```
Loopuim:
    mov dword [k], ECX
    mov EBX, dword [copie]
    shr EBX, 1
    mov dword [copie], EBX
    jc inmultim
    jnc Trecem

inmultim:
    ADD EDX, dword [p]

Trecem:
    mov EAX, dword [p]
    mov EBX, 2
    mul EBX
    mov dword [p], EAX
```

i în p se află puterea lui 2 corespunzătoare transf. din baza 2 în baza 10 pentru bit-ul curent din EBX / [copie]

```
    mov ECX, dword [k]
    Loop Loopuim
```

④

; acum, în EDX se află
~~nr~~ nr. în baza 10
de pe poz curentă din
ESi

mov ECX, ~~0~~ EDX ;

~~Suma Cifrelor:~~

jECXZ Punem în şir

mov EAX, EDX > mov ECX, 0 ; adunăm în ECX cifrele din EDX

Suma Cifrelor:

mov EAX, EAX

mov EDX, 0 > mov EBX, 10

DiV EBX ; EAX = câtul EDX:EAX / 10
EDX = restul

ADD ECX, EDX
CMP EAX, 0
je Punem în şir
jne Suma Cifrelor

Punem în şir:

mov EAX, ECX
stosd

mov ECX, dword [k]

Loop Repeta

(5) ret 4 * 3 ; în EDi se află şirul dorit

afiseaza:

```
mov esi, dword [esp+4]
mov edi, dword [esp+8]
mov ecx, dword [esp+12]
```

Grupa 212 22-07-2021
Eusiac Andrei Test

a.asm

```
b:ts 32

global start
extern exit, construieste, afiseaza
import exit msvcrt.dll
segment data use32 class=data
sir dd ....
 len equ ($ - sir) / 4
d times len dd 0

segment code use 32 class=code
    push dword len
    push dword d
    push dword sir
    call construieste
```

~~push dw~~

```
mov EDi, d
push dword EDi len
push dword EDi
push dword s:r
call afiseata
push dword 0
call [exit]
```

b. asm

```
bits 32

Segment code use 32 class= data
    k dd 0          copie dd 0
    p dd 1          ~~cont dd 0~~
    k1 dd 0


segment code use 32 class= code
                                public
global construieste, afiseata

construieste:

    mov esi, [esp+4] ; sirul initial
    mov edi, [esp+8] ; sirul dest.
    mov ecx, [esp+12] ; lungimea
    cld
    Repeta:  mov dword [k], ECX
        lodsd ; EAX= elem. curent din
                              sirul initial
        moV EDX, 0
        mov EBX, EAX

        moV ECX, 32
        moV dword [copie], EBX
```

(3)