Hardverközeli programozás 1
2. gyakorlat

Kocsis Gergely

2019.02.18.

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 binárissá:

- A számot elsoztom 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 megszorzom a megfelelő helyiértékkel.

```
 1   1   0   0   0   0   1
64  32  16   8   4   2   1
```

→ 64+32+1=97

Számrendszerek

Tízes számrendszerbeli szám átváltása hexadecimálissá:

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

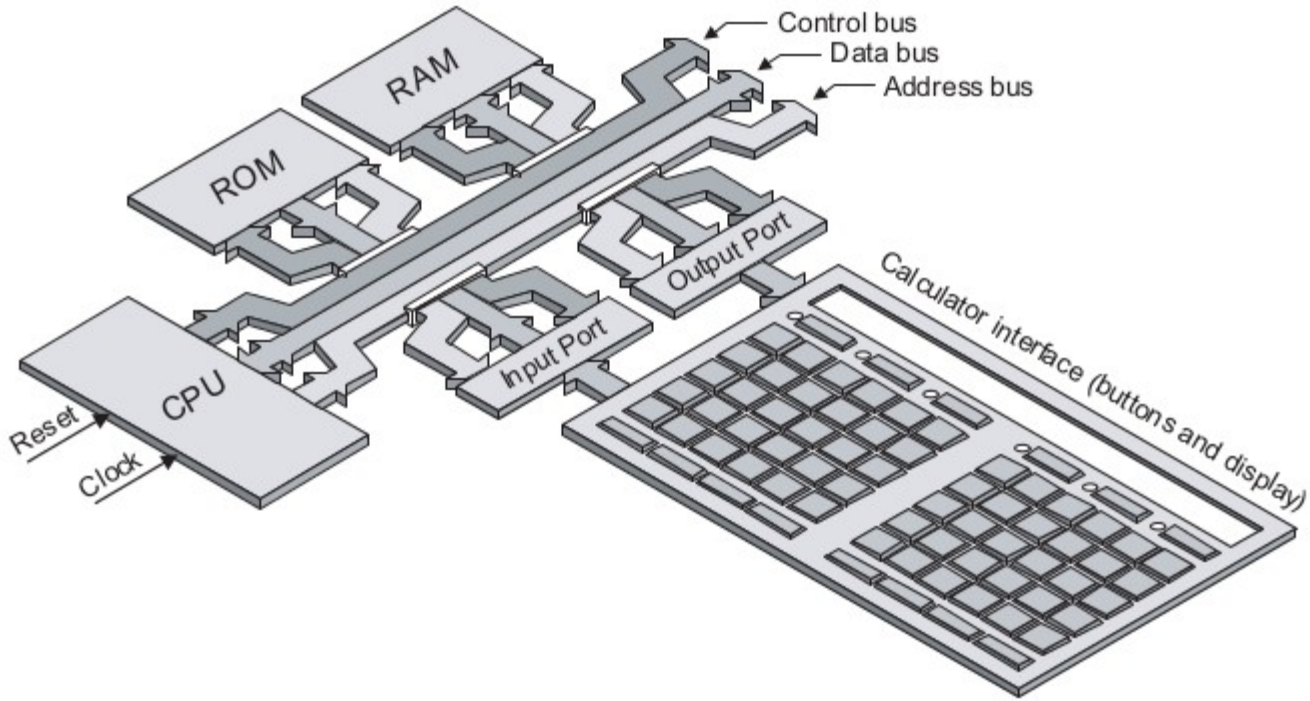
Bináris szám átváltása hexadecimálissá:

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

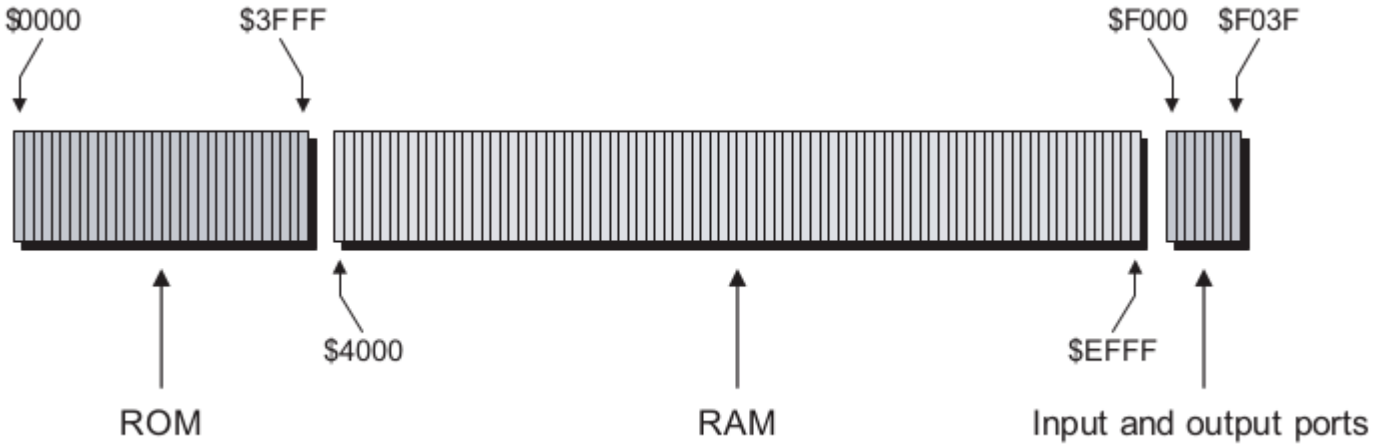
%1100001 ⟶ %0110 0001 ⟶ $61

A DIY Calculator



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

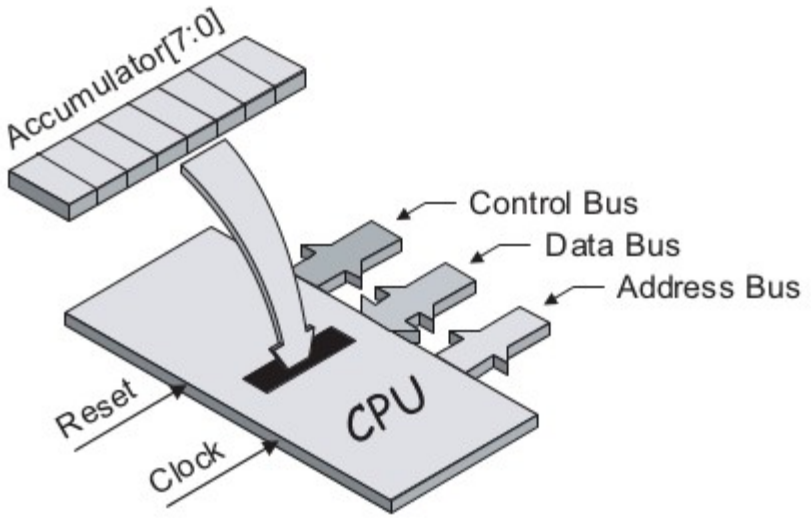
A DIY Calculator memóriatérképe



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

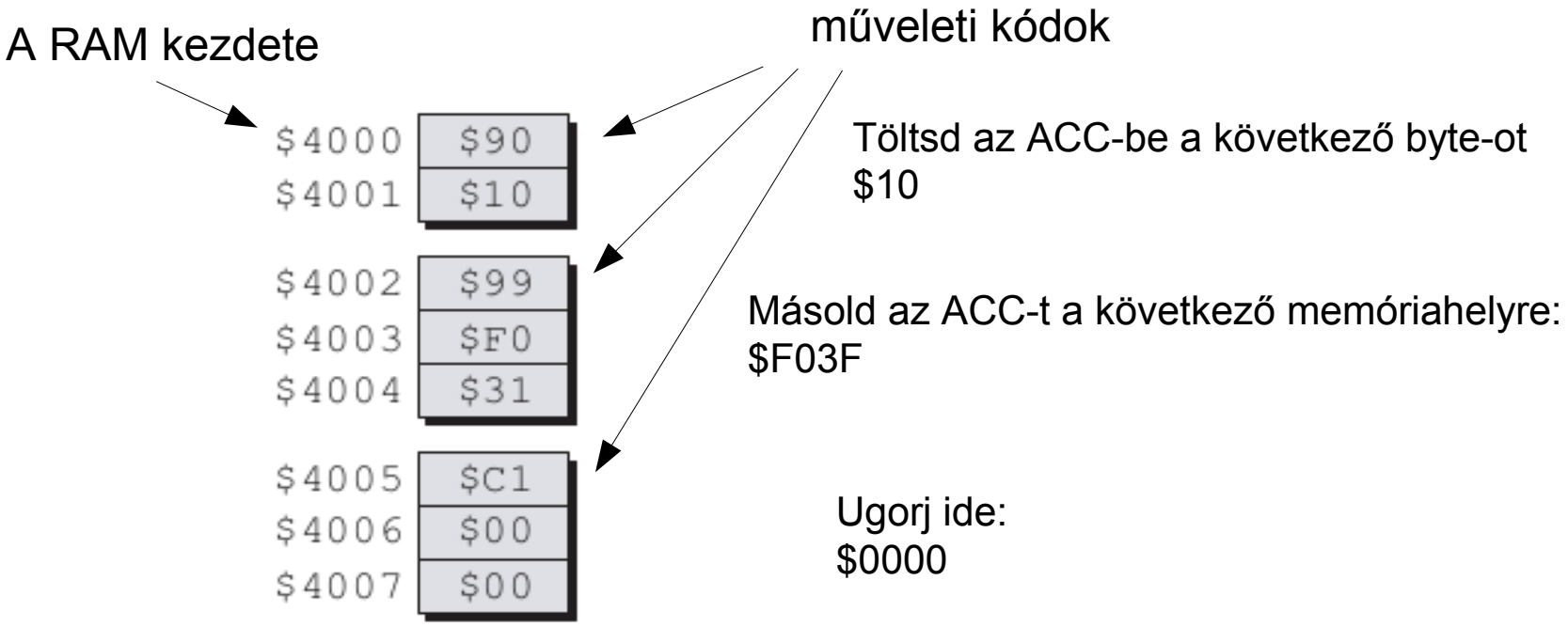


8 bites regisztertömb
Kapcsolódó funkciók:
- Adatok betöltése az ACC-be a memóriából
- Aritmetikai és logikai műveletek végzése a benne tárolt adatokon
- Adatok mentése egy memóriapozícióra (vagy portra)

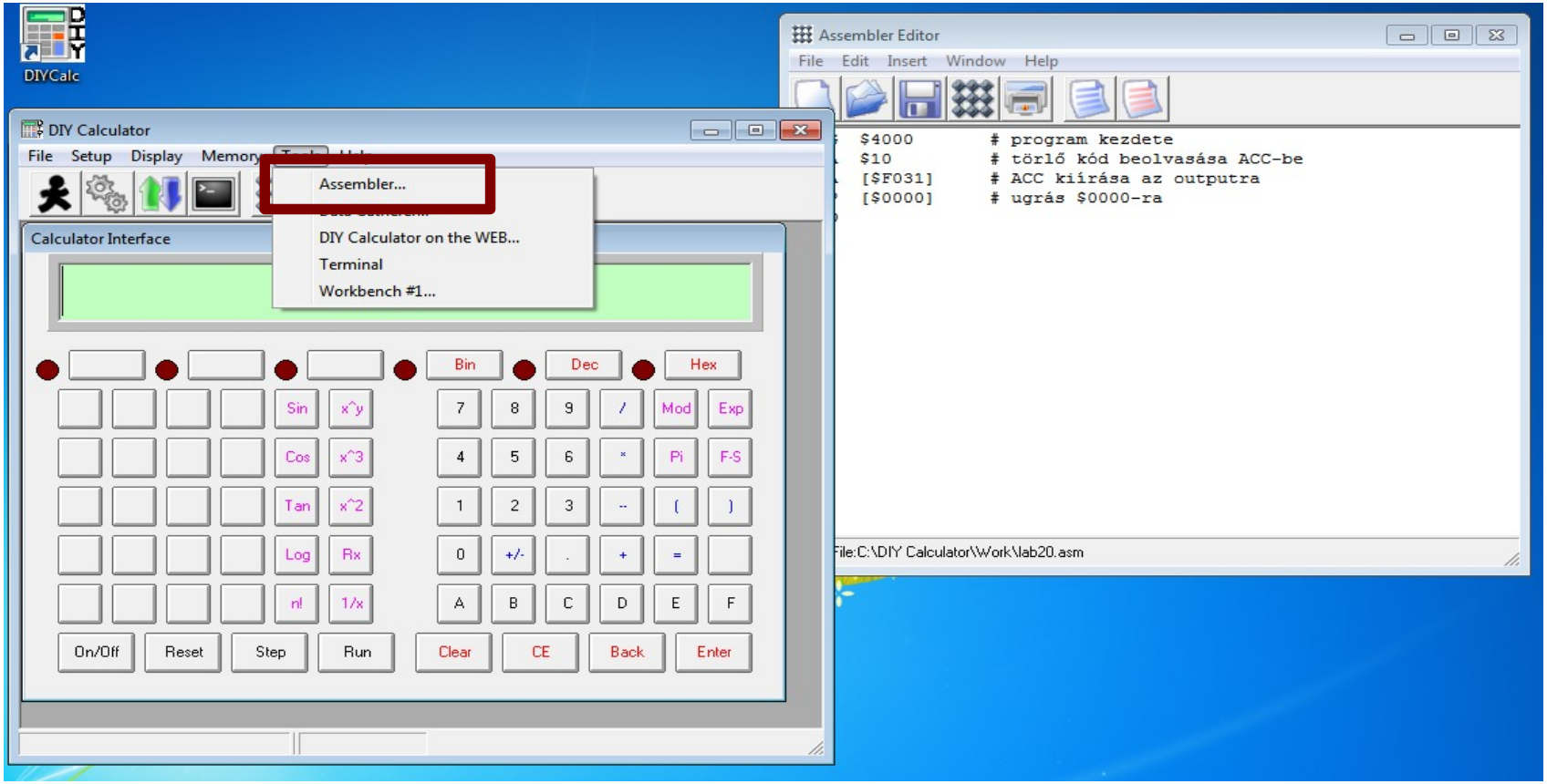
Az első DIY program

Készítsünk programot, ami bekapcsolás után letölri a számológép képernyőjét.

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

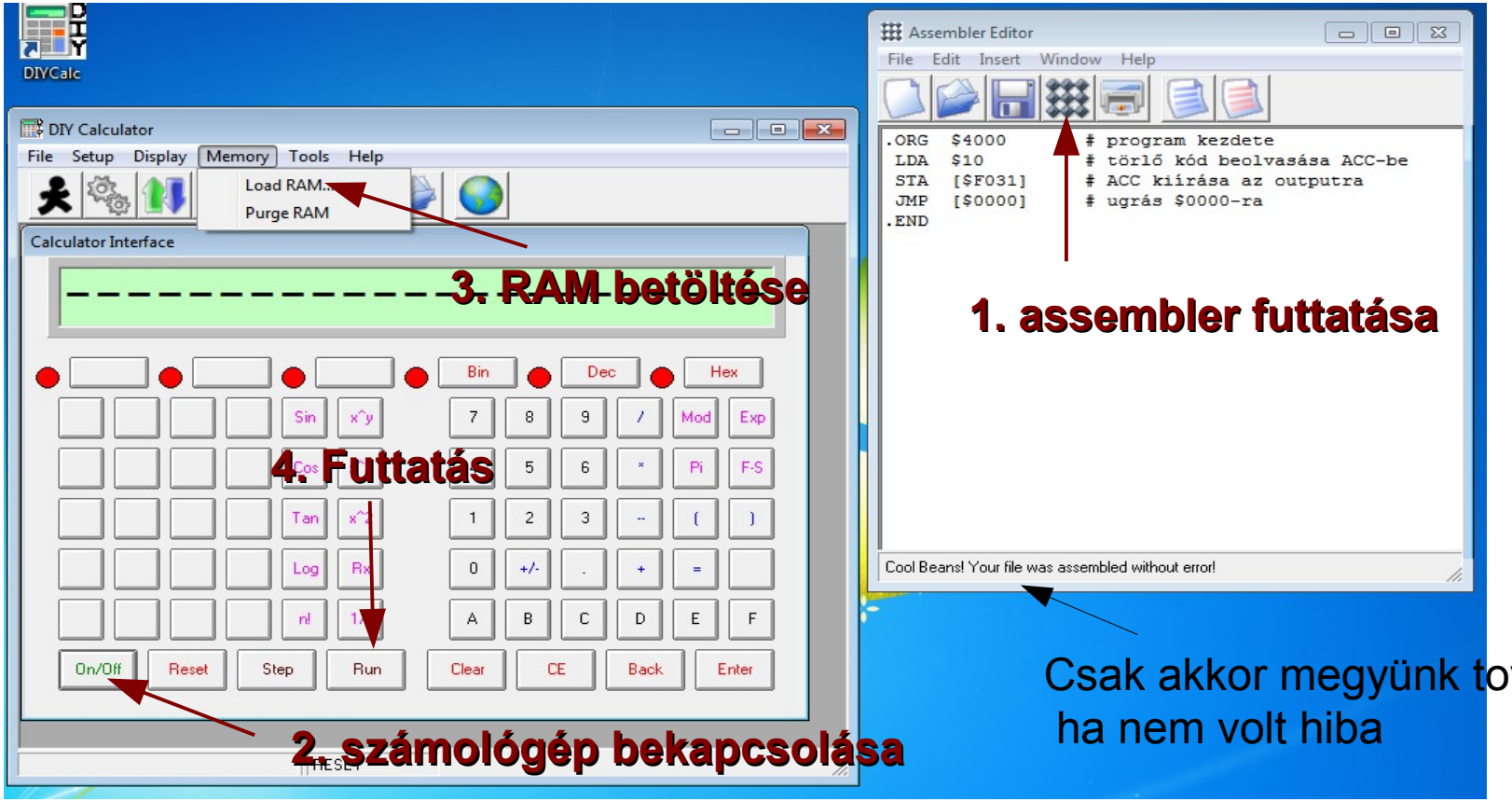


Az első DIY program




```
.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
```

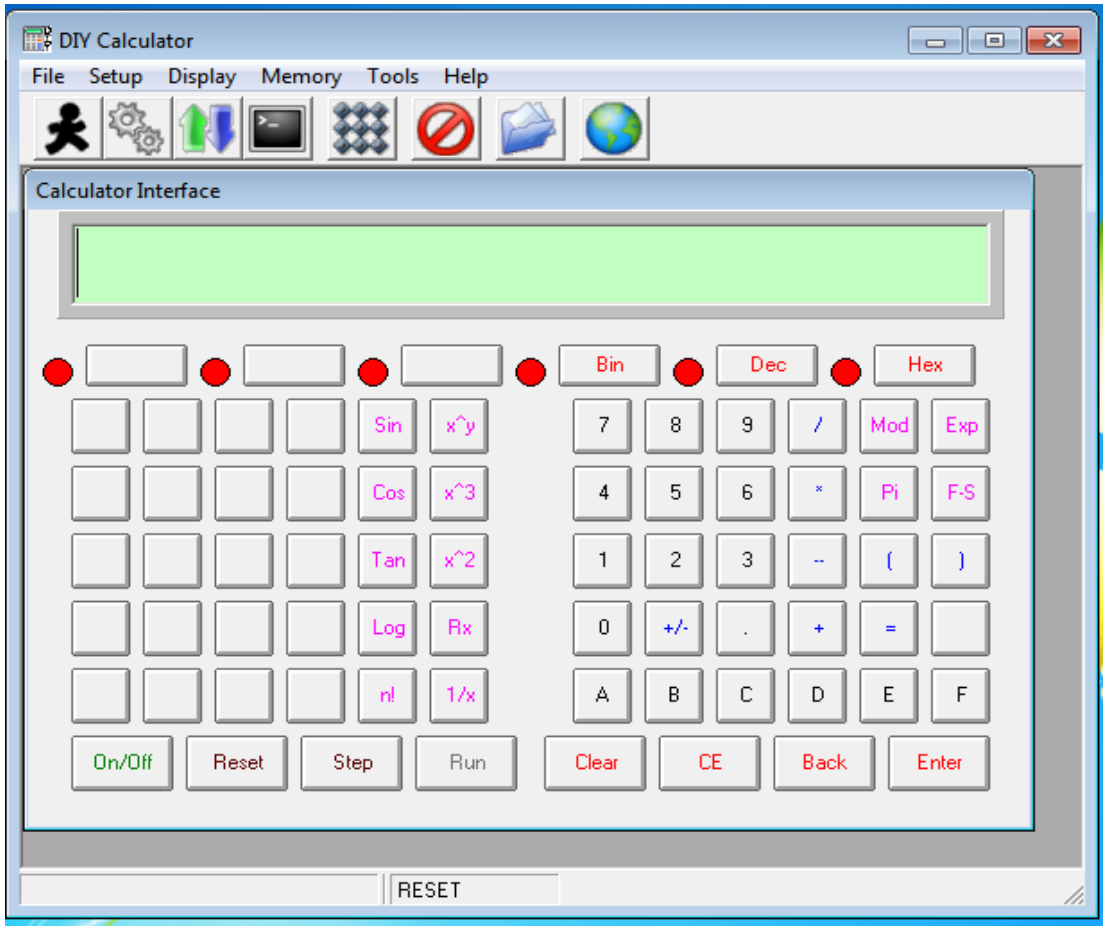
Az első DIY program




```
.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)



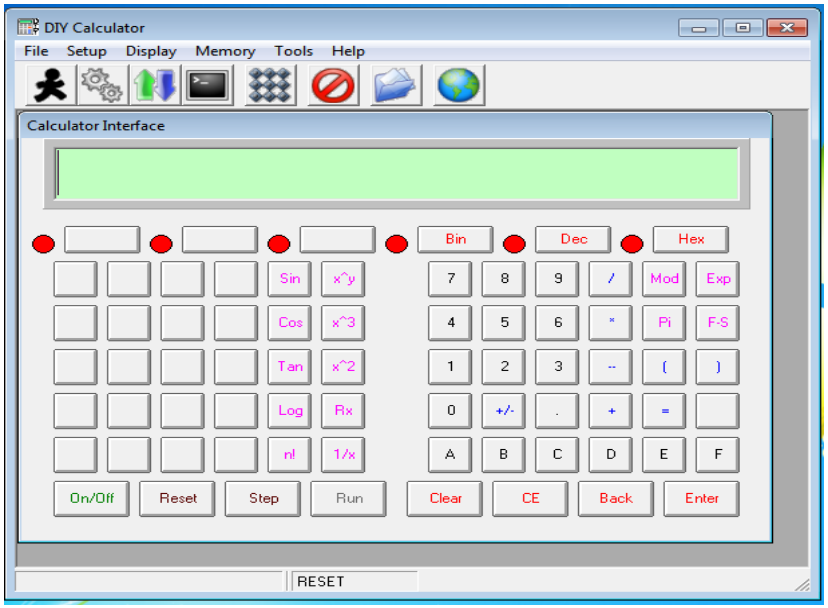
Az első DIY program

Az olvashatóság érdekében hozzunk létre cimkéket a .EQU direktívával

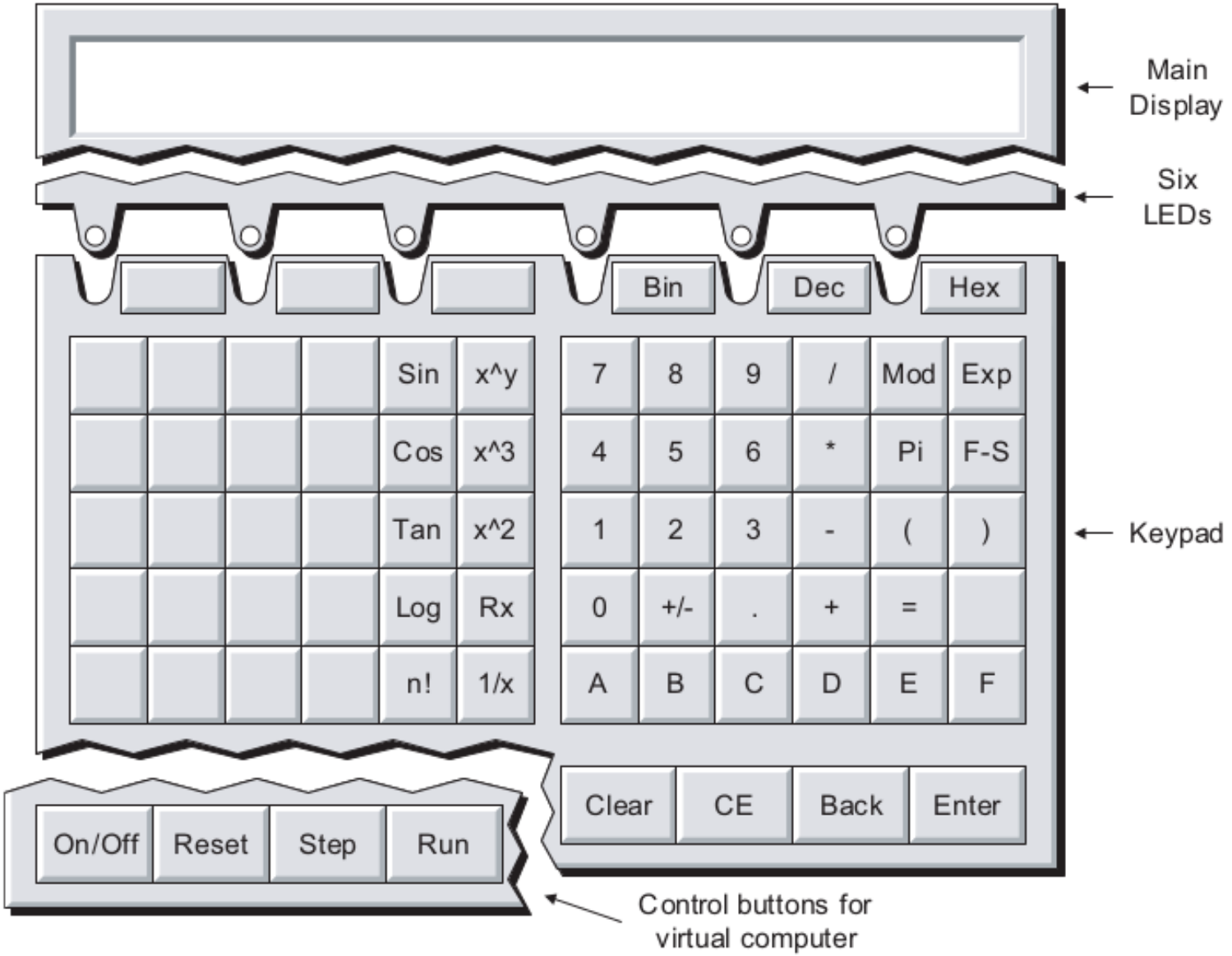
```
CLRCODE:   .EQU     $10          # A képernyőt törlő speciális kód
MAINDISP:  .EQU     $F031        # A kijelző output portjának címe
           .ORG     $4000        # A program a $4000 memóriacímen  kezdődik
             LDA    CLRCODE      # Kerüljön az ACC-be a törlő op-kód
             STA    [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.



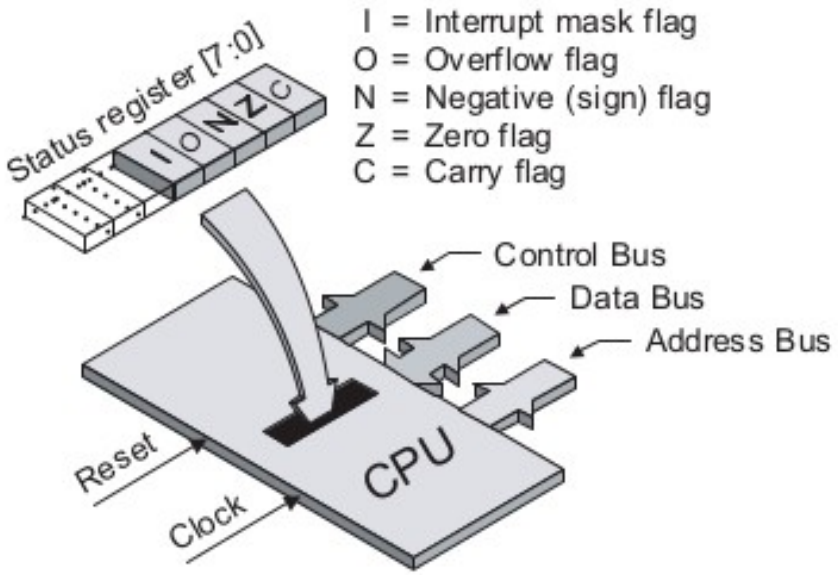
A számológép részei



A számológép kódtáblája

Not Standard ASCII				Standard ASCII											
$00	0	$10	Clr	$20	SP	$30	0	$40	@	$50	P	$60	`	$70	p
$01	1	$11	Bell	$21	!	$31	1	$41	A	$51	Q	$61	a	$71	q
$02	2	$12	Back	$22	"	$32	2	$42	B	$52	R	$62	b	$72	r
$03	3	$13		$23	#	$33	3	$43	C	$53	S	$63	c	$73	s
$04	4	$14		$24	$	$34	4	$44	D	$54	T	$64	d	$74	t
$05	5	$15		$25	%	$35	5	$45	E	$55	U	$65	e	$75	u
$06	6	$16		$26	&	$36	6	$46	F	$56	V	$66	f	$76	v
$07	7	$17		$27	'	$37	7	$47	G	$57	W	$67	g	$77	w
$08	8	$18		$28	(	$38	8	$48	H	$58	X	$68	h	$78	x
$09	9	$19		$29	)	$39	9	$49	I	$59	Y	$69	i	$79	y
$0A	A	$1A		$2A	*	$3A	:	$4A	J	$5A	Z	$6A	j	$7A	z
$0B	B	$1B		$2B	+	$3B	;	$4B	K	$5B	[	$6B	k	$7B	{
$0C	C	$1C		$2C	,	$3C	<	$4C	L	$5C	\	$6C	l	$7C	|
$0D	D	$1D		$2D	-	$3D	=	$4D	M	$5D	]	$6D	m	$7D	}
$0E	E	$1E		$2E	.	$3E	>	$4E	N	$5E	^	$6E	n	$7E	~
$0F	F	$1F		$2F	/	$3F	?	$4F	O	$5F	_	$6F	o	$7F	

A státuszregiszter (SR)



A Z flag értéke 1, ha az ACC értéke csupa 0
és 0, ha az ACC értéke nem csupa 0.
(Logikai értékkel válaszol arra a kérésre, hogy az ACC 0-e.)

N akkor 1, ha az ACC első bitje (MSB) szintén 1
C akkor 1, ha egy összehasonlítás után az ACC értéke nagyobb, mint az érték, amivel összehasonlítottuk (Pl.: `CMPA $09`)

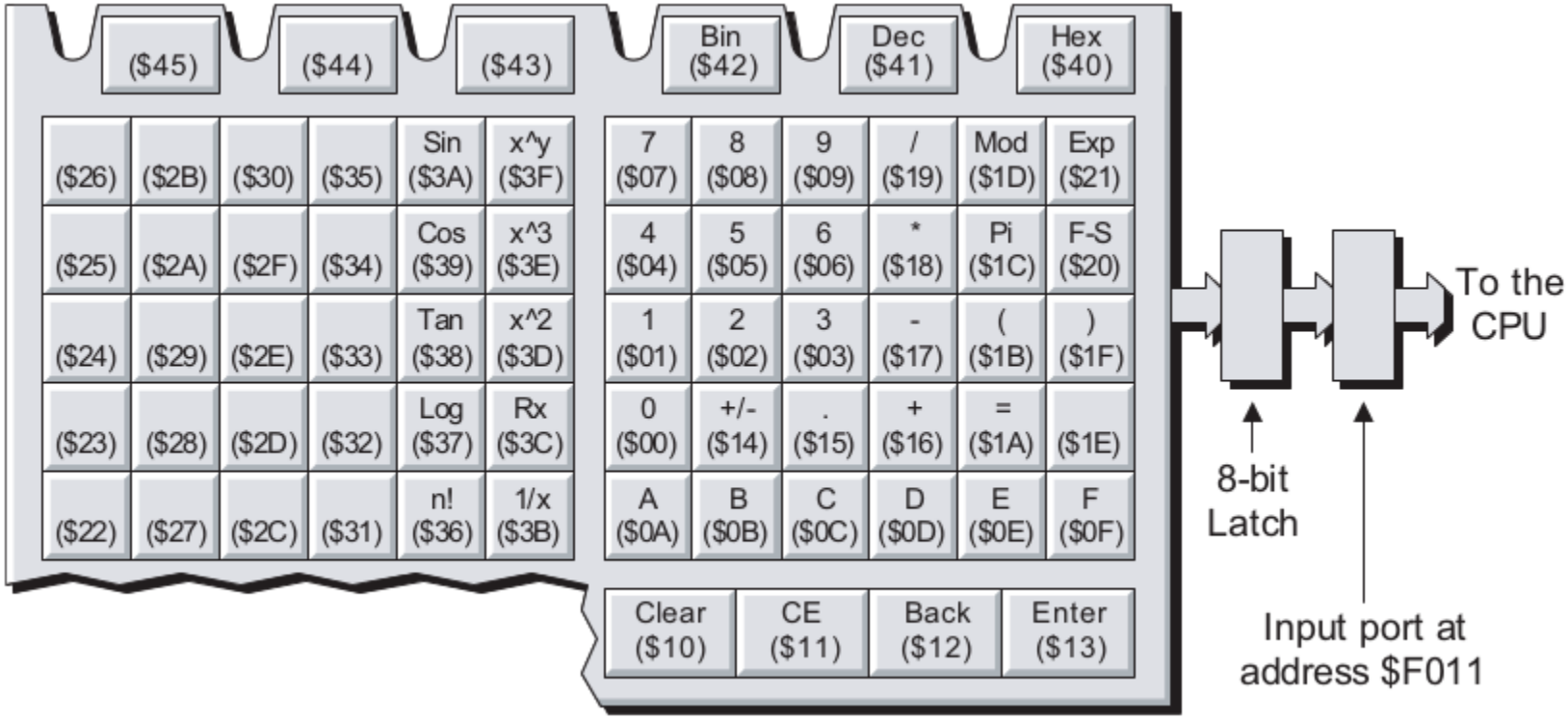
Írás a képernyőre

```
CLRCODE:        .EQU    $10             # képernyőtörlő kód
MAINDISP:       .EQU    $F031           # output port címe
                .ORG    $4000           # a program $4000-nél kezdődik
                 LDA    CLRCODE         # az ACC-be a törlő kód kerül
                 STA    [MAINDISP]      # ACC kiírása a kijelzőre
                 LDA    $09             # $09 beírása az ACC-be
LOOP:            STA    [MAINDISP]      # az ACC kiírása a kijelzőre
                 DECA                   # ACC csökkentése 1-gyel
                 JNZ    [LOOP]          # ugrás LOOP-ra, ha az ACC !0
                 JMP    [$0000]         # ugrás $0000-ra
                .END                    # program vége
```

DECA – ACC csökkentése 1-gyel
INCA – ACC növelése 1-gyel

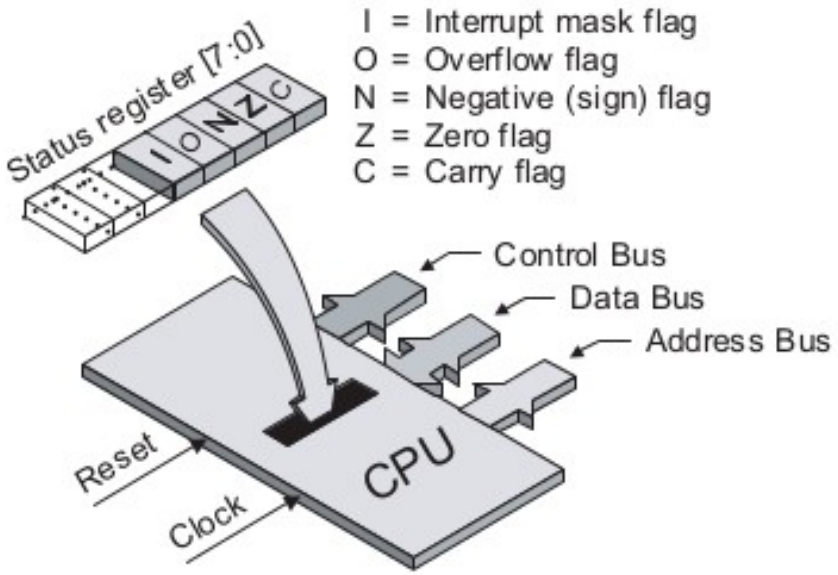
JNZ – ugrás, ha ACC nem 0
JZ – ugrás, ha ACC 0

Írás a képernyőre



Kód	Jelölés	Elvárt működés
$10	Clear	Törli az aktuális számítást és a képernyőt
$11	CE	Törli a képernyő aktuális értékét
$12	Back	Törli az utoljára beírt karaktert
$13	Enter	A beírt érték végét jelzi
$14	+/–	Előjelet vált

A státuszregiszter (SR)



A Z flag értéke 1, ha az ACC értéke csupa 0
és 0, ha az ACC értéke nem csupa 0.
(Logikai értékkel válaszol arra a kérésre, hogy az ACC 0-e.)

N akkor 1, ha az ACC első bitje (MSB) szintén 1
C akkor 1, ha egy összehasonlítás után az ACC értéke nagyobb, mint az érték, amivel összehasonlítottuk (Pl.: `CMPA $09`)

Írás a képernyőre

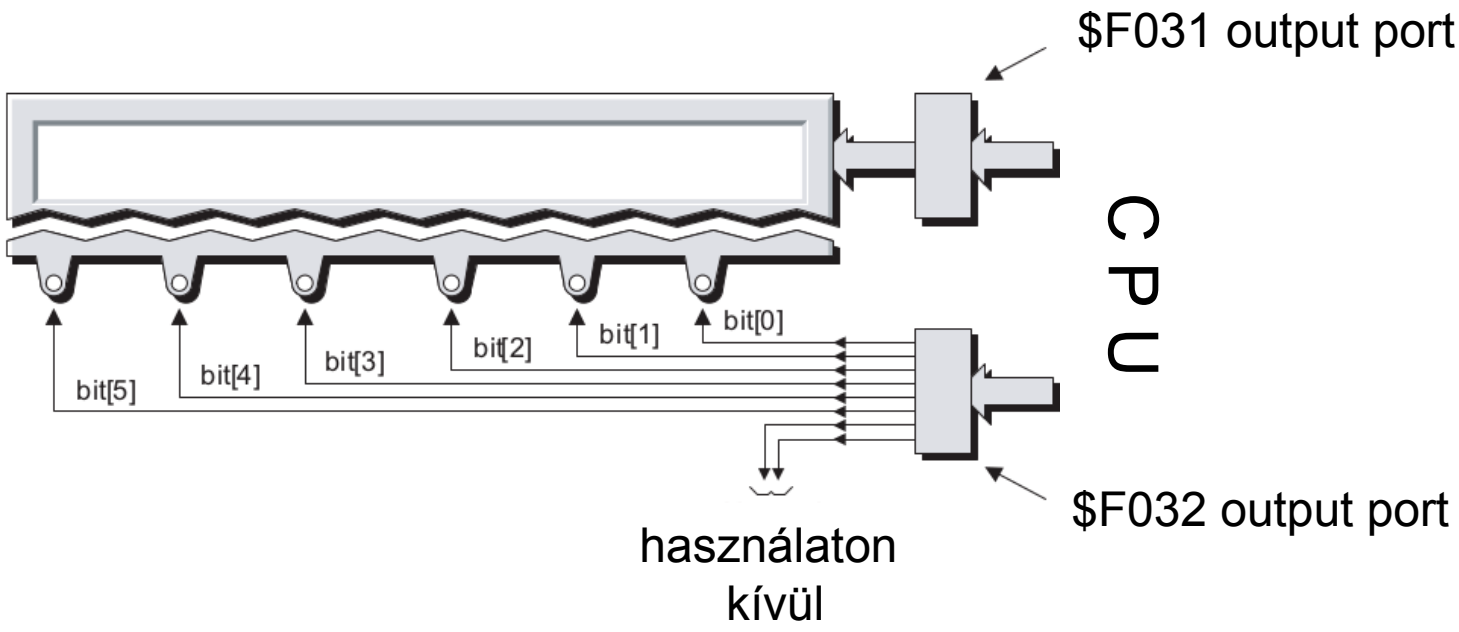
```
CLRCODE:        .EQU    $10             # képernyőtörlő kód
MAINDISP:       .EQU    $F031           # output port címe
KEYPAD:         .EQU    $F011           # input port címe
                .ORG    $4000           # a program $4000-nél kezdődik
                 LDA    CLRCODE         # az ACC-be a törlő kód kerül
                 STA    [MAINDISP]      # ACC kiírása a kijelzőre
LOOP:            LDA    [KEYPAD]        # az input beírása az ACC-be
                 JN     [LOOP]          # ugrás LOOP-ra, ha N==1
                 CMPA   $09             # az ACC nagyobb, mint $09?
                 JC     [LOOP]          # ha igen, ugrás LOOP-ra
                 STA    [MAINDISP]      # az ACC kiírása a kijelzőre
                 JMP    [LOOP]          # ugrás LOOP-ra
                 JMP    [$0000]         # ugrás $0000-ra
                .END                    # program vége
```

Ha nem nyomtunk gombot, várunk a gombnyomásra

Újabb gomb beolvasása

Ha a nyomott gomb nem egy számjegy

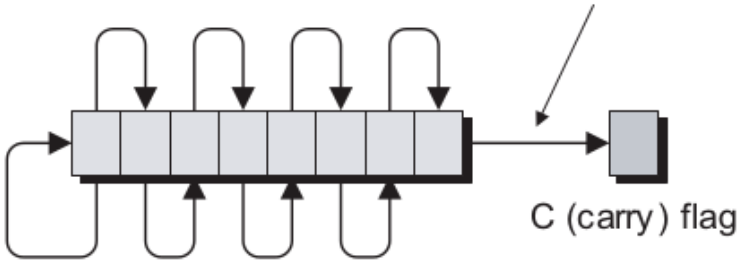
LEDek vezérlése



SHR:

Az MSB-t jobbra másoljuk,
önmaga viszont megtartja értékét

Az LSB értéke a C flagbe kerül



LEDek vezérlése

Készítsünk programot, mely körbefutó fényt mutat a ledeken balról jobbra haladva.

```
CLRCODE:   .EQU    $10          # képernyőtörlő kód
MAINDISP:  .EQU    $F031        # output port címe
SIXLEDS:   .EQU    $F032        # ledsor output port
           .ORG    $4000        # a program $4000-nél kezdődik
            LDA    CLRCODE      # az ACC-be a törlő kód kerül
            STA    [MAINDISP]   # ACC kiírása a kijelzőre
LOOPA:      LDA    $20          # o----- az akkuba
LOOPB:      STA    [SIXLEDS]    # o----- a ledekre
            SHR                 # o----- → -o---- jobbra shift
            JNZ    [LOOPB]      # ha nem ------ akkor LOOPB
            JMP    [LOOPA]      # amugy LOOPA (------ → o-----)
            JMP    [$0000]      # ugrás $0000-ra
           .END                 # program vége
```

Az első led világítson, a többi ne

Ha nincs jobbra led, akkor újra az első világítson

Világítson az eggyel jobbra lévő led