CS224
Section No.: 2
Spring 2018
Lab No.: 1
EFE ACER / 21602217

1. Code for the first question:

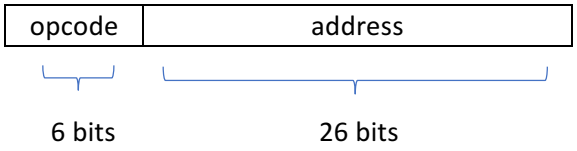
```
#PreLab 2 - Question 1 - EFE ACER
#This program converts a hexadecimal number that the user enters to a
decimal number and displays it.

	.text
	.globl __start
__start:
	la $a0, intro #printing an intro message
	li $v0, 4
	syscall
	jal interactWithUser #calling the method to interact with the user
	li $v0, 10 #stopping the execution
	syscall

#Subprogram to convert a hexadecimal number to a decimal number.
convertHexToDec:
	subi $sp, $sp, 24 #push the values to the stack in order to save them
	sw $s1, 20($sp)
	sw $s2, 16($sp)
	sw $s3, 12($sp)
	sw $s4, 8($sp)
	sw $s5, 4($sp)
	sw $ra, 0($sp)
	li $v0, 0 #return value is initialized to 0
		move $s5, $a0 #$s5 points to the base address now
		findLastBytesAddress: #go to the last item's address
			lb $s1, 0($s5) #loading the character to $s1
			addi $s5, $s5, 1 #incrementing the byte pointer
			beq $s1, 10, done #finish if there is a newline(ascii 10)
			bnez  $s1, findLastBytesAddress #iterate until NUL comes
		done:
			subi $s5, $s5, 2 #$s5 points to the last item's address
		li $s4, 1 #the multiplier
	next:
		lb $s1, 0($s5) #loading the character to $s1
		#Checking if $s1 is between 0 and 9 or not
		slti $s2, $s1, 58 #$s2 = 1 if ($s1 <= 57 (ascii of 9))
		slti $s3, $s1, 48
		not $s3, $s3 #$s3 = 1 if ($s1 >= 48 (ascii of 0))
		and $s2, $s2, $s3 #$s2 = $s2 and $s3
		beq $s2, 1, inDecimalRange #branch if $s2 is between 0 and 9
		#Checking if $s1 is between A and F or not
		slti $s2, $s1, 71 #$s2 = 1 if ($s1 <= 70 (ascii of F))
		slti $s3, $s1, 65
		not $s3, $s3 #$s3 = 1 if ($s1 >= 65 (ascii of A))
		and $s2, $s2, $s3 #$s2 = $s2 and $s3
		beq $s2, 1, outOfDecimalRangeCapital #branch if $s2 is between
A and F
		#Checking if $s1 is between a and f or not
```

```
            slti $s2, $s1, 103 #$s2 = 1 if ($s1 <= 102 (ascii of F))
            slti $s3, $s1, 97
            not $s3, $s3 #$s3 = 1 if ($s1 >= 97 (ascii of A))
            and $s2, $s2, $s3 #$s2 = $s2 and $s3
            beq $s2, 1, outOfDecimalRangeLowercase #branch if $s2 is
between a and f
            j notAValidDigit #then $s2 is not a valid hexadecimal digit
            inDecimalRange:
                  subi $s1, $s1, 48 #ASCII to the corresponding decimal
                  mul $s1, $s1, $s4 #exact value of the digit in $s1
                  j continue
            outOfDecimalRangeCapital:
                  subi $s1, $s1, 55 #ASCII to the corresponding decimal
                  mul $s1, $s1, $s4 #exact value of the digit in $s1
                  j continue
            outOfDecimalRangeLowercase:
                  subi $s1, $s1, 87 #ASCII to the corresponding decimal
                  mul $s1, $s1, $s4 #exact value of the digit in $s1
                  j continue
            notAValidDigit:
                  li $v0, -1 #-1 indicates invalid character
                  j end
            continue:
                  add $v0, $v0, $s1 #accumulating the sum
                  mul $s4, $s4, 16 #scaling the digit multiplier
                  subi $s5, $s5, 1 #$s5 points to the previous character
                  bge $s5, $a0, next #until $s5 is less than base address
      end:
            lw $ra, 0($sp) #pop the values to reload the originals
            lw $s5, 4($sp)
            lw $s4, 8($sp)
            lw $s3, 12($sp)
            lw $s2, 16($sp)
            lw $s1, 20($sp)
            addi $sp, $sp, 24
            jr $ra #return

#Subprogram that reads a hexadecimal from the user, it prints its decimal
value.
interactWithUser:
      subi $sp, $sp, 8 #push the values to the stack in order to save them
      sw $s1, 4($sp)
      sw $ra, 0($sp)
      again:
            la $a0, prompt #prompt to read the hexadecimal number
            li $v0, 4
            syscall
            la $a0, hexNo #address of input buffer
            li $a1, 21 #20 bytes is the max number of characters to read +
(newLine + null) or null
            li $v0, 8 #reading the hexademical number
            syscall
            jal convertHexToDec #calling the converter function
            move $s1, $v0 #saving the return value to $s1
            bne $s1, -1, valid #a valid input is received
            la $a0, error #printing an error message
            li $v0, 4
            syscall
            j again #reading again in case of an invalid input
```

```
            valid:
                  la $a0, result #printing a message to display the result
                  li $v0, 4
                  syscall
                  move $a0, $s1 #printing the decimal value
                  li $v0, 1
                  syscall
            lw $ra, 0($sp) #pop the values to reload the originals
            lw $s1, 4($sp)
            addi $sp, $sp, 8
            jr $ra #return

#The data segment
            .data
hexNo:      .space    8 #after that MIPS gives the wrong results for some
bigger hexadecimals
intro:      .asciiz   "The program converts a hexadecimal number to a
decimal number.\n"
prompt:     .asciiz   "Please enter a hexadecimal number (max 7 digits -
MIPS is not capable of more):\n"
result:     .asciiz   "\nThe corresponding decimal number is:\n"
error:      .asciiz   "You entered an invalid hexadecimal number, try
again.\n"
```

2. Machine Instructions for the second question:

Jump instructions (j, [PC = JumpAddr] of opcode 0x2) are of the type:



Where, the first 4 bits of the 32-bit Jump Address is provided by the Program Counter and the last 2 bits are omitted since they are always 00 (since the addresses of the instructions are always in word boundary).

The address of the previous label is given as: 0x1001001C

A nop (no operation) occupies 4 words. There are 5 nops, 1 j and 1 add between the previous label and the next label. Hence,

The address of the next label is: 0x1001001C + 28 = 0x10010038

Thus, the machine instruction for

j next is: 000010 0000 0000 0001 0000 0000 0011 10
0000 1000 0000 0000 0100 0000 0000 1110
0x0800400E

j previous is: 000010 0000 0000 0001 0000 0000 0001 11
0000 1000 0000 0000 0100 0000 0000 0111
0x08004007