1. FELADAT: Adott az alábbi forráskód.
a) Add meg a *main* program kifejtett szövegét!
b) Add meg, hogy a program végén mi lesz az EAX, EBX, ECX regiszterek tartalma!

```
extern printf
section .data
        kiir db "%d %d %d",0xa,0
section .bss
n equ 10
vektor resd n

%assign i 0             ;i:=0
%macro ha 3
        CMP %1, %3  ;első és harmadik param.
        J%+2 %%igaz         ;"belső jmp utasítás"
        JMP hamis %+ i      ;"belső jmp utasítás"
%%igaz
%endm

%macro havege 0
hamis %+ i
%assign i i+1           ;i:=i+1
%endm
```

```
section .text
global main
main:
        mov al, 'B'
        ha al, G, 'A'
                mov eax, 20h
        havege
        ha al, E, 32
                mov ecx, n
                mov edi, vektor
                cld
                rep stosd
        havege
                mov ebx, [vektor+4*9]

                ret
```

```
[section .bss]
n equ 10
vektor resd n
```

```
[section .text]
[global main]
main:
        MOV AL, 'B'
        CMP AL, 'A'
                JG ..@3.igaz
                JMP hamis0
                ..@3.igaz
                        MOV EAX, 20h
        hamis0
        CMP AL, 32
                JE ..@5.igaz
                JMP hamis1
                ..@5.igaz
                        MOV ECX, n
                        MOV EDI,
vektor
                        CLD
                        REP STOSD
        hamis1
        MOV EBX, [vektor+4*9]

        ret
```

A regiszterek tartalma:

EAX = 32
EBX = 32
ECX = 0

2. FELADAT *(megoldás assembly-vel és C-vel)*

Tegyük fel, hogy egy program adatszekciója az alábbi, ami 2 hatványait tárolja! Írj olyan függvényt, amelynek bemenő paramétere egy 0 és 30 közötti szám és kimenő értéke a 2 megfelelő hatványa, azaz az *int exp2(int n)* függvényt kell megírni. Ha nem jó a bemenő érték, akkor a adjon a függvény -1-t vissza. A függvény konstans időben adja vissza a 2^n értéket!

```
section .data
%assign i 1
hatvany:
%rep 31
     dd i
%assign i i*2
%endrep
```

[MEGOLDAS2A.ASM]

```
section .bss
kitevo resd 1

section .data
%assign i 1

hatvany:
%rep 31
    dd i
%assign i i*2
%endrep

section .text
global exp2

exp2:
    cmp [esp+4],dword 0
    jl rossz
    cmp [esp+4],dword 30
    jg rossz
    mov ecx, [esp+4]
    mov eax, [hatvany+ecx*4]
    ret

rossz:
    mov eax,-1
    ret
```

[MEGOLDAS2A.C]

```
#include <stdio.h>

extern int exp2 ( int n );

int main()
{
  int x, y;
  printf("Hányadik kettőhatványt számoljam ki? ");
  scanf("%d", &x);

  y = exp2(x);

  if (y==-1)
  {
    printf("A szám 0-nál kisebb, 30-nál nagyobb,
vagy hibás!\n");
  }
  else if (y < -1)
  {
    printf("Nem várt hiba!\n");
  }
  else
  {
    printf("A(z) %d-ik kettőhatvány: %d\n", x, y);
  }

  return 0;
}
```

2. FELADAT *(megoldás csak assembly-vel)*

Tegyük fel, hogy egy program adatszekciója az alábbi, ami 2 hatványait tárolja! Írj olyan függvényt, amelynek bemenő paramétere egy 0 és 30 közötti szám és kimenő értéke a 2 megfelelő hatványa, azaz az *int exp2(int n)* függvényt kell megírni. Ha nem jó a bemenő érték, akkor a adjon a függvény -1-t vissza. A függvény konstans időben adja vissza a 2^n értéket!

```
section .data
%assign i 1
hatvany:
%rep 31
     dd i
%assign i i*2
%endrep
```

[MEGOLDAS2B.ASM]

```
;a 30-nál nagyobb számokra még nincs megoldva a hibakezelés

extern printf

section .data
%assign i 1

hatvany:
%rep 31
    dd i
%assign i i*2
%endrep

index       dd    10
kiir        db    "%d", 0xA, 0

section .text
global main

main:
     push dword [index]
     call exp2
     add esp,4
     ret
exp2:
     cmp [esp+4],dword 0
     jl rossz
     jmp folyt
rossz:
    mov eax,-1
    ret
folyt:
    mov ecx, [esp+4]
    mov eax, [hatvany+ecx*4]

    push eax
    push kiir
    call printf
    add esp,8
    ret
```

3. FELADAT *(megoldás assembly-vel és C-vel)*

Írj rekurzív assembly eljárást az alábbi függvénydefiníciónak megfelelően!

$f(n) = 3^n$, ahol $n \geq 0$, ennek rekurzív megfelelője:

$$f(n) = \begin{cases} 1, \text{ ha } n = 1 \\ f(n-1) + 2*f(n-1), \text{ ha } n \text{ pozitív egész.} \end{cases}$$

[MEGOLDAS3A.ASM]

```
section .bss
eredmeny resd 1

section .text
global Haromhatvany

Haromhatvany:
    cmp [esp+4],dword 0
    jne folyt1
    mov eax, 1
    ret

folyt1:
    cmp [esp+4],dword 1
    jge folyt2
    mov eax,-1
    ret

folyt2:
    mov eax,[esp+4]
    dec eax
    push eax
    call Haromhatvany
    pop ebx
    push eax
    sal eax,1
    pop ebx
    add eax, ebx
    ret
```

[MEGOLDAS3A.C]

```
#include <stdio.h>

extern int Haromhatvany ( int adat );

int main()
{
  int x, y;
  printf("Hányadik háromhatványt számoljam ki? ");
  scanf("%d", &x);

  y = Haromhatvany(x);

  if (y==-1)
  {
    printf("A szám 0-nál kisebb, vagy hibás!\n");
  }
  else
  {
    printf("A(z) %d-ik háromhatvány: %d\n", x, y);
  }

  return 0;
}
```

3. FELADAT ***(megoldás assembly-vel)***

Írj rekurzív assembly eljárást az alábbi függvénydefiníciónak megfelelően!

$f(n) = 3^n$, ahol $n \geq 0$, ennek rekurzív megfelelője:

$$f(n) = \begin{cases} 1, \text{ ha } n = 1 \\ f(n-1) + 2*f(n-1), \text{ ha n pozitív egész.} \end{cases}$$

```
extern printf

section .data
     n    dd 4
     kiir db "%d", 0xa,0

section .text
global main

main:
     push dword [n]
     call exp3
     pop edx

     push eax
     push kiir
     call printf
     add esp, 8
     ret
exp3:
     cmp [esp+4], dword 0
     jl rossz
     jg ag2
     mov eax, 1
     ret
rossz:
     mov eax, -1
     ret
ag2: mov eax, [esp+4]
     dec eax

     push eax
     call exp3

     pop ebx
     push eax  ; lementem f(n-1)-t
     sal eax, 1 ; 2*f(n-1)
     pop ebx
     add eax, ebx
     ret
```