Ramzi El-abdallah
CS 231
March 17, 2014

# Assignment 2

1) Signed System:
$s1 = 0x21AC = (0010 0001 1010 1100) = 8620
$s2 = 0x8C15 = (1000 1100 0001 0101) = -29675
$s3 = ?

a) add $s3, $s1, $s2

0010 0001 1010 1100 (8620)
+1000 1100 0001 0101 (-29675)
$s3 = 1010 1101 1100 0001 (-21055) = 0xADC1

b) slt $s3, $s1, $s2

If $s1<$s2 then $s3 is set to 1. Otherwise it is set to 0. In this case $s1 is greater than $s2, therefore $s3 = 0.

c) sub $s3, $s2, $s1 ----------> -29675 + (-8620)

0010 0001 1010 1100 (8620)
1101 1110 0101 0011 (1's compliment)
1 (add 1)
1101 1110 0101 0100 (-8620)
1000 1100 0001 0101 (-29675)
$s3 = 1 0110 1010 0110 1001 (-38295) = 0x16A69

2)

a) Y[10] = Y[3] + 8

```
la $s0, Y
lw $t1, 12($s0)
addi $t1, $t1, 8
sw $t1, 40($s0)
```

b) a = b – 5 (a = $t2, b = $t1)

```
li $t0, 5
sub $t2, $t1, $t0
```

c) k = 3*m + 2 (m = $t1, k = $t3)

```
li $t0, 3
mult $t0, $t1
mflo $t2
```

addi $t3, $t2, 2

d) a++ (a = $t0)
addi $t0, $t0, 1

3)

For the following code segment write the machine language representation of each instruction in binary. The instruction codes are add->32,beq->4,addi->8,
lw->35,j->2. Assume that Loop has the address of Ox4CB23

| | | | |
|---|---|---|---|
| Loop: | | | 0000 0000 0000 0100 1100 1011 0010 0011 (address of loop) |
| | beq | $t1, $t2, done | 000100-01001-01010-0000 0000 0000 0101 (4-9-10-5) |
| | lw | $s1, 0($t0) | 100011-01000-10001-0000 0000 0000 0000 (35-8-17-0) |
| | add | $s0, $s1, $s0 | 000000-10001-10000-10000-00000-100000 (0-17-16-16-0-32) |
| | addi | $t1, $t1, 1 | 000100-01001-01001-0000 0000 0000 0001 (8-9-9-1) |
| | addi | $t0, $t0, 4 | 000100-01000-01000-0000 0000 0000 0100 (8-8-8-4) |
| | j | Loop | 000010-00 0000 0100 1100 1011 0010 0011 (2-Ox4CB23) |
| done: | | | 0000 0000 0000 0100 1100 1011 0011 1111 (address of done) |

4)

“jal sub1” means jump and link “sub1” subroutine. Once that is called then the address of the next line of code after “jal sub1” is stored in $ra as the return address. The program then jumps to subroutine “sub1”, does what it needs to do and then when “jr $ra” is called, this is telling the pointer to go to the address that is in $ra, which is the address of where we left off in main.

**Programming Questions**

```
#Ramzi El-abdallah
#Assignment 2

        .data
                array: .space 80
                str0: .asciiz "\nPlease enter the size of your array (1-19): "
                str1: .asciiz "Sorry invalid entry.\n"
                str2: .asciiz "\nElement #"
                str3: .asciiz ": "
                str4: .asciiz "\n\nThe array in reverse order is:"
        .text

main:
        la $s3, array

        li $v0, 4
        la $a0, str0
        syscall

        jal readNum             #jumps to readNum subroutine
```

```
        addi $s0, $v0, 0        #stores input into $s0 after returning from readNum
        addi $a0, $s0, 0        #stroes array size into $a0 to send to verifySize

        jal verifySize          #jumps to verifySize subroutine
        addi $s1, $v0, 0        #true or false value from verifySize

        bne $s1, $0, proceed

        li $v0, 4
        la $a0, str1
        syscall

        j main

proceed:
        addi $a0, $s0, 0        #loads array size to pass
        jal createArray


        addi $a0, $s0, 0
        jal printArray

        addi $a0, $s0, 0
        jal reverseArray

        addi $a0, $s0, 0
        jal printArray

        j exits

readNum:
        li $v0, 5
        syscall                 #user inputs size of array

        jr $ra

verifySize:
        li $t1, 20              #maximum
        addi $t0, $a0, 0        #loads user input into $t0

        check1:
                bgt $t0, $0, check2     #checks to see if greater than 0

                addi $v0, $0, 0                 #returns false if less than 0

                jr $ra

        check2:
                blt $t0, $t1, returnTrue        #checks to see if less than 20
```

```
        addi $v0, $0, 0                 #returns false if less greater than 20

        jr $ra

    returnTrue:
        addi $v0, $t0, 1        #returns true if 0<input<20
        jr $ra

createArray:
    addi $t0, $0, 0         #count
    addi $t1, $a0, 0        #array size
    la $s3, array           #loads address of array into $s3

    sub $sp, $sp, 4
    sw $ra, 0($sp)          #saves return address to stack pointer

    makingArray:
        beq $t0, $t1, returnMain
        jal readNum
        addi $s2, $v0, 0        #saves input into $s2

        j checkNumPositive


    checkNumPositive:
        sgt $t4, $s2, $0        #sets $t4 to 1 if number is greater than 0

        bne $t4, $0, divisibleBy3

        li $v0, 4
        la $a0, str1
        syscall

        j makingArray

    divisibleBy3:
        li $t4, 3

        div $s2, $t4
        mfhi $t5

        seq $t6, $t5, $0

        beq $t6, $0, makingArray

        sw $s2, 0($s3)
        addi $t0, $t0, 1
        addi $s3, $s3, 4
```

```
            j makingArray

        returnMain:
            lw $ra 0($sp)
            addi $sp, $sp, 4
            jr $ra

reverseArray:
        addi $t0, $0, 1         #count
        addi $s0, $a0, 0        #array size
        la $t3, array           #address of array
        la $t4, array           #address of array
        li $t1, 4

        mult $s0, $t1
        mflo $t2
        sub $t2, $t2, 4
        add $t4, $t4, $t2
        loop:
        ble $t4, $t3, endReverse

        lw $t5, 0($t3)
        lw $t6, 0($t4)

        sw $t5, 0($t4)
        sw $t6, 0($t3)

        addi $t3, $t3, 4
        sub $t4, $t4, 4

        j loop

        endReverse:
            li $v0, 4
            la $a0, str4
            syscall

            jr $ra


printArray:
        addi $t0, $0, 1
        addi $s0, $a0, 0
        la $t3, array

        start:
        bgt $t0, $s0, returnMain1
        lw $t4, 0($t3)
```

```
        li $v0, 4
        la $a0, str2
        syscall

        li $v0, 1
        addi $a0, $t0, 0
        syscall

        li $v0, 4
        la $a0, str3
        syscall

        li $v0, 1
        addi $a0, $t4, 0
        syscall

        addi $t0, $t0, 1
        addi $t3, $t3, 4

        j start

        returnMain1:
        jr $ra

exits:
        li $v0, 10
        syscall
```