PRELIMINARY REPORT

Lab 02

MUHAMMAD ARHAM KHAN

21701848 - CS

SECTION 06

SPRING 2019

Dated: 02 March, 2019

INTERACT WITH USER

```
interactWithUser:
                #print prompt
                la      $a0, interactionPrompt
                li      $v0,4
                syscall
                #input integer
                li      $v0, 5
                syscall
                #menu selection
                beq     $v0, 3, returnToMain
                beq     $v0, 1, option1
                beq     $v0, 2, option2
                j       interactWithUser
        option1:
                #initialize registers
                add     $s0, $0, $0
                add     $s1, $0, $0
                add     $s2, $0, $0
                add     $s3, $0, $ra
                add     $s4, $0, $0
                #print prompt
                la      $a0, convertToDecPrompt
                li      $v0,4
                syscall
                #input string
                la      $a0, inputBuffer
                li      $a1, 30
                move    $s0,$a0
                li      $v0,8
                syscall
                #initialize stack
                addi    $sp, $sp, -20
                sw      $s0, 0($sp)
                sw      $s1, 4($sp)
                sw      $s2, 8($sp)
                sw      $s3, 12($sp)
                sw      $s4, 16($sp)
                #got to sub
                jal     convertToDec
                #reload stack
                lw      $s0, 0($sp)
                lw      $s1, 4($sp)
                lw      $s2, 8($sp)
                lw      $s3, 12($sp)
                lw      $s4, 16($sp)
                #print prompt
                move    $ra, $s3
                la      $a0, convertToDecOutputPrompt
                li      $v0,4
                syscall
                #print string outh
                move    $a0, $s4
                li      $v0,1
                syscall
                #print newline
                la      $a0, newline
                li      $v0,4
                syscall
                j       interactWithUser
        option2:
                #initialize registers
                add     $s0, $0, $0
                add     $s1, $0, $0
```

```
                add     $s2, $0, $0
                add     $s3, $0, $ra
                #input string
                la      $s4, storageBuffer
                la      $a0, reverseNumberPrompt
                li      $v0,4
                syscall
                #input integer
                li      $v0, 5
                syscall
                move    $s0, $v0
                #initlizing stack
                sw      $s0, 0($sp)
                sw      $s1, 4($sp)
                sw      $s2, 8($sp)
                sw      $s3, 12($sp)
                sw      $s4, 16($sp)
                #go to sub
                jal     reverseNumber
                #reload stack
                lw      $s0, 0($sp)
                lw      $s1, 4($sp)
                lw      $s2, 8($sp)
                lw      $s3, 12($sp)
                lw      $s4, 16($sp)
                #print prompt
                move    $ra, $s3
                la      $a0, reverseNumberOutputPrompt2
                li      $v0,4
                syscall
                #print string out
                move    $a0, $s4
                li      $v0,4
                syscall
                #print newline
                la      $a0, newline
                li      $v0,4
                syscall
                j       interactWithUser
returnToMain:
                jr      $ra
```

CONVERT TO DEC

```
convertToDec:
                #reload stack // initialize values
                lw      $s0, 0($sp)
                lw      $s1, 4($sp)
                lw      $s2, 8($sp)
                add     $s3, $0, $0
                lw      $s4, 16($sp)
                add     $s6, $0, $0
                findOctalSize:
                        lb      $s6, 0($s0) #current character
                        beq     $s6, $0, endLoop
                        addi    $s1, $s1, 1
                        sw      $s1, 4($sp)
                        addi    $s0, $s0, 1
                        j       findOctalSize
                endLoop:
                        addi    $s1, $s1, -1
```

```
        sw      $s1, 4($sp)
        lw      $s0, 0($sp)
        lw      $s2, 8($sp)
        addi    $s2, $s2, 1
goToEnd:
        beq     $s2, $s1, atEnd
        addi    $s2, $s2, 1
        addi    $s0, $s0, 1
        j       goToEnd
atEnd:
        add     $s2, $0, $0 #iteration counter
        add     $s3, $0, $0 #head power counter
        add     $s4, $0, $0 #sum
         loopAddition:
                beq     $s2, $s1, partADone
                add     $s5, $0, $0
                addi    $s6, $0, 1
                loopPower:
                        beq     $s5, $s3, endloopPower
                        mul     $s6, $s6, 8
                        addi    $s5, $s5, 1
                        j       loopPower
                endloopPower:
                lb      $s5, 0($s0)
                addi    $s5, $s5, -48
                mul     $s5, $s5, $s6
                add     $s4, $s4, $s5
                sw      $s4, 16($sp)
                addi    $s2, $s2, 1
                addi    $s3, $s3, 1
                addi    $s0, $s0, -1
                j       loopAddition
partADone:
        move    $v0, $s4
        jr      $ra
```

REVERSE NUMBER

```
reverseNumber:
        #load stack
        lw      $s0, 0($sp)
        lw      $s1, 4($sp)
        lw      $s2, 8($sp)
        lw      $s3, 12($sp)
        lw      $s4, 16($sp)
        #print in hex
        la      $a0, reverseNumberOutputPrompt1
        li      $v0, 4
        syscall
        #output hex
        move    $a0, $s0
        li      $v0, 34
        syscall
        la      $a0, newline
        li      $v0, 4
        syscall
        add     $s5, $0, $0
        loopToHex:
                sra     $s3, $s0, 4
```

```
                sll     $s1, $s3, 4
                xor     $s2, $s0, $s1
                move    $s0, $s3
                #if value is greater than 9
                bgt     $s2, 9, convertToAlphabet
                addi    $s2, $s2, 48
                #storing value
                sb      $s2, 0($s4)
                ble     $s0, $0, endConversion
                #increment
                addi    $s4, $s4, 1
                j       loopToHex
        convertToAlphabet:
                addi    $s2, $s2, 55
                sb      $s2, 0($s4)
                ble     $s0, $0, endConversion
                addi    $s4, $s4, 1
                j       loopToHex
        endConversion:
                addi    $s4, $s4, 1
                sb      $0, 0($s4)
                move    $v0, $s4
                jr      $ra
```

WHOLE PROGRAM

```
.text
        main:
                jal     interactWithUser
                #returns from sub
                li      $v0, 10
                syscall
        interactWithUser:
                #print prompt
                la      $a0, interactionPrompt
                li      $v0,4
                syscall
                #input integer
                li      $v0, 5
                syscall
                #menu selection
                beq     $v0, 3, returnToMain
                beq     $v0, 1, option1
                beq     $v0, 2, option2
                j       interactWithUser
        option1:
                #initialize registers
                add     $s0, $0, $0
                add     $s1, $0, $0
                add     $s2, $0, $0
                add     $s3, $0, $ra
                add     $s4, $0, $0
                #print prompt
                la      $a0, convertToDecPrompt
                li      $v0,4
                syscall
                #input string
                la      $a0, inputBuffer
                li      $a1, 30
                move    $s0,$a0
```

```
        li      $v0,8
        syscall
        #initialize stack
        addi    $sp, $sp, -20
        sw      $s0, 0($sp)
        sw      $s1, 4($sp)
        sw      $s2, 8($sp)
        sw      $s3, 12($sp)
        sw      $s4, 16($sp)
        #got to sub
        jal     convertToDec
        #reload stack
        lw      $s0, 0($sp)
        lw      $s1, 4($sp)
        lw      $s2, 8($sp)
        lw      $s3, 12($sp)
        lw      $s4, 16($sp)
        #print prompt
        move    $ra, $s3
        la      $a0, convertToDecOutputPrompt
        li      $v0,4
        syscall
        #print string outh
        move    $a0, $s4
        li      $v0,1
        syscall
        #print newline
        la      $a0, newline
        li      $v0,4
        syscall
        j       interactWithUser
option2:
        #initialize registers
        add     $s0, $0, $0
        add     $s1, $0, $0
        add     $s2, $0, $0
        add     $s3, $0, $ra
        #input string
        la      $s4, storageBuffer
        la      $a0, reverseNumberPrompt
        li      $v0,4
        syscall
        #input integer
        li      $v0, 5
        syscall
        move    $s0, $v0
        #initlizing stack
        sw      $s0, 0($sp)
        sw      $s1, 4($sp)
        sw      $s2, 8($sp)
        sw      $s3, 12($sp)
        sw      $s4, 16($sp)
        #go to sub
        jal     reverseNumber
        #reload stack
        lw      $s0, 0($sp)
        lw      $s1, 4($sp)
        lw      $s2, 8($sp)
        lw      $s3, 12($sp)
        lw      $s4, 16($sp)
        #print prompt
        move    $ra, $s3
        la      $a0, reverseNumberOutputPrompt2
        li      $v0,4
        syscall
        #print string out
```

```
        move    $a0, $s4
        li      $v0,4
        syscall
        #print newline
        la      $a0, newline
        li      $v0,4
        syscall
        j       interactWithUser
convertToDec:
        #reload stack // initialize values
        lw      $s0, 0($sp)
        lw      $s1, 4($sp)
        lw      $s2, 8($sp)
        add     $s3, $0, $0
        lw      $s4, 16($sp)
        add     $s6, $0, $0
        findOctalSize:
                lb      $s6, 0($s0) #current character
                beq     $s6, $0, endLoop
                addi    $s1, $s1, 1
                sw      $s1, 4($sp)
                addi    $s0, $s0, 1
                j       findOctalSize
        endLoop:
                addi    $s1, $s1, -1
                sw      $s1, 4($sp)
                lw      $s0, 0($sp)
                lw      $s2, 8($sp)
                addi    $s2, $s2, 1
        goToEnd:
                beq     $s2, $s1, atEnd
                addi    $s2, $s2, 1
                addi    $s0, $s0, 1
                j       goToEnd
        atEnd:
                add     $s2, $0, $0 #iteration counter
                add     $s3, $0, $0 #head power counter
                add     $s4, $0, $0 #sum
                 loopAddition:
                        beq     $s2, $s1, partADone
                        add     $s5, $0, $0
                        addi    $s6, $0, 1
                        loopPower:
                                beq     $s5, $s3, endloopPower
                                mul     $s6, $s6, 8
                                addi    $s5, $s5, 1
                                j       loopPower
                        endloopPower:
                        lb      $s5, 0($s0)
                        addi    $s5, $s5, -48
                        mul     $s5, $s5, $s6
                        add     $s4, $s4, $s5
                        sw      $s4, 16($sp)
                        addi    $s2, $s2, 1
                        addi    $s3, $s3, 1
                        addi    $s0, $s0, -1
                        j       loopAddition
        partADone:
                move    $v0, $s4
                jr      $ra
reverseNumber:
        #load stack
        lw      $s0, 0($sp)
        lw      $s1, 4($sp)
        lw      $s2, 8($sp)
        lw      $s3, 12($sp)
```

```
                lw      $s4, 16($sp)
                #print in hex
                la      $a0, reverseNumberOutputPrompt1
                li      $v0, 4
                syscall
                #output hex
                move    $a0, $s0
                li      $v0, 34
                syscall
                la      $a0, newline
                li      $v0, 4
                syscall
                add     $s5, $0, $0
                loopToHex:
                        sra     $s3, $s0, 4
                        sll     $s1, $s3, 4
                        xor     $s2, $s0, $s1
                        move    $s0, $s3
                        #if value is greater than 9
                        bgt     $s2, 9, convertToAlphabet
                        addi    $s2, $s2, 48
                        #storing value
                        sb      $s2, 0($s4)
                        ble     $s0, $0, endConversion
                        #increment
                        addi    $s4, $s4, 1
                        j       loopToHex
                convertToAlphabet:
                        addi    $s2, $s2, 55
                        sb      $s2, 0($s4)
                        ble     $s0, $0, endConversion
                        addi    $s4, $s4, 1
                        j       loopToHex
                endConversion:
                        addi    $s4, $s4, 1
                        sb      $0, 0($s4)
                        move    $v0, $s4
                        jr      $ra
        returnToMain:
                jr      $ra
.data
inputBuffer: .space 30
storageBuffer: .space  30
interactionPrompt:     .asciiz "Please select one of the options:\n1. Octal to decimal\n2.
reverse Hex\n3. Return to main\nYour selection: "
convertToDecPrompt:    .asciiz "Enter octal: "
convertToDecOutputPrompt:      .asciiz "The decimal value is: "
newline:       .asciiz "\n"
test:   .asciiz "test\n"
reverseNumberPrompt:   .asciiz "Please enter the decimal number: "
reverseNumberOutputPrompt2:    .asciiz "Reversed hex: "
reverseNumberOutputPrompt1:    .asciiz "normal hex: "
```

PART D: GENERATING MACHINE INSTRUCTION

A. beq $t0, $t1, next

if equal, go from: 0x10010054 to 0x10010064, so immediate = (0x10010064 – 0x10010054) / 4 = 0100

000100	01000	01001	0000 0000 0000 0100

Hex instructions: 0x11090004

B. bne $t2, $t3, again

if not equal, go from: 0x10010058 to 0x10010040, so immediate = (0x10010040 – 0x10010058) / 4 = -0110

000101	01010	01011	1111 1111 1111 1010

Hex instructions: 0x154BFFFA

C. jr $ra

000000	11111	00000	00000	00000	001000

Hex instructions: 0x03E00008

D. j again

if equal, go from: 0x10010070 to 0x10010040, so immediate = (0x10010040 – 0x10010068) / 4 = -1010

000010	0000 0000 0001 0000 0000 0100 00

Hex instructions: 0x08004010