 0
rt->$zero->reg 0
rd->$t0->reg 8
shamt->0 shift
funct->add->function code 32
Byte address=8000

op	rs	rt	rd	shamt	funct
0	0	0	8	0	32
000000	00000	00000	01000	00000	100000

loop1: sll $t1, $t0, 2

This instruction shifts register $t0 to left by 4 bits and stores the result of this operation into register $t1 as the offset of i.
The instruction type for this sll instruction is R-format/register addressing.
OP/funct->0/00 hex
op->sll->op code 0
rs->0
rt->$t0->reg 8
rd->$t1->reg 9
shamt->2
funct->sll->00 hex->function code 0
Byte address=8004

op	rs	rt	rd	shamt	funct
0	0	8	9	2	0
000000	00000	01000	01001	00010	000000

add $t2, $a0, $t1

The instruction adds the base of array save (associated with $a0 register) to offset(value of $t1 register) and stores it in $t2 register (associated with save array).

The instruction type for this add command is R-format/register addressing.
op->add->op code 0
rs->$a0->4
rt->$t1->reg 9
rd->$t2->reg 10
shamt->0 shift
funct->add->function code 0
Byte address: 8008

op	rs	rt	rd	shamt	funct
0	4	9	10	0	32
000000	00100	01001	01010	00000	100000

sw $zero, 0($t2)
This instruction stores the value of zero to save[i] (value associated with register $t2) based on index. As long as loop1 runs, every element of the save array will have a value of zero.
I-format/base addressing is the instruction type for sw.
op->26 hex->op code 43
rs->$t2->reg 10
rt->$zero->reg 0
Constant/address->0
Byte address: 8012

op	rs	rt	Constant/address
43	10	0	0
101011	01010	00000	0000000000000000

addi $t0, $t0, 1
This instruction increases the value of i (associated with $t0 register) by 1 and stores new result in register $t0.
I-format/immediate addressing is the instruction type for addi.
op->8 hex->op code 8
rs->$t0->reg 8
rt->$t0->reg 8
constant/address->1
Byte address: 8016

op	rs	rt	Constant/address
8	8	8	1
001000	01000	01000	0000000000000001

slt $t3, $t0, $a1

Instruction checks if i is less than the size of array save. If i is still less than 0, the value of 1 is assigned to register $t3, which means the conditional statement remains true. Otherwise, $t3 has the value of 0, which means the conditional becomes false.
Slt has an R-format/register addressing instruction type.
op->slt->op code 0
rs->$t0
rt->$a1
rd->$t3
shamt->0 shift
funct->slt->function code 42
Byte address: 8020

op	rs	rt	rd	shamt	funct
0	8	5	11	0	42
000000	01000	00101	01011	00000	101010

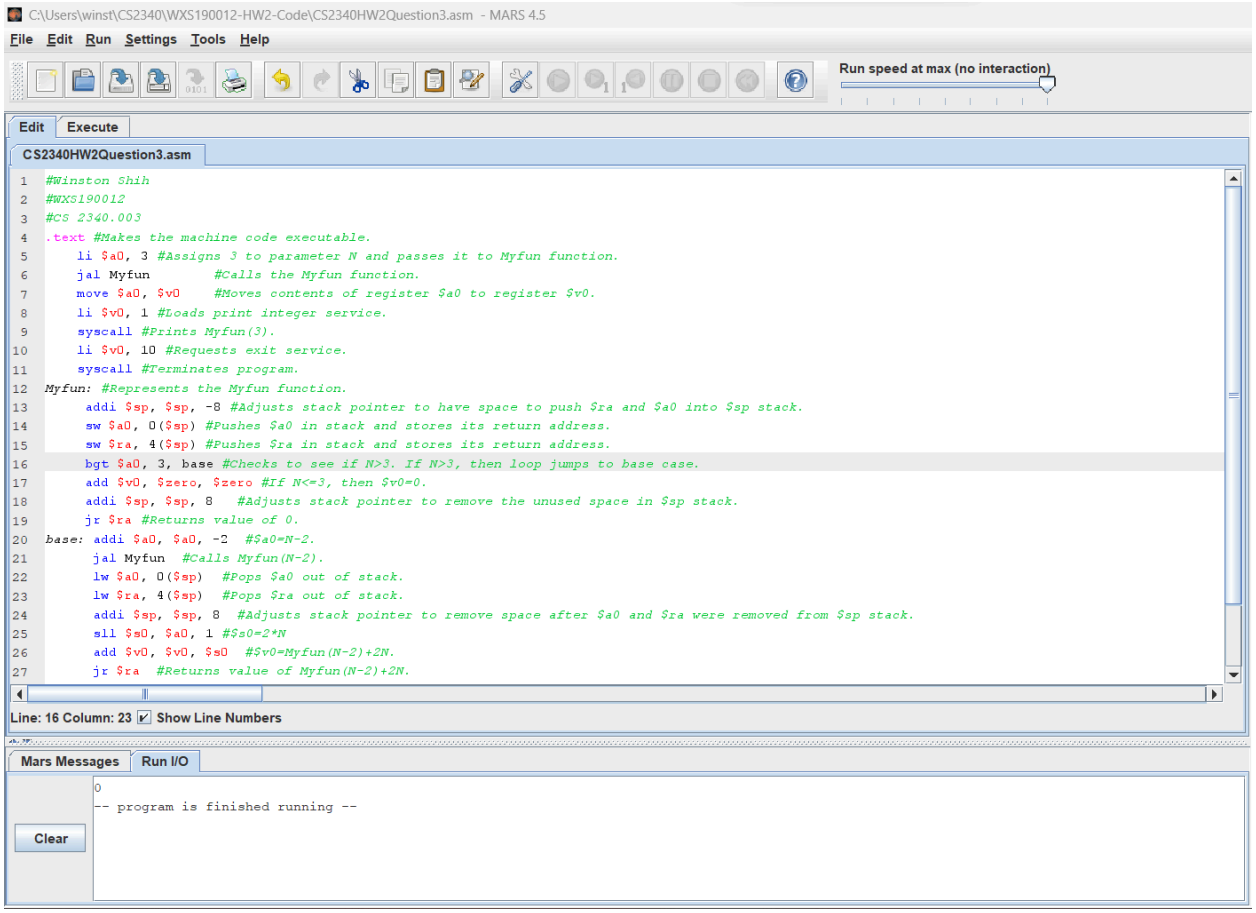
bne $t3, $zero, loop1
BNE statement determines if the value of $t3 is zero. If the value of $t3 is zero, i is not less than the size of the save array. The code ends after this statement because it does not branch back to loop1. If the value of $t3 is one, then code branches back to loop1 until i is equal to the size of the save array.
BNE has an I-format/PC-Relative Addressing instruction type.
op->bne->op code 5
rs->$t3
rt->$zero
Constant/address->loop1 address=relative address=0
Byte address: 8024

op	rs	rt	Constant/address
5	11	0	0
000101	001011	000000	0000000000000000

C loop:

```
for(int i=0;i<size;i++)
{
  save[i]=0;
}
```

Q3.



Q4.

CS2340HW2Question4.asm

```asm
#Winston Shih
#WXS190012
#CS 2340.003
.data #This represents the data section of the program.
    input1: .asciiz "Enter the size of array: " #Prompt to enter size of array.
    input2: .asciiz "Enter the elements: " #Prompt to insert elements into array.
    input3: .asciiz "Array after swapping: " #Prints message "Array after swapping: ".
    nl: .asciiz "\n" #Prints a new line.
    blank: .asciiz " " #Prints a blank space between two array eleements.
.text #Makes the MIPS machine code executable.
main: #Represents the main method of program.
    li $v0,4 #Loads print string service.
    la $a0, input1 #Loads address of prompt message for array size to register $a0.
    syscall #Prints prompt for array size.
    li $v0, 5 #Reads input for array size.
    syscall #Ensures read input for array size service is implemented.
    move $s0, $v0 #Stores size of array.
    sll $t0, $s0, 2#$t0=$s0*4 calculates array size
    li $v0, 9 #Allocates memory for array size.
    move $a0, $t0 #Moves memory for array size from $t1 to $a0.
    syscall #Ensures memory allocation service is implemented.
    move $s1, $v0 #Stores base address of array.
    li $v0, 4 #Loads print string service.
    la $a0, nl #Loads address for nl.
    syscall #Prints a new line in output.
    li $v0, 4  #Prints array element prompt.
    la $a0, input2 #Loads address of array element prompt to $a0.
```

CS2340HW2Question4.asm

```
28      syscall #Ensures service to prompt user for array elements is implemented.
29      move $t1, $s1 #Initializes for loop to the first element of array.
30      add $t2, $zero, $zero #Creates i and initializes to value of 0.
31      forloop1: slt $s2, $t2, $s0 #Checks to see if i<n. If true, $s2=1. Otherwises, $s2=0.
32                beq $s2, $zero, exit1 #For loop ends if $s2=0.
33                li $v0, 5   #Compiler reads input for array elements.
34                syscall #Requests read integer service for array elements.
35                sw $v0, 0($t1) #Compiler stores input value at given i.
36                addi $t1, $t1, 4 #For loop moves to next element in array a.
37                addi $t2, $t2, 1 #i is increased by 1.
38                j forloop1 #Repeats loop to input another element.
39      exit1: sub $t2, $t2, $t2 #Resets i back to 0
40      outerforloop: slt $s3, $t2, $s0 #Checks to see if i<n. If it is true, $s3=1. Else, $s3=0.
41                    beq $s3, $zero, exit2  #If $s3=0, then loop branches to exit2
42                    addi $t3, $t2, 1  # Declares j and setting it equal to i plus 1.
43      innerforloop: slt $s4, $t3, $s0 #Checks to see if j<n. If j<n, $s4=1. If j>=n, then $s4=0.
44                    beq $s4, $zero, nexti #Redirects to nexti if $s4=0.
45                    sll $t4, $t2, 2  #Calculates offset for a[i] and stores it in $t4.
46                    add $t4, $s1, $t4 #Creates address of a[i] and stores it in $t4.
47                    sll $t5, $t3, 2  #Calculates offset for a[j] and stores in register $t5.
48                    add $t5, $s1, $t5  # Creates a[j]'s address and stores it in $t5.
49                    lw $t6, 0($t4)  # Loads address of a[i] into $t6
50                    lw $t7, 0($t5)  # Loads address of a[j] into $t7.
51                    bgt $t6, $t7, swap  # Checks to see if a[i]>a[j] and will branch if true.
52                    addi $t3, $t3, 1 #Increases j counter by 1 if a[i]<=a[j]
53                    j innerforloop #Redirects to beginning of innerforloop
54      swap: move $t8, $t6 #temp=a[i] moves a[i] into temporary register $t8.
```

CS2340HW2Question4.asm

```
54      swap: move $t8, $t6 #temp=a[i] moves a[i] into temporary register $t8.
55            move $t6, $t7 #a[i]=a[j] moves value of a[j] into $t6.
56            move $t7, $t8 #a[j]=temp moves value of temp variable to a[j].
57            sw $t6, 0($t4) #Stores a[j] into a[i].
58            sw $t7, 0($t5) #Stores a[i] into a[j].
59            j innerforloop # Redirects to beginning of the inner for loop.
60      nexti: addi $t2, $t2, 1 #i++ increases i's value by 1.
61             j outerforloop   # Redirects to beginning of the outer for loop.
62      exit2: li $v0, 4 #Loads print string service.
63             la $a0, nl #Loads address of nl to $a0 register.
64             syscall #Prints a new line.
65             li $v0, 4  #Loads print string service.
66             la $a0, input3 #Loads address of input3.
67             syscall  #Prints "Array after swapping: ".
68      move $s5, $s1 #Moves base address of array to $s5.
69      sub $t2, $t2, $t2 #Resets i back to 0.
70      forloop4: slt $t9, $t2, $s0 #Checks to see if i<n. If i<n, $t9=1. Else, $t9=0
71                beq $t9, $zero, exit3 #Checks to see if $t9=0. If $t9=0, it redirects to exit3.
72                lw $a0, 0($s5)  #Loads address of a[i]
73                li $v0, 1  #Loads print integer service
74                syscall #Prints a[i].
75                li $v0, 4 #Loads print string service.
76                la $a0, blank #Loads address for blank.
77                syscall  #Prints " " after a[i].
78                addi $s5, $s5, 4  #Makes for loop move onto the next element in array a.
79                addi $t2, $t2, 1 #Increases i by 1.
80                j forloop4 #Redirects to beginning of forloop4
81      exit3: li $v0, 10  #Loads service to terminate program
82             syscall   #Terminates program.
```

Mars Messages | Run I/O

Clear

```
Enter the size of array: 4

Enter the elements: 1
9
2
4

Array after swapping: 1 2 4 9
-- program is finished running --
```

Q5.

Edit | Execute

CS2340HW2Question5.asm

```
1  #Winston Shih
2  #WXS190012
3  #CS 2340.003
4  .data #.data represents the data section of the program.
5      str1: .asciiz "Geeks" #Stores "Geeks" into str1[100].
6      str2: .asciiz "World" #Stores "World" into str2[100]
7      print1: .asciiz "First string: " #Stores message "First string: ".
8      print2: .asciiz "Second string: " #Stores message "Second string: ".
9      print3: .asciiz "Concatenated string: " #Stores message "Concatenated string: ".
10     newLine: .asciiz "\n" #Stores "\n".
11     str3: .space 100 #Initializes char str3[100]
12 .text #Makes the machine code executable.
13 main: #Represents the main method of program.
14      li $v0, 4 #Loads print string service.
15      la $a0, newLine #Loads address for newline to register $a0.
16      syscall #Prints new line.
17      li $v0, 4 #Loads print string service.
18      la $a0, print1 #Loads address for print1 to register $a0
19      syscall #Prints "First string: "
20      li $v0, 4 #Loads print string service service
21      la $a0, str1 #Loads address for str1[100] to register $a0
22      syscall #Prints "Geeks".
23      li $v0, 4 #Loads print string service.
24      la $a0, newLine #Loads address for newline to register $a0.
25      syscall #Prints new line.
26      li $v0, 4 #Loads print string service.
27      la $a0, print2 #Loads address for print2 to register $a0.
28      syscall #Prints "Second string: "
29      li $v0, 4 #Loads print string service.
```

Edit | Execute

CS2340HW2Question5.asm

```
30      la $a0, str2 #Loads address for str2[100] to register $a0.
31      syscall #Prints "World".
32      add $t0, $zero, $zero #$t0=i=0
33      add $t1, $zero, $zero #$t1=j=0
34      while1: lb $t2, str1($t0) #Loads byte of str1[i] to $t2
35              beq $t2, $zero, exit1 #Checks to see if str1[i] equals '\0'. If it does, it branches to exit1.
36              sb $t2, str3($t1) #Store byte of str1[i] to str3[j].
37              addi $t0, $t0, 1 #Increases i counter by 1.
38              addi $t1, $t1, 1 #Increases j counter by 1.
39              j while1 #Redirects to beginning of while1 loop.
40      exit1: sub $t0, $t0, $t0 #i=i-i rests i counter back to 0.
41      while2: lb $t2, str2($t0) #Loads byte of str2[i] to $t2.
42              beq $t2, $zero, exit2 #Checks to see if str2[i] equals '\0'. If it does, it branches to exit2.
43              sb $t2, str3($t1) #Store byte of str2[i] to str3[j].
44              addi $t0, $t0, 1 #Increases i by 1.
45              addi $t1, $t1, 1 #Increases j by 1.
46              j while2 #redirects to beginning of while2 loop.
47      exit2: sb $zero, str3($t1) #Stores '\0' to str3[j]
48      li $v0, 4 #Loads print string service.
49      la $a0, newLine #Loads address for newline to register $a0.
50      syscall #Prints new line.
51      li $v0, 4 #Loads print string service.
52      la $a0, print3 #Loads address for print3 to register $a0.
53      syscall #Prints "Concatenated string: ".
54      li $v0, 4 #Loads print string service.
55      la $a0, str3 #Loads address for str3 to register $a0.
56      syscall #Prints concatenated result of "Geeks" and "World".
57      li $v0, 10 #Loads exit service.
58      syscall #Ends program.
```

Mars Messages | Run I/O

Clear

```
-- program is finished running --


First string: Geeks
Second string: World
Concatenated string: GeeksWorld
-- program is finished running --
```