CS 224

Computer Organization

Preliminary Design Report

Lab 02

Cholpon Mambetova

21402612

Section 4

March 6, 2017

12:59 pm

1A. Subprogram converting octal number to decimal

```
# ----------- HEADING ---------------

# CS 224 - 4
# Lab 2, Part 1a: A subprogram that converts
#                 the input octal number, string,
#                 to a decimal number and prints it.
#                 Argument: beginning of a string
#                 Return: decimal number
# Cholpon Mambetova
# 21402612

# ----------- VARIABLES -------------

# $s0 = decimal number
# $s1 = octal number as string
# $s2 = 0's value in ASCII, 48, lower bound
# $s3 = 7's value in ASCII, 55, upper bound
# $s4 = copy of $s1
# $t0 = octal number as number
# $t1 = k, factor by with each digit will be multiplied
# $t2 = i
# $t3 = temp chars in string
# $t4 = temp int from chars
# $t5 = comparison results

convertToDec:
      move $s1, $a0    # save string to $s1 (base address)

      # Initialize
      addi $t0, $0, 0    # sum of the digits in decimal
      addi $t1, $0, 1    # initilize k
      addi $t2, $0, 0    # initilize i
      lbu $t3, 0($s1)    # get first char
      addi $s2, $0, 48   # 0's ASCII value
      addi $s3, $0, 55   # 7's ASCII value
      move $s4, $s1      # copy of the address of the first char

icounter:  # calculate the length of the string
      beq $t3, $0, initk
      addi $t2, $t2, 1  # i++

      addi $s4, $s4, 1
      lbu $t3, 0($s4)

      j icounter

initk: # count the first factor to multiply the most sign dig by
      beq $t2, 1, endinitk
      sll $t1, $t1, 3
      addi $t2, $t2, -1
      j initk

endinitk:
      lbu $t3, 0($s1)    # get first char again

loop: # calculate decimal number
      beq $t3, $0, endloop  # leave if string char is empty

      # check if char is within 48 and 55
      # which is char is a number
      slt $t5, $t3, $s2
      beq $t5, 1, error
      slt $t5, $s3, $t3
      beq $t5, 1, error

      # convert char to int
      addi $t4, $t3, -48
      mul $t4, $t4, $t1

      # add decicam number to result
      add $t0, $t0, $t4

      # prep for next loop
      addi $s1, $s1, 1
      lbu $t3, 0($s1)
      srl $t1, $t1, 3
      j loop

error:  # if not an octal number
      addi $s0, $0, -1
endloop:
	move $v0, $t0
	jr $ra

#-------- END OF SUBPROGRAM ------
```

1B. Program interacting with user and converting octal number to decimal

```
# ----------- HEADING ---------------

# CS 224 - 4
# Lab 2, Part 1b: A program that converts
#               the input octal number, string,
#               to a decimal number and prints it
# Cholpon Mambetova
# 21402612

# ----------- VARIABLES -------------

# $s0 = decimal number
# $s1 = octal number as string
# $s2 = 0's value in ASCII, 48
# $s3 = 9's value in ASCII, 57
# $s4 = copy of $s1
# $t0 = octal number as number
# $t1 = k, factor by with each digit will be multiplied
# $t2 = i
# $t3 = temp chars in string
# $t4 = temp int from chars
# $t5 = comparison results

# ----------- PROGRAM START --------------
        .text
        .globl __start

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

        # Get the input
        li $v0, 8       # input string
        la $a0, octalNo
        li $a1, 20
        syscall

        move $s1, $a0    # save string to $s1 (base address)

        jal checkifoctal
        beq $v0, -1, __start

        move $a0, $s1
        jal convertToDec
        move $s0, $v0

        # Print output
        la $a0, outputMessage
        li $v0, 4
        syscall

        la $a0, ($s0)
        li $v0, 1
        syscall

        # End of program
        li $v0, 10
        syscall

# -------------- CHECKER -----------------
checkifoctal:   # check if char is within 48 and 55
                # which is char is a number
      #move $s1, $a0    # save string to $s1 (base address)

        # Initialize
        lbu $t3, 0($s1)    # get first char
        addi $s2, $0, 48   # 0's ASCII value
        addi $s3, $0, 55   # 7's ASCII value
        move $s4, $s1      # copy of the address of the first char
        addi $v0, $0, 1    # initialize checker's result to be true

checkloop:
        beq $t3, 10, endcheck

        slt $t5, $t3, $s2
        beq $t5, 1, error
        slt $t5, $s3, $t3
        beq $t5, 1, error

        addi $s4, $s4, 1
        lbu $t3, 0($s4)

        j checkloop

error:  # Print error
        la $a0, errorMessage
        li $v0, 4
        syscall

        addi $v0, $0, -1  # checker's result is false

endcheck:
        jr $ra

# ----------- END OF CHECKER -------------

# ----------- CONVERTER ---------------------
convertToDec:
        move $s1, $a0    # save string to $s1 (base address)

        # Initialize
        addi $t0, $0, 0    # sum of the digits in decimal
        addi $t1, $0, 1    # initilize k
        addi $t2, $0, 0    # initilize i
        lbu $t3, 0($s1)    # get first char
        move $s4, $s1      # copy of the address of the first char

icounter:  # calculate the length of the string
        beq $t3, 10, initk

        addi $t2, $t2, 1  # i++

        addi $s4, $s4, 1
        lbu $t3, 0($s4)

        j icounter

initk: # count the first factor to multiply the most sign dig by
```

```
	beq $t2, 1, endinitk
	   sll $t1, $t1, 3
	   addi $t2, $t2, -1
	   j initk

endinitk:
	   lbu $t3, 0($s1)    # get first char again

converterloop: # calculate decimal number
	   beq $t3, 10, endconverterloop  # leave if string char is empty

	   # convert char to int
	   addi $t4, $t3, -48
	   #convert octal digit to its decimal value
	   mul $t4, $t4, $t1
	   # add decimal value to result
	   add $t0, $t0, $t4

	   # prep for next loop
	   addi $s1, $s1, 1
	   lbu $t3, 0($s1)
	   srl $t1, $t1, 3
	   j converterloop

endconverterloop:
	addi $v0, $t0, 0
	jr $ra

# ----------- END OF CONVERTER ------

#------------ DATA ------------------

		.data
octalNo:	.space 20
prompt:			.asciiz "Please, enter the octal number: "
errorMessage:.asciiz "The entered number is not an octal number!\n"
outputMessage:		.asciiz "The decimal value is: "

#------------ END OF PROGRAM -------------
```

2. Generating Machine Instructions

Machine instruction for “`j again`” is `0x08004008`
Machine instruction for “`beq $t0, $t1, next`” is `0x12280010`
Machine instruction for “`bne $t0, $t1, again`” is `0x1628FFF8`

Because:

```
-----------------------------------------------------------------------------
--
(10 01 00 20) again: add ...
(10 01 00 24)       add ...
(10 01 00 28)        beq $t0, $t1, next   #(beq rt, rs, label)  offset is 4
(10 01 00 2C)        bne $t0, $t1, again  #(bne rt, rs, label)  offset is
-2
(10 01 00 30)        add ...
(10 01 00 34)        add ...
(10 01 00 38)        add ...
(10 01 00 3C) next:  j again
-----------------------------------------------------------------------------
--
t0 = 8 (dec) = 01000 (bin)
t1 = 9 (dec) = 10001 (bin)
```

BEQ and BNE are I-Type

op (6 bits) rs (5 bits) rt (5 bits) imm (16 bits)

```
beq has offset of 4x4 which makes 0000 0000 0001 0000 for 16-bits part
beq $t0, $t1, next:  000100 10001 01000 0000 0000 0001 0000 = 0x12280010

bne has offset of -2x4 which makes 1111 1111 1111 1000 for 16-bit part in
2’s complement
bne $t0, $t1, again: 000101 10001 01000 1111 1111 1111 1000 = 0x1628FFF8
```

J is J-Type

```
op (6 bits) addr (26 bits)
```

j again: 000010 ~~0001~~ 0000 0000 0001 0000 0000 0010 00~~00~~ = 0x08004008