Assembly and Machine Language - Spring 1397 (2018) Midterm Exam	Instructor: B. Nasihatkon	دانتگاه منتی خواجه سرالدین طوی ۲۰
Name:	ID:	Ordibehesht 1397 - May 2018

Question 1 Assume that the data section of your assembly code is as follows

1b11: dd 0 After running the following assembly commands

- xor EAX, EAX
 mov AX, -1
 mov AL, 7Dh
 mov [lbl1], EAX
 mov CL, [lbl1+1]
- a) What will be the **binary** representation of AL? Why? (3 points)
- b) As an unsigned integer, what decimal number does AL represent? Why? (3 pts)
- c) As a **2's complement signed integer**, what decimal number does AL represent? Why? (3 points)
- d) What is the Hexadecimal representation of AX? Why? (3 pts)
- e) As an **unsigned integer**, what decimal number does **AX** represent? Why? (3 pts)
- f) As a **2's complement signed integer**, what decimal number does AX represent? Why? (3 points)
- g) As an **unsigned integer**, what decimal number does **CL** represent? Why? Assume a little-endian system. (3 points)

Functions from the book

call print_int	prints EAX as an unsigned integer
call print_nl	prints a new line character
call read_int	reads an integer and stores it in EAX.
call print_string	prints a null-terminated string whose starting address is in EAX

Programming

Write programs in the designated code area as follows:

label	command	arguments
loop1:	call	prog2
	jne	loop1
prog2:		

Question 2 What does the following assembly code compute? Explain the relationship between the user input and the program output. (18 points)

```
call read int
       mov edi, eax
       mov ecx, 0
       mov ebx,1
       mov esi,1
loop1: cmp ebx, edi
       jg endloop1
       mov eax, ebx
       imul eax
       imul esi
       add ecx, eax
       neg esi
       inc ebx
       jmp loop1
endloop1:
       mov eax, ecx
       call print int
       call print nl
```

Question 3 Write an assembly program that checks the bits of the **AL** register. If bit 0 is equal to 1, bit 1 is equal to 0, and bit 6 is not equal to bit 4 it must print YES. Otherwise, it has to print NO. The **AL** register must remain unchanged (before printing the output). (16 points)

label	command	arguments	label	command	arguments
segment .	data				
yes_msg:	db "YES", 10, 0				
no_msg:	db "NO", 10, 0				
segment .text					

Question 4 The function sum (below on the right) takes 3 arguments, call them a,b and k. It computes and returns the sum of numbers between a and b with a skip of k, that is a + (a+k) + (a+2k) + ... + (a+mk) such that $(a+mk) \le b$.

The code on the left reads a,b and k, calls the sum function with these as arguments, and prints the return value. But there are **7** mistakes in the code (instructions to be modified, removed or added). Find and fix them. (20 points)

```
call read int
                             sum:
mov ebx, eax
                              push ebp
                              mov esp,ebp
call read int
                              push eax
mov ecx, eax
                              mov eax, 0
                               mov ecx, [ebp+16]
call read int
                             loop1:
                               cmp ecx, [ebp+12]
push eax
                               jg endloop1
push ecx
                               add eax, ecx
push ebx
call sum
                               add ecx, [ebp+8]
add esp, 12
                               jmp loop1
                             endloop1:
call print int
                               pop eax
call print nl
                               ret 12
```

Question 5 We want to implement a function which takes 2 positive integers **a,b** as arguments and **returns 2 return values**: **a/b** and **a%b**.

function divide(a,b) {
 return a/b, a%b
}

To do this, we follow a special calling convention in which both the parameters and return values are stored on the stack. The parameters and return values must be put on the stack as in the figure on the right. Notice that you need to reserve memory space for return values before calling the subprogram. In the left column below write down a piece of code that reads **a** and then **b** from input (can store them in



registers), calls the divide function to compute a/b and a%b, and then prints the values of a/b and then a%b given by the function divide. In the right column write down the body of the function divide. Your code must be fully compatible with the convention mentioned above. (25 points)

label	command	arguments	label	command	arguments
			divide:		