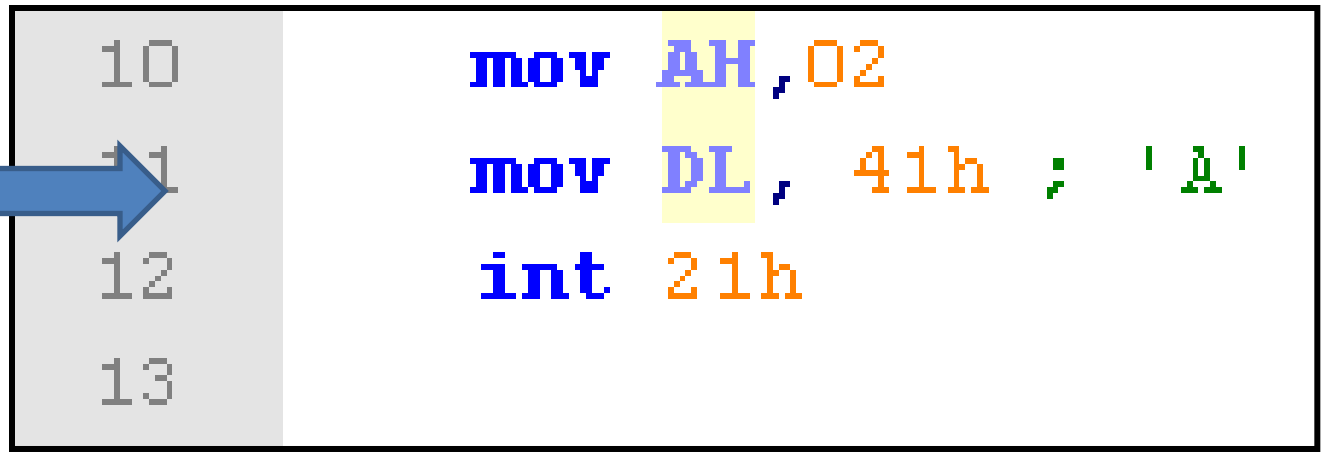
PRUC

2. Gyakorlat

Egyszerű feladatok

„A” betű kiíratása

```
1
2   Code    Segment
3       assume CS:Code, DS:Data, SS:Stack
4
5   Start:
6       mov ax, Code
7       mov DS, AX
8
9       ;comment :)
10      mov AH,02
11      mov DL, 41h ; 'A'
12      int 21h
13
14
15
16
17  Program_Vege:
18      mov ax, 4c00h
19      int 21h
20
21
22
23  Code    Ends
24
25  Data    Segment
26
27  Data    Ends
28
29  Stack   Segment
30
31  Stack   Ends
32      End Start
33
```



Utolsó sor=Sortörés!!!

Fordítási folyamat

```
d:\!oktatas\2011_tavasz\PRUC\masm>masm 02_01

Macro Assembler Version 5.10

Object filename [02_01.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  48060 + 411055 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

d:\!oktatas\2011_T~1\PRUC\masm>link 02_01

Microsoft (R) Overlay Linker  Version 3.61
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [02_01.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

d:\!oktatas\2011_T~1\PRUC\masm>02_01.EXE
A
d:\!oktatas\2011_T~1\PRUC\masm>_
```

Karakter kiíratása

INT 21h: DOS functions

DOS Fn 02h: Display Output

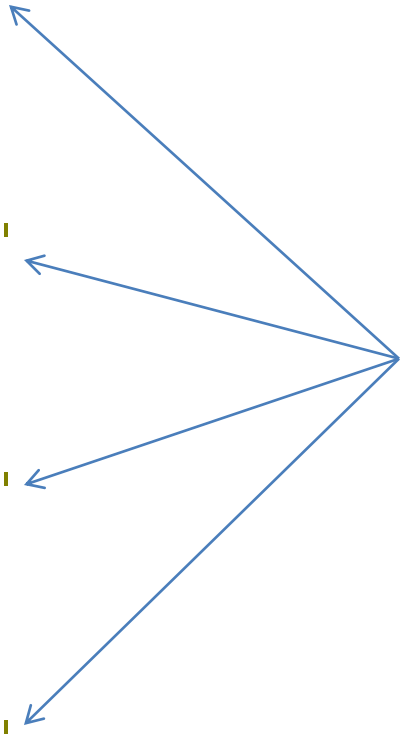
```
                    DOS Fn 02H: Display Output
                                                    Compatibility: 1.0+
Expects: AH     02H
         DL     character to send to the Standard Output
-------------------------------------------------------------------------
Returns: none
-------------------------------------------------------------------------
   Info: Sends the character in DL to the Standard Output. Handles
         backspace (ASCII 8) by moving the cursor left and leaving the
         cursor there.

         If Ctrl-Break is detected, INT 23H is executed.
```

Xview

```
11      mov AH,02
12      mov DL, 41h ;"A"
13      int 21h
```

Név kiíratása

```
10      mov AH,02
11      mov DL, 'D'
12      int 21h
13
14      mov AH,02
15      mov DL, 'a'
16      int 21h
17
18      mov AH,02
19      mov DL, 'n'
20      int 21h
21
22      mov AH,02
23      mov DL, 'i'
24      int 21h
```



Fordító elvégzi a konverziót!

String kiíratása

```
1
2   Code    Segment
3       assume CS:Code, DS:Data, SS:Stack
4
5   Start:
6       mov ax, Code
7       mov DS, AX
8
9       ;Uzenet kiirasa
10      mov dx, offset uzenet
11      mov ah, 09
12      int 21h
13
14  Program_Vege:
15      mov ax, 4c00h
16      int 21h
17
18  uzenet  db "Ez egy uzenet!$"
19
20  Code    Ends
21
22  Data    Segment
23
24  Data    Ends
25
26  Stack   Segment
27
28  Stack   Ends
29      End Start
30
```

DOS Fn 09H: Display String

Compatibility: 1.0+

Expects: AH 09H
DS:DX address of a string terminated with a '$' (ASCII 24H)

Returns: none

Info: The string, up to the terminating character '$' is sent to the Standard Output.

Backspaces are handled as in the 02H Display Char function.

The Normal procedure for displaying a 'newline' is to embed a CR/LF pair (ASCII 0dH followed by ASCII 0aH) in the string.

Strings containing '$' may be printed via 40H Write File (BX=0).

See Also: Character I/O Functions
DOS Functions

String kiíratása

```
5   Start:
6       mov ax, Code
7       mov DS, AX
8
9       ;Uzenet kiirasa
10      mov dx, offset uzenet
11      mov ah, 09
12      int 21h
13
14  Program_Vege:
15      mov ax, 4c00h
16      int 21h
17
18  uzenet   db "Ez egy uzenet!$"
19
20  Code     Ends
```

Különböző karakterek kiíratása

- 2 (szám)

```
5   Start:
6       mov ax, Code
7       mov DS, AX
8
9
10      mov ah, 02
11      mov DL, 2
12      int 21h
13
14  Program_Vege:
15      mov ax, 4c00h
```

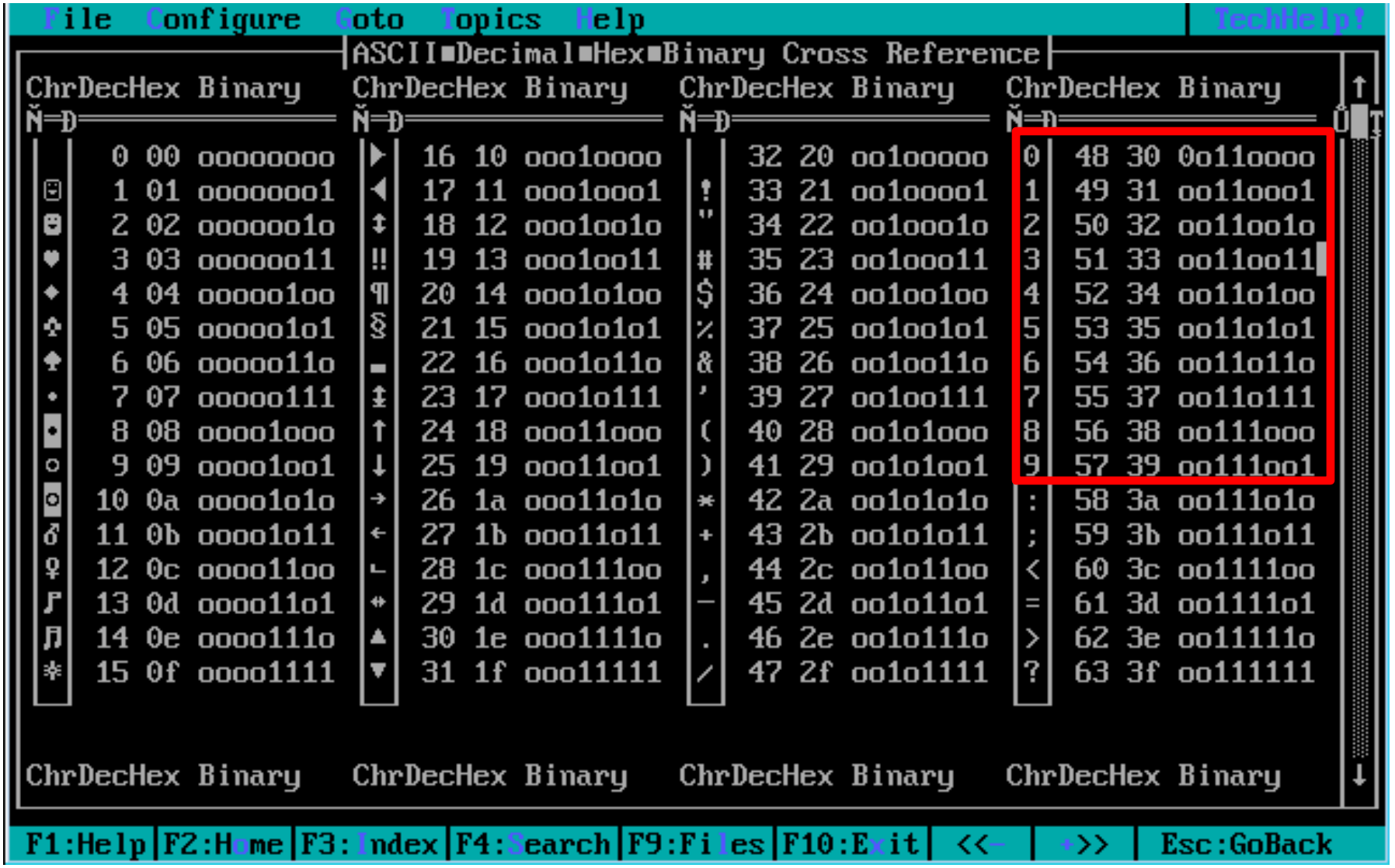
Különböző karakterek kiíratása

- 2 (szám)

```
d:\!oktatas\2011_T~1\PRUC\masm>02_01.EXE
☻
d:\!oktatas\2011_T~1\PRUC\masm>
```

```
 5  Start:
 6      mov  ax, Code
 7      mov  DS, AX
 8
12      int  21h
13
14  Program_Vege:
15      mov  ax, 4c00h
```

Különböző karakterek kiíratása



Különböző karakterek kiíratása

- 0 (karakter) többféle módon:

```
10      mov AH,02
11      mov DL, 48
12      int 21h
13
14      mov AH,02
15      mov DL, 30h
16      int 21h
17
18      mov AH,02
19      mov DL, 00110000b
20      int 21h
21
22      mov AH,02
23      mov DL, '0'
24      int 21h
25
```

Különböző karakterek kiíratása

- Tetszőleges szám (pl. 2)

```
5   Start:
6       mov ax, Code
7       mov DS, AX
8
9       mov BL,2
10      add BL, '0' ; 48d     (48 ... 57 = 0 ... 9)
11
12      mov ah, 02
13      mov DL, BL
14      int 21h
15
16  Program_Vege:
17      mov ax, 4c00h
18      int 21h
19
```

Műveletek
MOV

- MOV cél,forrás
  - a forrásoperandus tartalmát átmásolja a céloperandusba

ADD

- ADD cél, forrás
  - két operandus összeadását végzi el, az eredményt a céloperandus helyére írja
- befolyásolt flag-ek

  A, C, O, P, S, Z

```
 9      mov BL, 2
10      add BL, 3
11
12      add BL, '0' ; 48d    (48 ... 57 = 0 ... 9)
13
14      mov ah, 02
15      mov DL, BL
16      int 21h
17
```

SUB

- SUB cél, forrás
  - forrásoperandust kivonja a céloperandusból, és az eredményt a céloperandusba teszi
- befolyásolt flag-ek

  O, S, Z, P, A, C

```
 9       mov BL, 8
10       sub BL, 3
11
12       add BL, '0' ; 48d     (48 ... 57 = 0 ... 9)
13
14       mov ah, 02
15       mov DL, BL
16       int 21h
```

CMP

- CMP cél, forrás
  - két operandus összehasonlítása
  - kivonja a forrásoperandust a céloperandusból, az operandusok tartalma változatlan marad, azonban a jelzők megváltoznak
- befolyásolt flag-ek

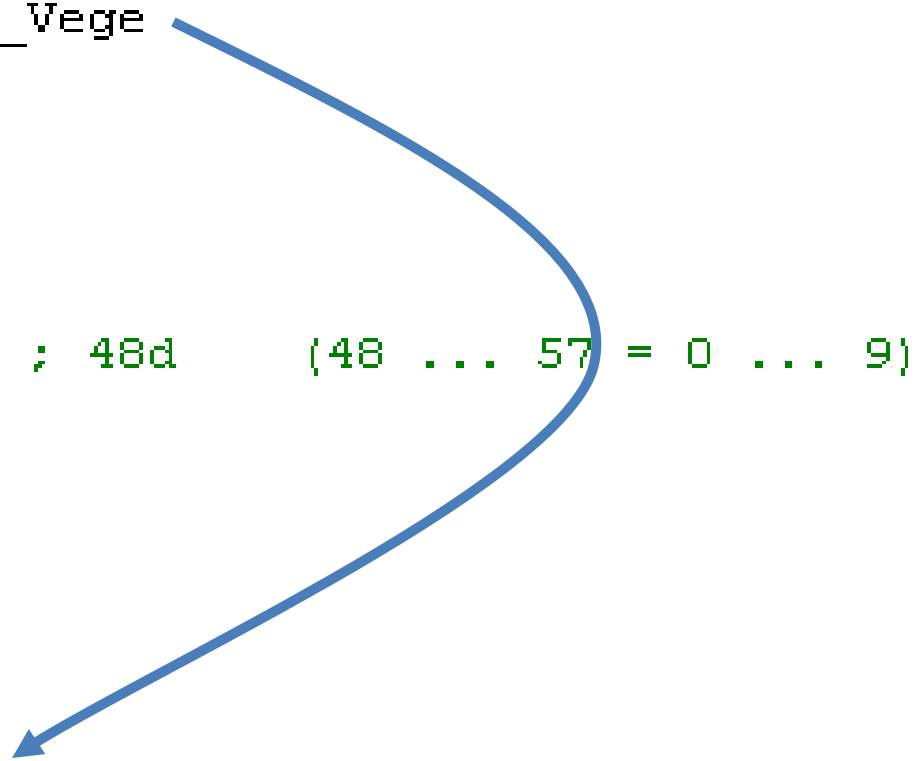
  O, S, Z, P, A, C

JMP

- JMP cél
- feltétel nélküli ugrás végrehajtása
  - hatására a program vezérlését egy másik utasítás kapja meg anélkül, hogy a visszatérésről bármilyen információt tárolna
  - ugrási távolság szegmensen belül bárhová

JMP

```
5   Start:
6       mov ax, Code
7       mov DS, AX
8
9       jmp Program_Vege
10
11      mov BL, 8
12      sub BL, 3
13
14      add BL, '0' ; 48d     (48 ... 57 = 0 ... 9)
15
16      mov ah, 02
17      mov DL, BL
18      int 21h
19
20  Program_Vege:
21      mov ax, 4c00h
22      int 21h
```



JZ,JNZ, JC, JNC

- feltételes ugrás végrehajtása
  - J(feltétel)
  - feltétel lehet pl. a jelzőbitek
  - ugrási távolság -128 -+ 127 bájt

JZ,JNZ, JC, JNC

```
5   Start:
6       mov ax, Code
7       mov DS, AX
8
9       mov dx, offset uzenet1
10      mov ah, 09
11      int 21h
12
13      mov BL, 8
14  ;if (BL== 2) { goto egyenlo_printf }
15      cmp BL, 2
16      jz Egyenlo
17  ;else
18  ; {
```

```
17  ;else
18  ; {
19  Nem_egyenlo:
20      mov dx, offset uzenet3
21      mov ah, 09
22      int 21h                 ;printf("Nem egyenlő")
23
24      jmp Program_vege        ; !!!!
25  ;}
26
27  Egyenlo:
28      mov dx, offset uzenet2
29      mov ah, 09
30      int 21h                 ;printf("Egyenlő")
31
32  Program_Vege:
33      mov ax, 4c00h
34      int 21h
35
36  uzenet1: db "~(BL == 2)? $"
37  uzenet2: db "Egyenlo$"
38  uzenet3: db "Ner  egyenlo$"
39
40  Code    Ends
```

```
5  Start:
6      mov ax, Code
7      mov DS, AX
8
9      mov dx, offset uzenet1
10     mov ah, 09
11     int 21h
12
13     mov BL, 8
14 ;if (BL== 2) { goto egyenlo_printf }
15     cmp BL, 2
16     jz Egyenlo
17 ;else
18 ; {
19 Nem_egyenlo:
20     mov dx, offset uzenet3
21     mov ah, 09
22     int 21h              ;printf("Nem egyenlő")
23
24     jmp Program_vege     ; !!!!
25 ;}
26
27 Egyenlo:
28     mov dx, offset uzenet2
29     mov ah, 09
30     int 21h              ;printf("Egyenlő")
31
32 Program_Vege:
33     mov ax, 4c00h
34     int 21h
35
36 uzenet1  db "~(BL == 2)? $"
37 uzenet2  db "Egyenlo$"
38 uzenet3  db "Nem egyenlo$"
39
40 Code     Ends
```

```
5   Start:
6       mov ax, Code
7       mov DS, AX
8
9       mov dx, offset uzenet1
10      mov ah, 09
11      int 21h
12
13
14  ;if (BL== 2) { goto egyenlo_printf }
15      cmp BL, 2
16      jz Egyenlo
17  ;else
18  ; {
19  Nem_egyen  :
20      mov d   offset uzenet3
21      mov a   09
22      int 2                   ;printf("Nem egyenlő")
23
24      jmp P   gram_vege       ; !!!!
25  ;}
26
27  Egyenlo:
28      mov dx, offset uzenet2
29      mov ah, 09
30      int 21h                 ;printf("Egyenlő")
31
32  Program_Vege:
33      mov ax, 4c00h
34      int 21h
35
36  uzenet1   db "~(BL == 2)? $"
37  uzenet2   db "Egyenlo$"
38  uzenet3   db "Nem egyenlo$"
39
40  Code      Ends
```

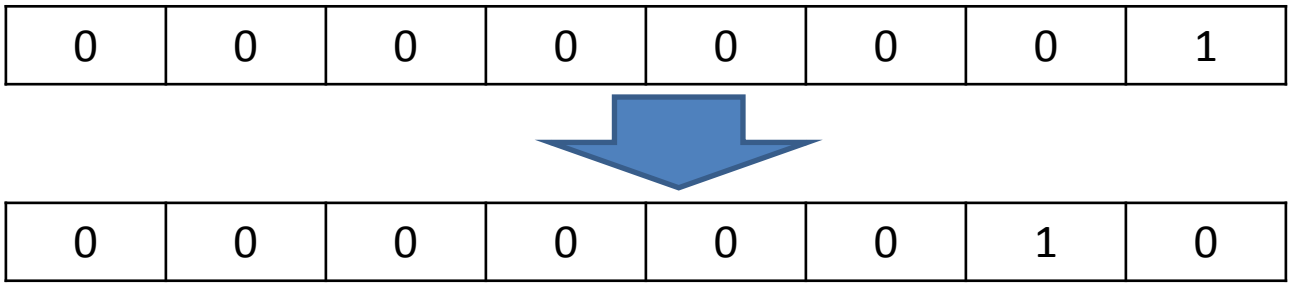
mov BL, 2

Bináris műveletek

- AND
  - csak akkor 1, ha mindkét bit értéke 1
  - maszkolásra lehet használni
- OR
  - csak akkor 0, ha mindkét bit értéke 0
- XOR
  - csak akkor 1, ha különböznek a bit értékei
  - komplemens képzésre lehet használni
- SHL
  - bitléptetés balra, szorozva 2-vel, belépő bit 0
- SHR
  - bitléptetés jobbra, osztva 2-vel, belépő bit 0
- ROR
  - bitléptetés jobbra, belépő bit a kilépő bit értéke
- ROL
  - bitléptetés balra, belépő bit a kilépő bit értéke

Bináris műveletek

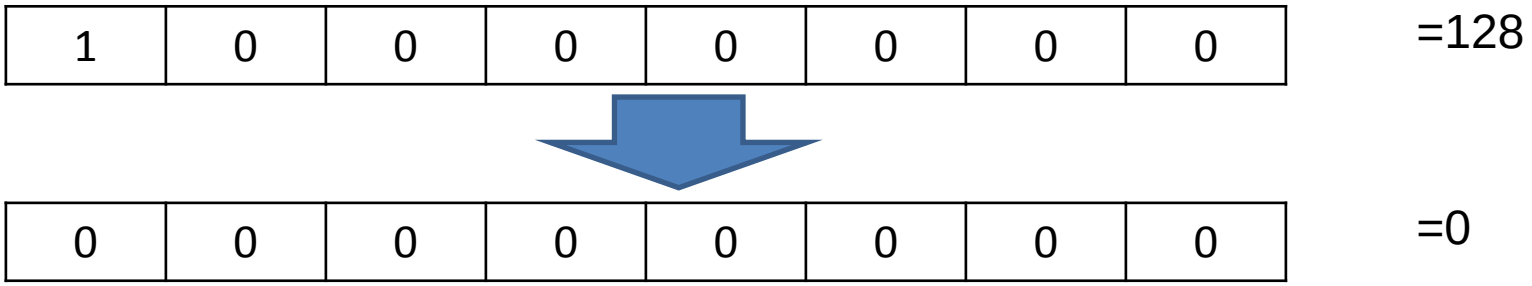
- SHL



Shift: belépő bit 0

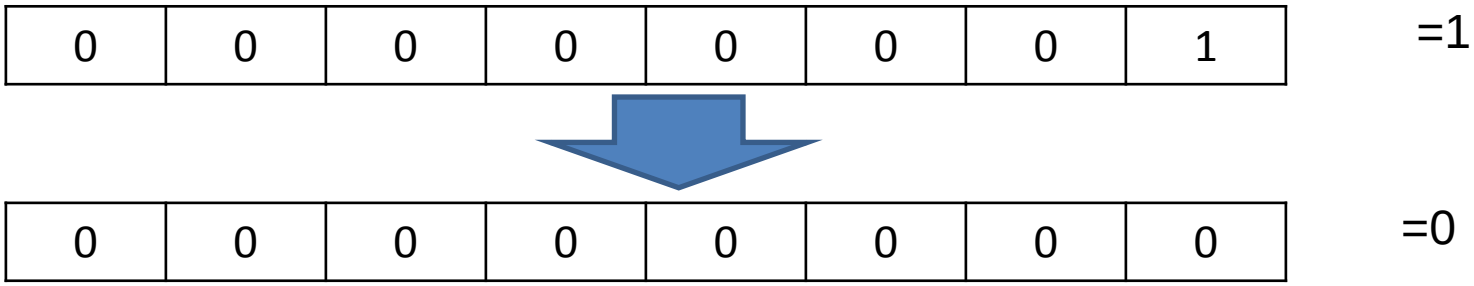
Bináris műveletek

- SHL



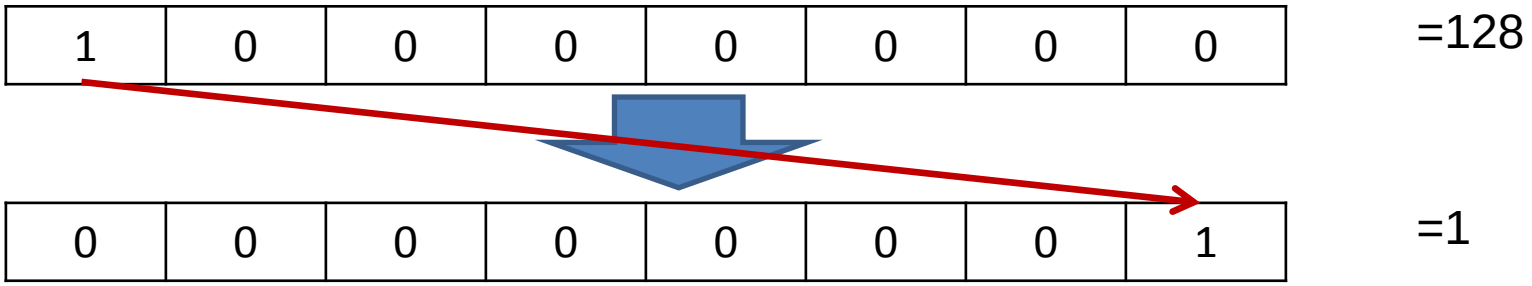
Bináris műveletek

- SHR



Bináris műveletek

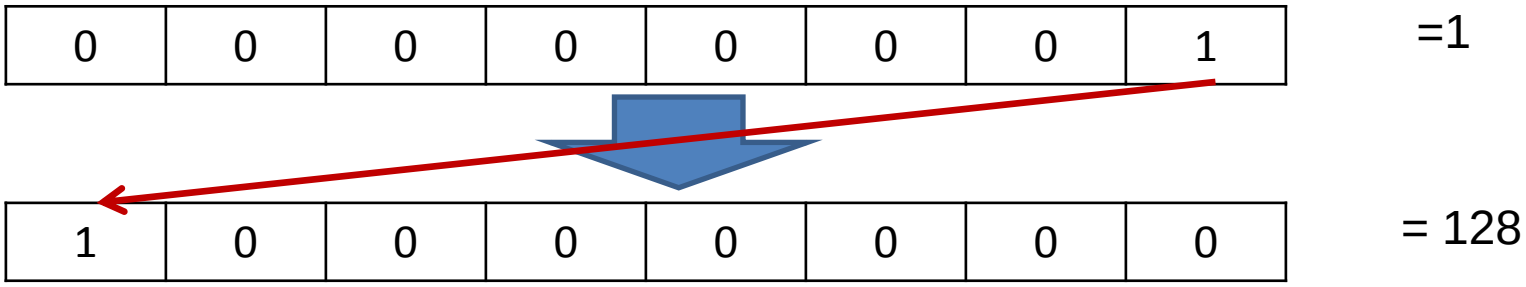
- ROL



- Rotate: belépő bit a kilépő bit

Bináris műveletek

- ROR



MOODLE

- http://nik.uni-obuda.hu/mobil/