AuBuchon Lab-12 (50 points)

Q1 (25 points)

Draw the flow chart and explain how “printDec” function, presented in Lecture#11, work.

Explain the main idea of the function and the task of each code line

printDec**:**

section **.bss**

decstr resb 10 ;10 digit variable for 32 bits

ct1 resd 1 ;keep track of string size

section **.text**

**pusha**

**mov** **dword[**ct1**],**0 ;assume initially 0

**mov** **edi,**decstr ;edi points to dec-string in memory

**add** **edi,**9 ;mov the last element of string

**xor** **edx,** **edx** ;clear out edx for 64 bit division

whileNotZero**:**

**mov** **ebx,** 10 ;store 10 for division

**div** **ebx** ;divide by 10

**add** **ebx,**'0' ;convert to ascii char

**mov** **byte[edi],dl** ;move to string

**dec** **edi** ; mov to next char in string

**inc** **dword[**ct1**]** ;increment char counter

**xor** **edx,** **edx** ;clear edx

**cmp** **eax,** 0 ;is remainder 0

**jne** whileNotZero ;if not keep looping

**inc** **edi** ;conversion, finish, bring edi

**mov** **ecx,** **edi** ;back to beginning of string, make ecx

**mov** **edx,** **[**ct1**]** ;point to it, and edx gets # chars

**mov** **eax,** 4 ;and print! to the stndout

**mov** **ebx,** 1

**int** 80h

**popa**

**ret**

printString**:**

;save register values of the called function

**pusha**

;string is pointed by ecx, edx has it's length

**mov** **eax,** 4

**mov** **ebx,** 1

**int** 80h

;return old register values of the called function

**popa**

**ret**

printLn**:**

section .data

n1 db 10

section **.text**

**pusha**

**mov** **ecx,** n1

**mov** **edx,** 1

**call** printString

**popa**

**ret**

;saves all the registers so that they are not changed by the function

;we build the function to handle the dword size

;print1.asm

section .data

z1 dd 232

z2 dd 434

z3 dd 0

msgZ db "z="

section **.text**

global main

main**:**

**push** **ebp**

**mov** **ebp,** **esp**

;print z1 as ascii

**mov** **eax,** **[**z1**]**

**call** printDec

**call** printLn

;print z2 as ascii

**mov** **eax,** **[**z2**]**

**call** printDec

**call** printLn

;add z1 and z2

**mov** **eax,** **[**z1**]**

**add** **eax,** **[**z2**]**

;mov to z3

**mov** **[**z3**],** **eax**

**;print z3**

**call** printDec

**call** printLn

;exit

**mov** **esp,** **ebp**

**pop** **ebp**

**ret**

**dec** **edi**

move the pointer down to the next decimal

**inc** **dword[**ct1**]**

increase counter by 1 to keep track of length of new string

**popa**

**ret**

return all registers values from calling function

decstr resb 10

initializes decstr variable, 10 digit number or 32 bits

**xor** **edx,** **edx**

zero out edx for next loop division

**int** 80h

system call

**mov** **byte[edi],dl**

move lower 8 bit address into byte[edi]

ct1 resd 1

Initializes counter variable to keep track of string size inside of loop

**cmp** **eax,** 0

see if all the decimals have been converted by looking at remainder of ebx division

**mov** **ebx,** 1

specify standard output for where to write system call

**add** **ebx,**'0'

convert corresponding decimal value into ascii by adding 48

**pusha**

saves all stack register values from main function

**jne** whileNotZero

If decimals have not all been converted in memory restart loop

**mov** **eax,** 4

mov 4 to eax in prep for write system call

**div** **ebx**

divide ebx by 10

**mov** **dword[**ct1**],**0

put 0 in the counter variable in preparation for the loop

**mov** **edx,** **[**ct1**]**

mov length counter to edx in preparation for write system call

**mov** **edi,**decstr

grabs the decimal from memory

whileNotZero**:**

**mov** **ebx,** 10

beginning of loop for all elements within decimal and stores 10 in ebx for division

no

yes

**mov** **ecx,** **edi**

move string start location to ecx in prep for write system call

**add** **edi,**9

moves to the last element of the decimal

**xor** **edx,** **edx**

clears edx register for 64 bit division

**inc** **edi**

conversion complete, point edi to beginning of string

Q2 (25 points)

1- Create a file “file1.txt” that has your name

2- Using system calls, write assembly program to copy “fileIN.txt” to a new file “fileOUT.txt”

3- Run the code and take a screen shot of the output

4- In one file, include your code and the screen shot of the run

