Assembly and Machine Language - Fall 1397 (2018) Midterm Exam	Instructor: B. Nasihatkon	دانشگاه صنعتی خواجه نصیرالدین طوسی K. N. TOOSI UNIVERSITY OF TECHNOLOGY
Name:	**ID:**	**Azar 1397 - November 2018**

Functions from the book

`call print_int`	prints EAX as a signed integer
`call print_nl`	prints a newline character

Use 32-bit Netwide assembler code on a Linux machine.

Programming

Write programs in the designated code area as follows:

label	command	arguments
`loop1:`	`call`	`prog2`
	`add`	`eax, ebx`
`prog2:`		

Question 1 (16 points) After running the next assembly instructions

```
mov AX, 12
shl AX, 2
mov AL, 8Eh
not AX
```

a) What will be the **binary** representation of AX? Why? (4 points)

b) What is the Hexadecimal representation of AX? Why? (4 points)

c) As an **unsigned integer**, what **decimal** number does AX represent? Why? (4 pts)

d) As a **2's complement signed integer**, what decimal number does AX represent? Why? (4 points)

Question 2 The following assembly code prints five lines of output. What number is printed in each line and why? Assume a little-endian architecture. You may write the answers as a sum of products. (15 points)

```
segment .data

lbl: dd 1,10,100,1000,10000

segment .text

        ⋮
    mov eax, [lbl]
    call print_int
    call print_nl

    mov eax, [lbl+1]
    call print_int
    call print_nl

    mov eax, [lbl+2]
    call print_int
    call print_nl

    mov eax, [lbl+3]
    call print_int
    call print_nl

    mov eax, [lbl+4]
    call print_int
    call print_nl
```

Question 3 In each piece of assembly code in the left column, write a single assembly instruction performing the computations on **EAX** and also **EDX** (if they change). Explain your answer. (22 points)

	Single Instruction	**Explanation**
`neg eax` `dec eax`		
`not eax` `xor eax, -2`		

`cmp eax, 0`
`jge positive`
 	
`mov edx,-1`
`jmp end1`
`positive:`
`mov edx, 0`
`end1:`		

`mov ecx, 32`
`loop1:`
`xor eax, 1`
`ror eax, 1`
 	
`loop loop1`		

`mov ebx, 1`
`loop1:`
`xor eax, ebx`
`test eax, ebx`
`jnz endloop1`
 	
`shl ebx,1`
`jnc loop1`
`endloop1:`		

Question 4 The C function **gcd** on the left receives two parameters and computes their Greatest Common Divisor (GCD) using the formula **GCD(a,b) = GCD(b,a%b)**. Complete the assembly program on the right to call **gcd(16,12)** and print the return value using the **printf** function from the C standard library. You are not allowed to use the **print_int** function. (20 points)

```
int gcd(int a, int b) {
  int r;

  while (b != 0) {
    r = a % b;
    a = b;
    b = r;
  }

  return a;

}
```

label	command	arguments
segment .data		
segment .text extern global ; call gcd(16,12) and print the result		
main:		
	mov	ebx, 0
	mov	eax, 1
	int	0x80

Question 5 We do the opposite of Question 4. Now, you have to write the gcd function in assembly such that the C code on the left is able to call it. Complete the assembly code to write the body of the gcd function. Use appropriate directives (global, extern, etc.) if needed. **Observe all C calling conventions. Your algorithm must be the same as the C function in Question 4.** (27 points)

```c
#include <stdio.h>

int gcd(int a, int b);

int main() {
  int c;

  c = gcd(16,12);

  printf("%d\n", c);

  return 0;
}
```

label	command	arguments
segment .text		
gcd:	push	ebp
	mov	ebp, esp

label	command	arguments