CS224

SECTION 03

LAB02 PRELIMINARY REPORT

AYBÜKE CEREN DURAN/21302686

**1**. Write MIPS assembly language programs as described below.

a. convertToDec:

**(EXECUTED ON MARS)**

.data

myOctalNumber: .asciiz "20" # octal number 20 is declared

tempArray: .space 80

.text

la $a0, myOctalNumber

la $a2, tempArray

addi $a2, $a2, 1 #to overcome word align problem

move $t9, $a2

move $t0, $a0 # for further use, the address is copied into $t0

move $t3, $a0 #it will be used for iteration later

li $t6,1

#in the base-8, the digits from 0 to 7 are used, so this condition will be checked.

#in the ASCII table, the digit 7 is encoded as 0x37

#in the ASCII table, the digit 0 is encoded as 0x30

li $t5, 0x37

li $s1, 2 # the register $s1 is used for holding the size of the octal number

li $s2, 0 #decimal value in initialized as 0.

li $s3, 0

li $t4, 0#this will be used for holding the size of temporary array later.

jal convertToDec

li $v0,10

syscall

convertToDec:

addi $sp, $sp, -16

sw $s3, 0($sp)#isValidOctal will use $s3 later :)

sw $s2, 4($sp)

sw $s1, 8($sp)

sw $ra, 12($sp)

#now, check the condition whether if the given number is octal or not.

jal isValidOctal

lw $s3, 0($sp)

lw $s2, 4($sp)

lw $s1, 8($sp)

lw $ra, 12($sp)

addi $sp, $sp, 16

jr $ra

isValidOctal:

lb $s3, 0($t3) # in the function convertToDec, value of $s3 IS IN THE STACK

beq $s3, 0x00, doCalculations #if the null character is encountered, go to lastStep

#as mentioned above, $t5 holds 0x37 which is the digit 7 in ASCII table

#we will check whether if the byte of the input number is greater than 7

sgt $t7, $s3, $t5 # $t7 holds the result of the condition above

bne $t7, 1, yesValid

yesValid:

addi $t3, $t3, 1

subi $t7, $s3, 48 **# when we subtract 48, we get the actual number**

sw $t7, 0($t9) **#I stored decimal ASCIIS in temporary array**

addi $t4, $t4, 1 **#update the size of the temporary array as you put elements.**

add $t9, $t9, 4

j isValidOctal

doCalculations:

subi $t4, $t4, 1

beqz $t4, sum

lw $t7, 0($a2) **#get digits from the temporary array**

move $t8, $t4 # copied for further use

bge $t8, 1, intCalc

blt $t8, 1, continue

intCalc:

mul $t6, $t6, 8 **#$t6 will hold 8^N**

subi $t8, $t8, 1

beq $t8, 0, continue

j intCalc

continue:

mul $t5, $t6, $t7

li $t6, 1

add $s2, $s2, $t5

add $a2, $a2, 4

j doCalculations

sum:

lw $t7, 0($a2)

add $s2, $s2, $t7

j printNow

printNow:

move $a0, $s2

li $v0,1

syscall

jr $ra

**b**. **interactWithUser**:

**(EXECUTED ON MARS)**

.data

decimalArray:.space 100

tempArray: .space 80

askOctalNumber: .asciiz "Please enter an octal number:\n"

errorMessage: .asciiz "\nThe number you entered is not a valid octal number! \n "

decimal: .asciiz "\nThis is the decimal version of the octal number : "

userOctalNumber: .space 100

askSize: "Please enter the size of the number: \n"

.text

entering:

li $v0,4

la $a0, askOctalNumber

syscall

li $v0,8#read a string in MIPS

la $a0, userOctalNumber

li $a1, 20

syscall

la $a0, userOctalNumber

#add $a0, $a0, 0#to overcome word align problem

move $t8, $a0 #for further use, the address is copied

move $t0, $a0 # for further use, the address is copied into $t0

move $t3, $a0 #it will be used for iteration later

#the next step is to calculate the size of the user's input

li $v0, 4

la $a0, askSize

syscall

li $v0, 5

syscall

move $s1, $v0

**# $s1 register is used to hold the size of the user octal number**

**#the next step is to decide whether if the user's input octal or not.**

#in the base-8, the digits from 0 to 7 are used, so this condition will be checked.

#in the ASCII table, the digit 7 is encoded as 0x37

#li $t8, 0

li $t5, 0x37

la $a2, tempArray

move $t9, $a2

li $t6,1

li $s2, 0 #decimal value in initialized as 0.

li $s3, 0

li $t4, 0#this will be used for holding the size of temporary array later.

jal convertToDec

li $v0,10

syscall

convertToDec:

addi $sp, $sp, -16

sw $s3, 0($sp)#isValidOctal will use $s3 later :)

sw $s2, 4($sp)

sw $s1, 8($sp)

sw $ra, 12($sp)

**#now, check the condition whether if the given number is octal or not.**

jal isValidOctal

lw $s3, 0($sp)

lw $s2, 4($sp)

lw $s1, 8($sp)

lw $ra, 12($sp)

addi $sp, $sp, 16

jr $ra

isValidOctal:

lb $s3, 0($t3) # in the function convertToDec, value of $s3 IS IN THE STACK

beq $s3, 0x00, doCalculations **#if the null character is encountered**

beq $s3, 0x0A, skip #this instruction skips the data link character

#as mentioned above, $t5 holds 0x37 which is the digit 7 in ASCII table

#we will check whether if the byte of the input number is greater that 7

sgt $t7, $s3, $t5 # $t7 holds the result of the condition above

bne $t7, 1, yesValid

beq $t7, 1, notValid

skip:

addi $t3, $t3, 1

j isValidOctal

notValid:

li $v0, 4

la $a0, errorMessage

syscall

j entering

yesValid:

addi $t3, $t3, 1

subi $t7, $s3, 48 **# when we subtract 48, we get the actual number**

sw $t7, 0($t9) **#I stored decimal ASCIIS in temporary array**

addi $t4, $t4, 1 #update the size of the temporary array as you put elements.

add $t9, $t9, 4

j isValidOctal

doCalculations:

subi $t4, $t4, 1

beqz $t4, sum

lw $t7, 0($a2) #get digits from the temporary array

move $t8, $t4 # copied for further use

bge $t8, 1, intCalc

blt $t8, 1, continue

intCalc:

mul $t6, $t6, 8#$t6 will hold 8^N

subi $t8, $t8, 1

beq $t8, 0, continue

j intCalc

continue:

mul $t5, $t6, $t7

li $t6, 1

add $s2, $s2, $t5

add $a2, $a2, 4

j doCalculations

sum:

lw $t7, 0($a2)

add $s2, $s2, $t7

j printNow

printNow:

move $a0, $s2

li $v0,1

syscall

jr $ra

**2**. Generating machine instructions

Firstly, I will calculate **j again** instruction. The JTA of the **j again** instruction is 00 40 00 A016 . The top 4 bits and bottom 2 bits are discarded (in bold).

The JTA of the **j again**: **0000** 0000 0100 0000 0000 0000 1010 00**00**.

The bold ones in the JTA part will be discarded.

26-bit address of the JTA will become 0000 0100 0000 0000 0000 1010 00. 6-bit opcode is 2.

opcode JTA

j again

2 0x0100028

opcode JTA

j again

000010 00 0001 0000 0000 0000 0010 1000

**Answer:** 0x08100028hex

Second, I will calculate the “beq $t0, $t6, next” this is an I-type instruction meaning that we need an immediate value to calculate the machine instruction. For that, we will count the number of instructions after the beq instruction have been executed. In this case there are 4 lines (excluding the instruction itself) of code so our immediate value is basically +4.

opcode: 4 in hex

rs: 8

rt: 14

immediate: 3

op rs rt imm

beq $t0, $t6, next

000100 00100 01110 0000000000000011

**Answer:** 0x108E0003hex

Third, I will calculate bne $t0, $t6, again

opcode: 5

rs: 8

rt: 14

immediate: -6

op rs rt imm

bne $t0, $t6, again

000101 00100 01110 1111111111111010

**Answer:** 0x148EFFFA