

# Hardverközeli programozás 1

## 1. gyakorlat

Kocsis Gergely  
2019.02.11.

# Információk

---

**Kocsis Gergely**

<http://irh.inf.unideb.hu/user/kocsisg>

2 zh + 1 javító (a gyengébbikre)

A zh sikeres, ha az elért eredmény legalább 50%

Követelmény: Legalább 2 sikeres zh

**+ a két legjobb zh átlaga nagyobb vagy egyenlő, mint 60%**



# Információk

---

Maximálisan megengedett hiányzások száma: 3 alkalom

Maximálisan megengedett késés óráról: 20 perc

20 perc késés után a megjelenés engedélyezett, de hiányzásnak minősül

Zh-ról történő igazolatlan hiányzás esetén a zh eredménye 0%

Különösen indokolt esetben, egyéni elbírálás alapján, a zh előtt legalább egy héttel jelezve a zh-t lehetőség van az órán megbeszélttől más időpontban megírni



# Számrendszerek

Kettes számrendszer {0, 1}

Tízes számrendszer {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Tizenhatos (hexadecimális) számrendszer {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Tízes számrendszerbeli szám átváltása binárisra:

- A számot elosztom 2-vel. A maradékot leírom jobbra, az eredményt a szám alá.

- Ezt addig ismétlem, míg eredményül 0-t nem kapok.

- Lentől felfelé kiolvasom a számot

97		1
48		0
24		0
12		0
6		0
3		1
1		1
0		

97=%1100001

Visszaalakítás: A számjegyeket megszorozom a megfelelő helyiértékkel.

1	1	0	0	0	0	1	→	64+32+1=97
64	32	16	8	4	2	1		



# Számrendszerek

Tíz-es számrendszerbeli szám átváltása hexadecimálissá:

Használhatom a kettes számrendszerben alkalmazott módszert, de egyszerűbb először binárisra alakítani a számot, majd azt átváltani hexadecimálissá.

Bináris szám átváltása hexadecimálissá:

A számot 4 számjegyesével csoportosítjuk a legkisebb helyiértéktől indulva. Szükség esetén a szám elé plusz 0-kat írunk.

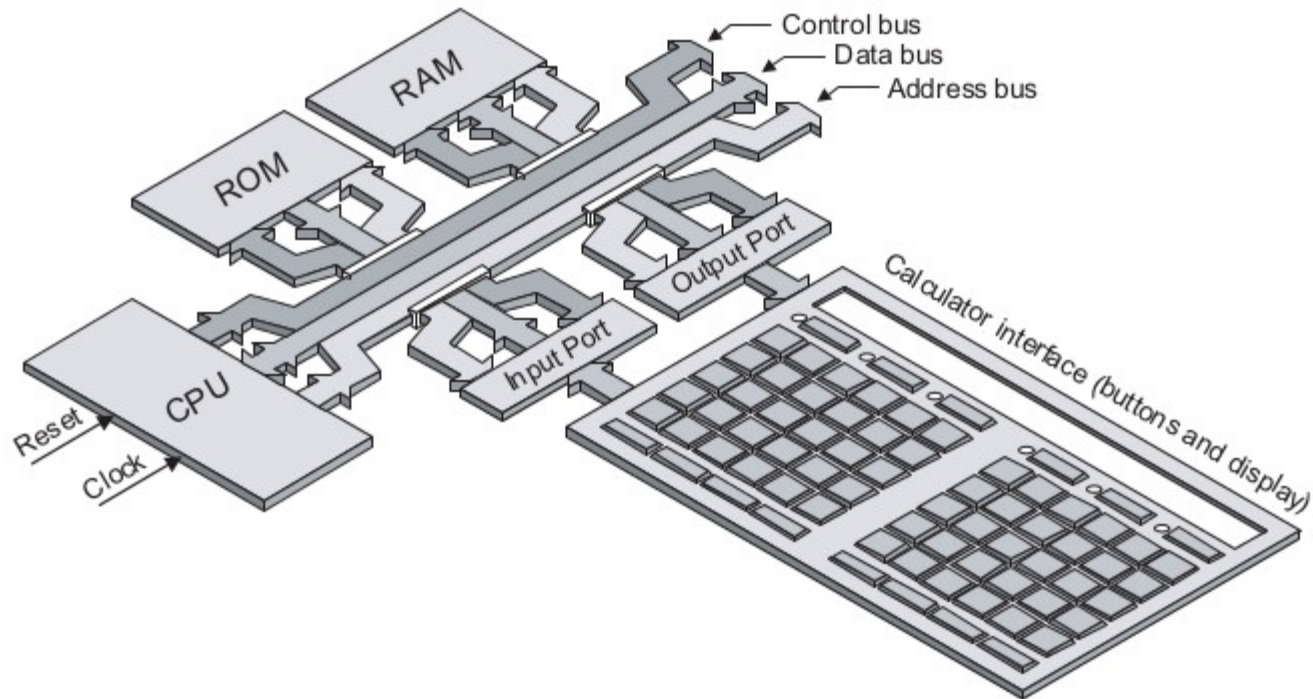
A csoportokat külön-külön hexadecimális számjeggyé váltjuk.

0 → 0000	4 → 0100	8 → 1000	C → 1100
1 → 0001	5 → 0101	9 → 1001	D → 1101
2 → 0010	6 → 0110	A → 1010	E → 1110
3 → 0011	7 → 0111	B → 1011	F → 1111

%1100001 → %0110 0001 → \$61



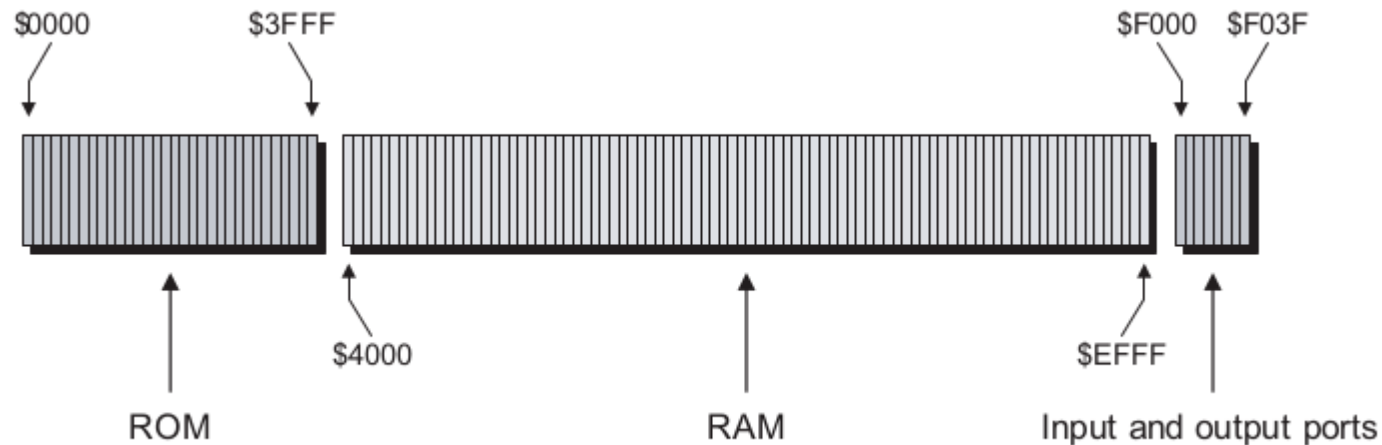
# A DIY Calculator



Egy valódi számológép esetében a ROM tárolná az inicializáló utasításokat, a billentyűk kezelését végző utasításokat, és a matematikai utasításokat is. Esetünkben ezeket, mi írjuk meg és a RAM-ba töltjük.



# A DIY Calculator memóriatérképe



16 bit → 0-65535 (\$0000-\$FFFF)

\$0000-\$3FFF: monitor program

\$4000-\$EFFF:

\$F000-\$F03F: I/O portok

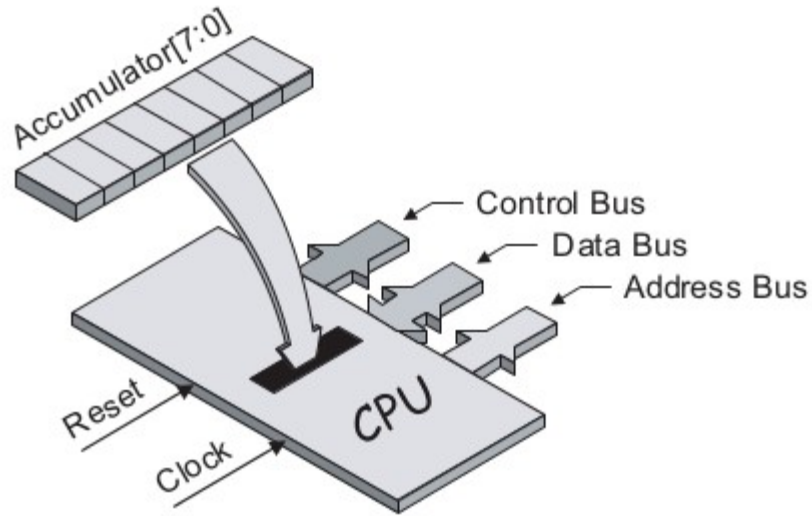
\$F000-\$F01F: Input portok      előlap: \$F011

\$F020-\$F03F: Output portok      előlap: \$F031, \$F032

\$F040-\$FFFF: használaton kívül



# A DIY Calculator → ACC



8 bites regisztertömb

Kapcsolódó funkciók:

- Adatok betöltése az ACC-be a memóriából
- Aritmetikai és logikai műveletek végzése a benne tárolt adatokon
- Adatok mentése egy memóriapozícióra (vagy portra)





# Az első DIY program

Készítsünk programot, ami bekapcsolás után letölri a számítógép képernyőjét.

1. \$10 betöltése a az ACC-be
2. Az ACC tartalmának kiírása az output portra (\$F03F)
3. Ugrás \$0000-ra → inicializáló állapot felvétele

A RAM kezdete

\$4000	\$90
\$4001	\$10
\$4002	\$99
\$4003	\$F0
\$4004	\$31
\$4005	\$C1
\$4006	\$00
\$4007	\$00

műveleti kódok

Töltsd az ACC-be a következő byte-ot  
\$10

Másold az ACC-t a következő memóriahelyre:  
\$F031

Ugorj ide:  
\$0000



# Az első DIY program

The screenshot shows a Windows desktop with a blue background. In the top-left corner, there is a 'DIYCalc' icon. The 'DIY Calculator' window is open, displaying a calculator interface with a menu bar (File, Setup, Display, Memory, Test, Help) and a dropdown menu with options: Assembler..., DIY Calculator on the WEB..., Terminal, and Workbench #1... The 'Assembler...' option is highlighted with a red rectangle. To the right, the 'Assembler Editor' window is open, showing assembly code in a text editor. The code is as follows:

```
$4000 # program kezdete
$10 # törlő kód beolvasása ACC-be
[$F031] # ACC kiírása az outputra
[$0000] # ugrás $0000-ra
```

The status bar at the bottom of the Assembler Editor window shows the file path: File:C:\DIY Calculator\Work\lab20.asm.

```
.ORG $4000 # A program a $4000 memóriacímen kezdődik
LDA $10 # Kerüljön az ACC-be a törlő op-kód
STA [$F031] # Másoljuk az ACC-t $F031 címre
JMP [$0000] # Ugrás ide: $0000
.END # Program vége
```



# Az első DIY program

The image shows two software windows side-by-side. On the left is the 'DIY Calculator' window, which has a menu bar (File, Setup, Display, Memory, Tools, Help) and a toolbar with icons for a person, gears, a green arrow, and a globe. Below the toolbar is a 'Calculator Interface' with a display area showing a dashed line and the text '3. RAM betöltése'. The calculator has a numeric keypad, function keys (Sin, Cos, Tan, Log, ln, x^y, x^2, Rx, 1/x), and mode buttons (Bin, Dec, Hex). At the bottom are buttons for 'On/Off', 'Reset', 'Step', 'Run', 'Clear', 'CE', 'Back', and 'Enter'. A red arrow points from the 'Load RAM..' button in the Memory menu to the display area. Another red arrow points from the 'Run' button to the text '4. Futtatás'. On the right is the 'Assembler Editor' window, which has a menu bar (File, Edit, Insert, Window, Help) and a toolbar with icons for file operations. The main text area contains assembly code: 

```
.ORG $4000 # program kezdete
LDA $10 # törlő kód beolvasása ACC-be
STA [$F031] # ACC kiírása az outputra
JMP [$0000] # ugrás $0000-ra
.END
```

 A red arrow points from the first line of code to the text '1. assembler futtatása'. Below the code is a status bar that says 'Cool Beans! Your file was assembled without error!'. A blue arrow points from this status bar to the text 'Csak akkor megyünk tovább, ha nem volt hiba'.

**3. RAM betöltése**

**4. Futtatás**

**2. számológép bekapcsolása**

**1. assembler futtatása**

Cool Beans! Your file was assembled without error!

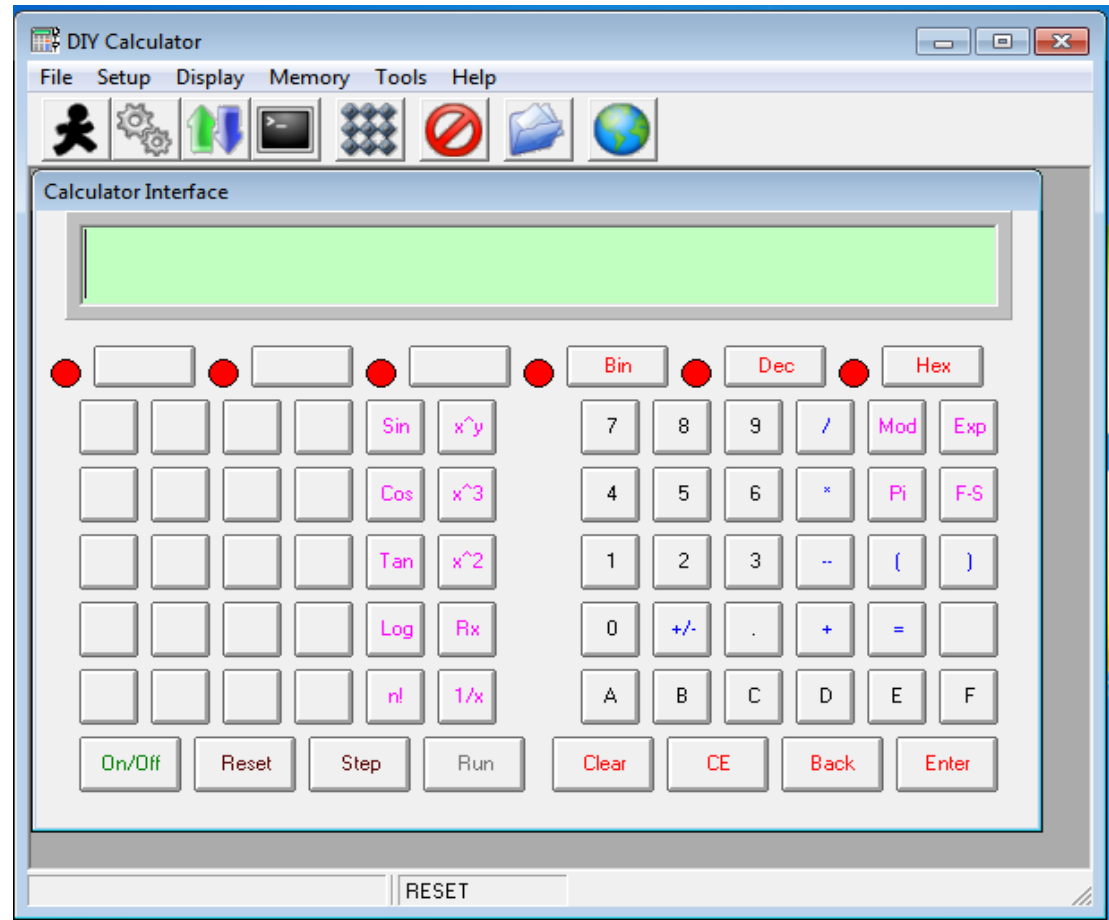
Csak akkor megyünk tovább, ha nem volt hiba

```
.ORG    $4000    # A program a $4000 memóriacímen kezdődik
LDA     $10      # Kerüljön az ACC-be a törlő op-kód
STA     [$F031] # Másoljuk az ACC-t $F031 címre
JMP     [$0000] # Ugrás ide: $0000
.END     # Program vége
```



# Az első DIY program

Ha minden rendben ment, akkor most ez van a képernyőn:  
(a kijelzőről eltűntek a ----- jelek)



# Az első DIY program

Az olvashatóság érdekében hozzunk létre címkeket a .EQU direktívával

```
CLRCODE: .EQU    $10      # A képernyőt törlő speciális kód
MAINDISP: .EQU    $F031    # A kijelző output portjának címe
          .ORG     $4000    # A program a $4000 memóriacímen kezdődik
          LDA     CLRCODE   # Kerüljön az ACC-be a törlő op-kód
          STA     [MAINDISP] # Másoljuk az ACC-t kijelző output portjára
          JMP     [$0000]   # Ugrás ide: $0000
          .END              # Program vége
```

Hajtsuk végre újra az előző folyamatot.  
(assembler, bekapcsolás, betöltés, futtatás)

Az előzővel azonos viselkedést  
kell kapnunk.

