CS 224

Section No: 02
Fall 2019

Lab No: 1

İrem Seven
21704269

Part 1: Input an Array

```
.data
array: .word 20
invalidtext: .asciiz "size should be less than 20"
prompt:.asciiz "Enter size: "
getNum:          .asciiz "enter number:  "
space: .asciiz " "
newLine: .asciiz "\n"
.text
        .globl __start

__start: #this program creates array of integers, make it reverse
        la $a0,prompt
        li $v0,4
        syscall

        li $v0, 5           #get input size
        syscall
        add $t1,$v0,$zero  #t1 is size of array
        addi $t3,$zero,20  #array will be max 20
        bgt $t1,$t3,invalid
        mul $t0,$t1,4  #multiply size by 4 since mips is byte addressable
        sub $t0,$zero,$t0 #sub since will be allocate space
        add $sp,$sp,$t0  #allocate space stack with specified size
        addi $t2,$t2,0  #index
        add $t4,$sp,$zero #array address in t4
        loop:
                beq $t2,$t1,done #t2 index t1 size
          la $a0,getNum
             li $v0,4
             syscall
             li $v0,5
             syscall
             add $t3,$v0,$zero
                 sw  $t3,0($t4)
             la $a0,newLine
             li $v0,4
             syscall
             addi $t4,$t4,4
             addi $t2,$t2,1
             j loop

             done:

            add $a1,$sp,$zero #array address in a1
            addi $a2,$zero,0 #i
            jal print
           addi $t4,$t4,-4
        addi $t7,$zero,0 #index
         reverse:   #put array elements reversely
                beq $t7,$t1,out #t2 i t1 size
                mul  $s2,$t7,4
                lw  $t8,0($t4)
             sw $t8,array($s2)
             addi $t4,$t4,-4  #decrement address
```

```
        addi $t7,$t7,1   #increment index
        j reverse

    out:

    addi $t9,$zero,0 #index
    addi $s3,$zero,0
    addi $t4,$sp,0
   goback:  #puts elements back to the original array
            beq $t9,$t1,out1 #t2 i t1 size
            mul  $s3,$t9,4
        lw $t8,array($s3)
        sw $t8,0($t4)
        addi $t4,$t4,4
        addi $s3,$s3,4
        addi $t9,$t9,1
        j goback

    out1:
      la $a0,newLine
      li $v0,4
      syscall
      add $a1,$sp,$zero #array address in a1
      addi $a2,$zero,0  #a2 is index for print function
      jal print
    j out3
invalid:
la $a0, invalidtext
li $v0,4
syscall

out3:

        li $v0,10  #system out
        syscall

  print:
        beq $a2,$t1,exit #t1 is size of array size
            lw $t5,0($a1)
            add $a0,$t5,$zero
         li $v0,1
        syscall
        la $a0,space
        li $v0,4
        syscall
        addi $a1,$a1,4 #incement address by 4 bytes
        addi $a2,$a2,1
        j print

        exit: #exit when done with printing

        jr $ra
```

Part 2: Palindrome

```
.data
invalidless: .asciiz "size should be more than 0"
invalidmore: .asciiz "size should be max 20"
prompt:.asciiz "Enter size: "
getNum:          .asciiz "enter number:  "
space: .asciiz " "
newLine: .asciiz "\n"
isPalindrome: .asciiz "It is a Palindrome"
notPalindrome: .asciiz "It is not a Palindrome"
.text
        .globl __start

__start:
        la $a0,prompt
        li $v0,4
        syscall

        li $v0, 5
        syscall
        add $s0,$v0,$zero #s0 : size of array

        addi $t0,$zero,20 #invalid if more than 20 elements
        bgt $s0,$t0,invalid1
        addi $t0,$zero,1 #invalid if less than 1 element
        blt $s0,$t0,invalid2

        mul $s1,$s0,4     # 4 bytes
        sub $s1,$zero,$s1 # allocating space from stack as size number
        add $sp,$sp,$s1   # allocating space
        addi $t0,$zero,0  #t0:index
        add $t1,$sp,$zero #array address in t1
        loop:
            beq $t0,$s0,done #t2 i t1 size
            la $a0,getNum  #print to get number
            li $v0,4
            syscall
            li $v0,5                #get element from user
            syscall
                sw  $v0,0($t1) #store element into array
            la $a0,newLine
            li $v0,4
            syscall
            addi $t1,$t1,4  #increment by 4 adress of sp
            addi $t0,$t0,1  #increment index
            j loop
            done:

        #t1 currently holds the address of the last element in array
        add $a1,$sp,$zero    #array address in a1 now its zero to print
        addi $a2,$zero,0     #a2:index

        jal print
        la $a0,newLine
        li $v0,4
        syscall
```

```
    addi $t3,$zero,1
    beq $s0,$t3,palindrome
    addi $t1,$t1,-4
    addi $t0,$zero,0  #to:index
    addi $t0,$sp,0     #t0 holds index from beginning
    addi $t2,$t1,0     #t2 holds index from last
    addi $t3,$zero,2
    div $s0,$t3 #s0 is size of array
    mfhi $t4
    beq $t4,$zero,even

    #t5 is counter for terminate palindrome loop
    divu $t5,$s0,$t3
    addi $t5,$t5,-1
    j else
    even:
    divu $t5,$s0,$t3
    else:

    #stop loop when t5 = 0
   palindromeloop:
            lw $s2,0($t0)    #s2 holds number from begining
            lw $s3,0($t2)    #s3 holds number from last
            bne $s2,$s3,notpalindrome
            addi $t0,$t0,4
            addi $t2,$t2,-4
            beq $t5,$zero,palindrome
            addi $t5,$t5,-1 #decrement the counter
            j palindromeloop

    palindrome:
            la $a0,isPalindrome
            li $v0,4
            syscall
            j else2
            notpalindrome:
            la $a0,notPalindrome
            li $v0,4
            syscall
    else2:

j eror1   #if user enters more than 20 elements
invalid1:
  la $a0, invalidmore
  li $v0,4
  syscall

eror1:
j eror2  #if user enters less than 1 elements
invalid2:
  la $a0, invalidless
  li $v0,4
  syscall

eror2:
    li $v0,10         #terminate program
    syscall
```

```
print:
	beq $a2,$s0,exit #t2 i t1 size
	    lw $t9,0($a1)
	    add $a0,$t9,$zero
	 li $v0,1
	syscall
	la $a0,space
	li $v0,4
	syscall
	addi $a1,$a1,4
	addi $a2,$a2,1
	j print

	exit:

	jr $ra
```

Part 3: Division Without Division Instruction

```
.data
text1: .asciiz "Enter divided: "
text2: .asciiz "Enter divisor: "
text3: .asciiz "quotient: "
text4: .asciiz "remainder: "
newLine: .asciiz "\n"
.text
        .globl __start
__start:
  la $a0,text1
  li $v0,4
  syscall

  li $v0,5
  syscall
  add $s0,$v0,$zero

  la $a0,text2
  li $v0,4
  syscall

  li $v0,5
  syscall
  add $s1,$v0,$zero
  addi $t0,$zero,0 #t0 is counter = quotient
  #substract divisor from dividend till dividend becomes smaller

  divide:
        blt $s0,$s1,done
        sub $s0,$s0,$s1
        addi $t0,$t0,1
        j divide
        done:
  la $a0,text3   #quotient
  li $v0,4
  syscall
  addi $a0,$t0,0
  li $v0,1
  syscall
  la $a0,newLine
  li $v0,4
  syscall
  la $a0,text4    #remainder
  li $v0,4
  syscall
  addi $a0,$s0,0
  li $v0,1
  syscall

  li $v0,10  #terminate program
  syscall
```

Part 4: Object Code Generation

Mips Assembly Code:

```
add     $t0, $t1, $t2
addi    $s0, $s3, 15
mult    $a0, $a1
sw      $t1, 8($t2)
lw      $t2, 8($t1)
```

Object Code in Binary:

0000 0001 0010 1010 0100 0000 0010 0000
0010 0010 0111 0000 0000 0000 0000 1111
0000 0000 1000 0101 0000 0000 0001 1000
1010 1101 0100 1001 0000 0000 0000 1000
1000 1101 0010 1010 0000 0000 0000 1000

Object Code in Hexadecimal:

0x012a4020
0x2270000f
0x00850018
0xad490008
0x8d2a0008

Part 5: Define Terms

a*.Symbolic Machine Instruction:* Instructions which are converted into machine code by assembler.

Ex:

```
addi $t0,$t1,100
beq $s0,$s1,label
```

b*.Machine Instructions:* Instructions which are consisted of hexadecimal or binary values. It can be directly interpreted by the computer.

Ex:

```
0x21100005    equivalent to:  addi $s0, $t0, 5
0x02288022                    sub  $s0, $s1, $t0
```

c*.Assembler Directive:* Assembler directives are the instructions which directs the assembler to execute something within the computer.

Ex:

```
.data
.text
```

d*.Pseudo Instruction:* Pseudo instructions cannot be converted to machine code directly by assembler. Assembler first converts them into symbolic code after that, it converts into machine code.

Ex:

```
bge $t1,-50,label      in symbolic code:      slti $1, $9, -50
label:                                        beq $1, $0, 0

divu $s0,$s1,$t0                              bne $8, $0, 1
                                               break
                                              divu $17, $8
                                              mflo $16
```