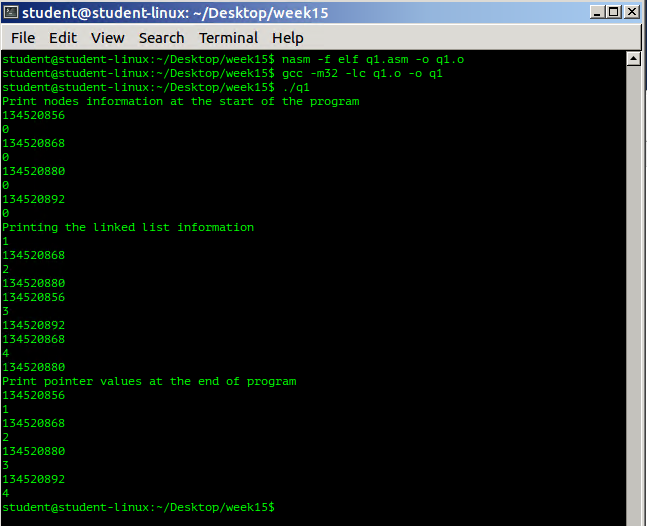
AuBuchon Lab-15 (50 points)

Write assembly program using system calls that implement doubly linked list with 4 nodes.

1. Define and declare a node structure that has key value, next pointer and previous pointer
2. Build doubly linked list from of 4 nodes
3. The key values of nodes are 1, 2, 3, and 4
4. Print the fields contents for each node



;lab15.asm

STRUC Node

.Value**:** resd 1

.NextPtr**:** resd 1

.PreviousPtr resd 1

.size**:**

ENDSTRUC

section .data

;declare four nodes and create the linked list

Head**:** ISTRUC Node

AT Node.Value**,** dd 0

AT Node.NextPtr**,** dd Second

AT Node.PreviousPtr**,** dd 0

IEND

Second**:** ISTRUC Node

AT Node.Value**,** dd 0

At Node.NextPtr**,** dd Third

AT Node.PreviousPtr**,** dd Head

IEND

Third**:** ISTRUC Node

AT Node.Value**,** dd 0

At Node.NextPtr**,** dd Tail

AT Node.PreviousPtr**,** dd Second

IEND

Tail**:**ISTRUC Node

AT Node.Value**,** dd 0

AT Node.NextPtr**,** dd 0

AT Node.PreviousPtr**,** dd Third

IEND

msg1**:** db"Print nodes information at the start of the program"**,**10**,**0

msgL1**:** equ **$-**msg1

msg2**:** db"Printing the linked list information"**,**10**,**0

msgL2**:** equ **$-**msg2

msg3**:** db"Print pointer values at the end of program"**,**10**,**0

msgL3**:** equ **$-**msg3

section **.bss**

section **.text**

global main

main**:**

**push** **ebp**

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

**mov** **ecx,** msg1 ;print start values

**mov** **edx,** msgL1

**call** printString

;print start value of each nodes

**mov** **eax,** Head ;memory location of head nodes

**call** printDec

**call** println

**mov** **eax,** **[**Head**]** ;print memory contents of head nodes

**call** printDec

**call** println

**mov** **eax,** Second ;memory location of second nodes

**call** printDec

**call** println

**mov** **eax,** **[**Second**]** ;contents of memory at Second

**call** printDec

**call** println

**mov** **eax,** Third ;memory location of Third nodes

**call** printDec

**call** println

**mov** **eax,** **[**Third**]** ;contents of memory at Third

**call** printDec

**call** println

**mov** **eax,** Tail ;memory location of tail nodes

**call** printDec

**call** println

**mov** **eax,** **[**Tail**]** ;memory contents at [Tail]

**call** printDec

**call** println

;set the head node value

**mov** **WORD[**Head **+** Node.Value**],**1

;set second node value

**mov** **WORD[**Second **+** Node.Value**],**2

;set the Third node value

**mov** **WORD[**Third **+** Node.Value**],**3

;set the tail node value

**mov** **WORD[**Tail **+** Node.Value**],**4

**mov** **ecx,** msg2

**mov** **edx,** msgL2

**call** printString

;print the data field of head node

**mov** **eax,** **[**Head **+** Node.Value**]** ;date value

**call** printDec

**call** println

;print the data field of head node

**mov** **eax,** **[**Head **+** Node.NextPtr**]** ;pointer value

**call** printDec

**call** println

;print the data field of Second node

**mov** **eax,** **[**Second **+** Node.Value**]** ;date value

**call** printDec

**call** println

;print the data field of Second node

**mov** **eax,** **[**Second **+** Node.NextPtr**]** ;pointer value

**call** printDec

**call** println

;print the data field of Second node

**mov** **eax,** **[**Second **+** Node.PreviousPtr**]** ;pointer value

**call** printDec

**call** println

;print the data field of Third node

**mov** **eax,** **[**Third **+** Node.Value**]** ;date value

**call** printDec

**call** println

;print the data field of Third node

**mov** **eax,** **[**Third **+** Node.NextPtr**]** ;pointer value

**call** printDec

**call** println

;print the data field of Third node

**mov** **eax,** **[**Third **+** Node.PreviousPtr**]** ;pointer value

**call** printDec

**call** println

;print the data field of Tail node

**mov** **eax,** **[**Tail **+** Node.Value**]** ;date value

**call** printDec

**call** println

;print the data field of Tail node

**mov** **eax,** **[**Tail **+** Node.PreviousPtr**]** ;date value

**call** printDec

**call** println

**mov** **ecx,** msg3

**mov** **edx,** msgL3

**call** printString

;print start value of each nodes

**mov** **eax,** Head ;memory location of head nodes

**call** printDec

**call** println

**mov** **eax,** **[**Head**]** ;print memory contents of head nodes

**call** printDec

**call** println

**mov** **eax,** Second ;memory location of second nodes

**call** printDec

**call** println

**mov** **eax,** **[**Second**]** ;contents of memory at Second

**call** printDec

**call** println

**mov** **eax,** Third ;memory location of third nodes

**call** printDec

**call** println

**mov** **eax,** **[**Third**]** ;contents of memory at third

**call** printDec

**call** println

**mov** **eax,** Tail ;memory location of tail nodes

**call** printDec

**call** println

**mov** **eax,** **[**Tail**]** ;memory contents at [Tail]

**call** printDec

**call** println

;exit program and clean

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

**pop** **ebp**

**ret**

printDec**:**

section **.bss**

decstr resb 10

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** **edx,**'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**