CS 224

Section No: 02
Fall 2019

Lab No: 3

İrem Seven
21704269

1) Floating Point Numbers Problem Solving:

a) -77.125

77 = 1001101

0.125 = 2 ^ -3

Single Precision:

1.001101001 * 2 ^ 6

Bias: 127

127 + 6 = 133

Sign = 1

Exponent = 10000101

Mantissa = 001101001 00000000000000

Hexadecimal : **0xc29a4000**

Double Precision:

Bias: 1023

Sign = 1

Exponent = 10000000101

Mantissa = 001101001 00000000000

Hexadecimal **: 0xc0534800**

b) -77.125

77 = 1001101

0.125 = 2 ^ -3

Single Precision:

1.001101001 * 2 ^ 6

Bias: 120

120 + 6 = 126

Sign = 1

Exponent = 01111110

Mantissa = 001101001 00000000000000

Hexadecimal **: 0xbf1a4000**

Double Precision:

Bias: 1020

1020 + 6 = 1026

Sign = 1

Exponent = 10000000010

Mantissa = 001101001 00000000000

Hexadecimal **: 0xc0234800**

c) 0xc1a00000

Binary: 1100 0001 1010 0000 0000 0000 0000 0000

: 1/ 100 0001 1/ 010 0000 0000 0000 0000 0000

Sign: 1

Exponent: 128 + 1 + 2 – 127 = 4

Mantissa: 0.01 (binary) = 0.25 (decimal)

(-1) * (1 + 0.25) * 2 ^ 4 = **-20**

Result : -20

Part 2) recursiveSummation:

```
.data
    string: .asciiz "1204"

    .text
        .globl _start
    _start:

    #this part calculates the lenght of the string
    size:
        lbu $s1,string($s0) #s0: string lenght
        beq $s1,$zero,done #if its null terminator
        addi $s0,$s0,1
        j size
        done:

    la $a0,string
    add $s0,$s0,$a0

    la $a0,string
    jal recursive

    add $a0,$v0,$zero
    li $v0,1
    syscall

    li $v0,10
    syscall

    recursive:
        addi $sp,$sp,-8
```

```
sw $t0,0($sp) #saving to stack since its value shouldnt change during recursive returning
sw $ra,4($sp)
#Base case
addi $v0,$v0,0
beq $a0,$s0,donef
#Do job
move $t0,$a0
addi $a0,$a0,1
jal recursive
lbu $t1,0($t0)
addi $t1,$t1,-48 #convert ascii to decimal
add $v0,$v0,$t1

donef:
        lw $t0,0($sp)
        lw $ra,4($sp)
        addi $sp,$sp,8
        jr $ra
```

Part 3) deleteAfter_x:

```
Delete_x:       #this subprogram works with given linked list main menu provided that:
    # move $a1, $v0 (v0 is x taken from user)
    # move $a0, $s0 before jal Delete_x
    #it gives num of elements in v0 and also list pointer s0 in v1
# We are not able to return the deleted node back to the heap since we are losing the pointer it
# points to. Although the data is in the heap it cannot be accessed without a pointer that points to it.

addi $sp, $sp,-8 # make room on stack for 2 new items

sw $s0, 4 ($sp) # push $s0 value onto stack

sw $s1, 0 ($sp) # push $s1 value onto stack
```

```
        addi $t4,$zero,0 #number of elements that deleted

        move $s0, $a0          # put the pointer to the current element in $s0

        bne $s0, $zero, devam88      # if pointer is NULL, there is no list

        la $a0, msg82          # its in the given code - put msg82 address into a0 NULL POINTER EROR

        li $v0, 4      # system call to print

        syscall                #   out the msg82 string

        j Return88  # done, so go home

devam88:                # top of loop

        lw $s1, ($s0)          # read the value of pointerToNext
```

```
    lw $t5, 4($s0)          # read the value part, put into a0

    beq $t5,$a1,deletenext # it allows to delete the elements after first number

Top88:  beq $s1, $zero, Return88# if pointerToNext is NULL, there are no more elements

    move $s0, $s1           # update the current pointer, to point to the new element

    lw $s1, ($s0)           # read the value of pointerToNext in current element

    lw $t5, 4($s0)          # read the value part, put into a0

    beq $t5,$a1,deletenext

    j deletepass

    deletenext:   #connects the current node's next pointer to the node which occurs after deleted node

        beq $s1,$zero,remain #if its last element dont change list
```

```
        lw $t6,($s1)
        sw $t6, ($s0)
        lw $s1,($s0)
        addi $t4,$t4,1    #increment deleted element number
    remain:

    deletepass:

    j Top88       # go back to top of loop

Return88:

    lw $s0, 4 ($sp) # restore $s0 value from stack


    lw $s1, 0 ($sp) # restore $s1 value from stack


    addi $sp, $sp, 8 # restore $sp to original value (i.e. pop 2 items)

    move $v0, $t4   #return number of deleted elements
    move $v1, $s0  #return original list pointer

    jr $ra                  # return to point of call
```