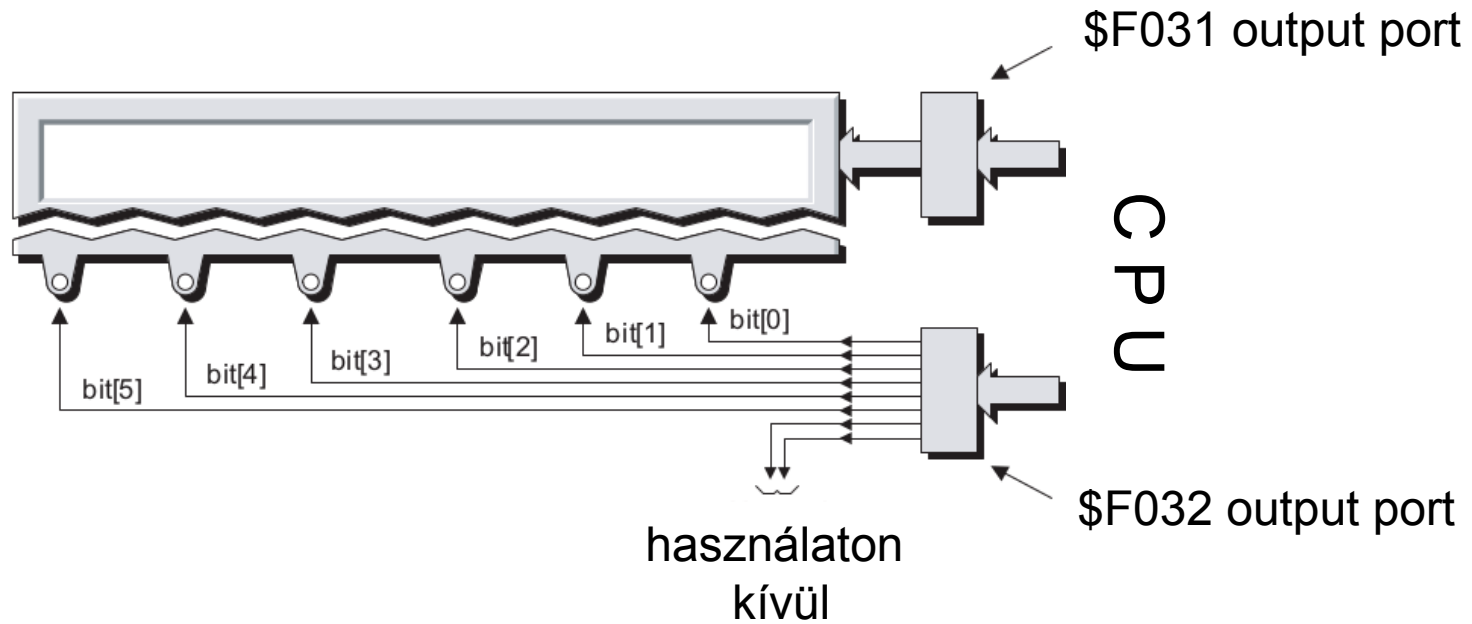


Hardverközeli programozás 1

3. gyakorlat

Kocsis Gergely
2019.02.25.

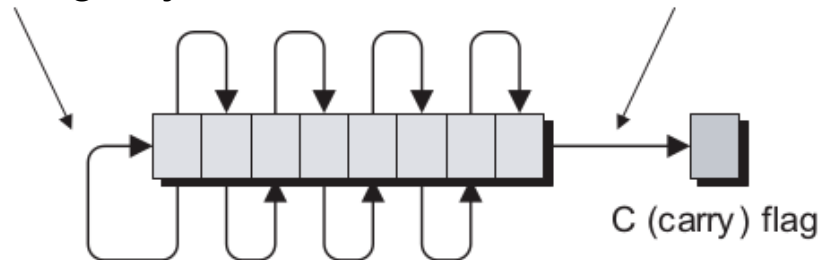
LEDek vezérlése



Az MSB-t jobbra másoljuk,
önmaga viszont megtartja értékét

Az LSB értéke a C flagbe kerül

SHR:



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          # 0----- az akkuba
LOOPB:   STA    [SIXLEDS]   # 0----- a ledekre
          SHR    # 0----- → -0---- jobbra shift
          JNZ    [LOOPB]      # ha nem ----- akkor LOOPB
          JMP    [LOOPA]      # amugy LOOPA (----- → 0-----)
          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



A rendszeróra állítása

The image shows a screenshot of the 'DIY Calculator' software interface. The 'Setup' menu is open, with 'System Clock...' selected. A red arrow points from this menu item to a dialog box titled 'Set the System Clock'. The dialog box has a 'Fast' radio button selected and a 'Slow' radio button. Below the radio buttons is a slider control with left and right arrows. There are 'Apply' and 'OK' buttons at the bottom right of the dialog box. The main calculator interface is visible in the background, featuring a green display area and a grid of buttons including mathematical functions (Sin, Cos, Tan, Log, nl, x^y, x^3, x^2, Rx, 1/x), basic arithmetic (7-9, /, *, -, +, =), and system controls (On/Off, Reset, Step, Run, Clear, CE, Back, Enter). The status bar at the bottom shows 'POWER_OFF'.



Mintaprogram megnyitása

The image shows a computer interface with two main windows. On the left is the 'DIY Calculator' application, which includes a 'Calculator Interface' with a green display and a keypad with buttons for mathematical functions (Sin, Cos, Tan, Log, nl) and control buttons (On/Off, Reset, Step, Run). Below the calculator is a 'POWER_OFF' indicator. On the right is a 'Megnyitás' (Open) file dialog window. The dialog shows the current directory as 'DIY Calculator > Work' with a search term 'Work'. The file list contains several .asm files:

Név	Módosítás dátuma	Típus
fmult.asm	2005.01.20. 20:17	ASM fájl
hello.asm	2005.08.01. 11:04	ASM fájl
lab20.asm	2012.02.09. 15:58	ASM fájl
skeleton.asm	2012.02.23. 11:45	ASM fájl
test.asm	2012.02.23. 11:54	ASM fájl

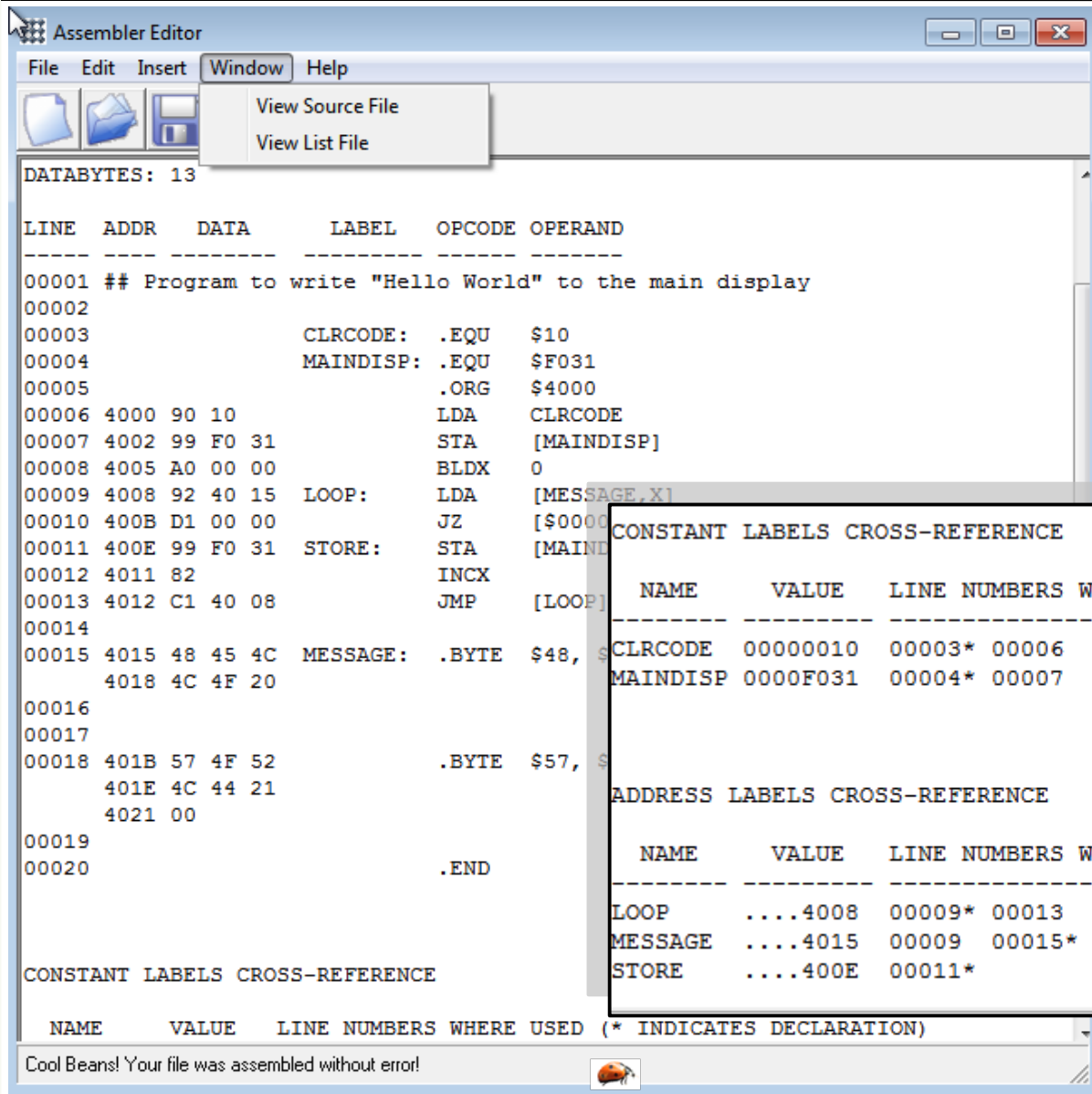
The 'hello.asm' file is selected. The 'Fájlnév:' field contains 'hello.asm' and the file type is set to 'Assembler Files'. The 'Megnyitás' (Open) button is highlighted. In the background, the 'Assembler Editor' window is visible, showing assembly code for a program that writes 'HELLO WORLD!' to the screen. The code includes comments and directives like CLRCODE, MAINDISP, LDA, STA, BLDX, JZ, INCX, JMP, and MESSAGE.

```
## Program to write "Hello World!"  
CLRCODE: .EQU    $10  
MAINDISP: .EQU  $F031  
          .ORG   $4000  
          LDA   CLRCODE  
          STA   [MAINDISP] ; Store CLRCODE  
          BLDX  0  
LOOP:    LDA   [MESSAGE] ; Load message  
          JZ   [$0000] ; If zero, skip  
          STA   [MAINDISP] ; Store message  
          INCX  
          JMP  [LOOP] ; Jump back to LOOP  
          # Jump back to LOOP  
MESSAGE: .BYTE $48, $45, $4C, $4C, $4F, $20  
          # H E L L O SPACE  
          .BYTE $57, $4F, $52, $4C, $44, $21, $00  
          # W O R L D ! NUL  
          .END
```

Open File: C:\DIY Calculator\Work\hello.asm



A listafájl megtekintése



The screenshot shows the Assembler Editor window with a menu open. The menu options are 'View Source File' and 'View List File'. The main window displays assembly code for a program that writes 'Hello World' to the main display. The code includes labels like CLRCODE, MAINDISP, LOOP, and MESSAGE. A status bar at the bottom indicates 'Cool Beans! Your file was assembled without error!'.

```
Assembler Editor
File Edit Insert Window Help
View Source File
View List File
DATABYTES: 13
LINE ADDR DATA LABEL OPCODE OPERAND
-----
00001 ## Program to write "Hello World" to the main display
00002
00003 CLRCODE: .EQU $10
00004 MAINDISP: .EQU $F031
00005 .ORG $4000
00006 4000 90 10 LDA CLRCODE
00007 4002 99 F0 31 STA [MAINDISP]
00008 4005 A0 00 00 BLDX 0
00009 4008 92 40 15 LOOP: LDA [MESSAGE_X1]
00010 400B D1 00 00 JZ [$0000]
00011 400E 99 F0 31 STORE: STA [MAINDISP]
00012 4011 82 INCX
00013 4012 C1 40 08 JMP [LOOP]
00014
00015 4015 48 45 4C MESSAGE: .BYTE $48, $
4018 4C 4F 20
00016
00017
00018 401B 57 4F 52 .BYTE $57, $
401E 4C 44 21
4021 00
00019
00020 .END
CONSTANT LABELS CROSS-REFERENCE
NAME VALUE LINE NUMBERS WHERE USED (* INDICATES DECLARATION)
-----
CLRCODE 00000010 00003* 00006
MAINDISP 0000F031 00004* 00007 00011
ADDRESS LABELS CROSS-REFERENCE
NAME VALUE LINE NUMBERS WHERE USED (* INDICATES DECLARATION)
-----
LOOP ....4008 00009* 00013
MESSAGE ....4015 00009 00015*
STORE ....400E 00011*
CONSTANT LABELS CROSS-REFERENCE
NAME VALUE LINE NUMBERS WHERE USED (* INDICATES DECLARATION)
Cool Beans! Your file was assembled without error!
```

Program ellenőrzése, hibakeresés

The image displays the 'DIY Calculator' software interface, which includes several debugging and utility windows. The main window has a menu bar with 'File', 'Setup', 'Display', 'Memory', 'Tools', and 'Help'. A dropdown menu is open under 'Display', showing options: 'Memory Walker...', 'CPU Registers...', 'I/O Ports...', and 'Message System...'. The main display area contains a calculator keypad with buttons for binary, decimal, and hexadecimal modes, and various mathematical functions.

Overlaid windows include:

- I/O Ports Display:** Shows data for addresses \$F031, \$F032, and \$F011. It has sections for 'O/P to Main Display', 'O/P to LED's', 'I/P from Buttons', and 'Old I/P from Button', along with 'Character etc' and 'Binary' sections.
- CPU Register Display:** Shows the state of various registers: Accumulator (\$48), Program Counter (\$400E), Instruction Reg (\$D1), Index Reg (\$0000), Interrupt Vector (\$XXXX), and Stack Pointer (\$XXXX). It also shows the Status Reg with bits I, O, N, Z, and C.
- Memory Walker:** A table showing memory addresses and data. The table has columns for BP, Step, Address, and Data.
- Breakpoint Manager:** Shows a set breakpoint at address \$400E with buttons for 'Set Break Point', 'Clear Break Point', and 'Clear All Break Points'.

BP	Step	Address	Data
		\$400D	\$0
BP	>>	\$400E	\$99
		\$400F	\$F0
		\$4010	\$31
		\$4011	\$82
		\$4012	\$C1
		\$4013	\$40
		\$4014	\$8
		\$4015	\$48
		\$4016	\$45
		\$4017	\$4C
		\$4018	\$4C
		\$4019	\$4F
		\$401A	\$20
		\$401B	\$57
		\$401C	\$4F
		\$401D	\$52
		\$401E	\$4C
		\$401F	\$44

Message System Display

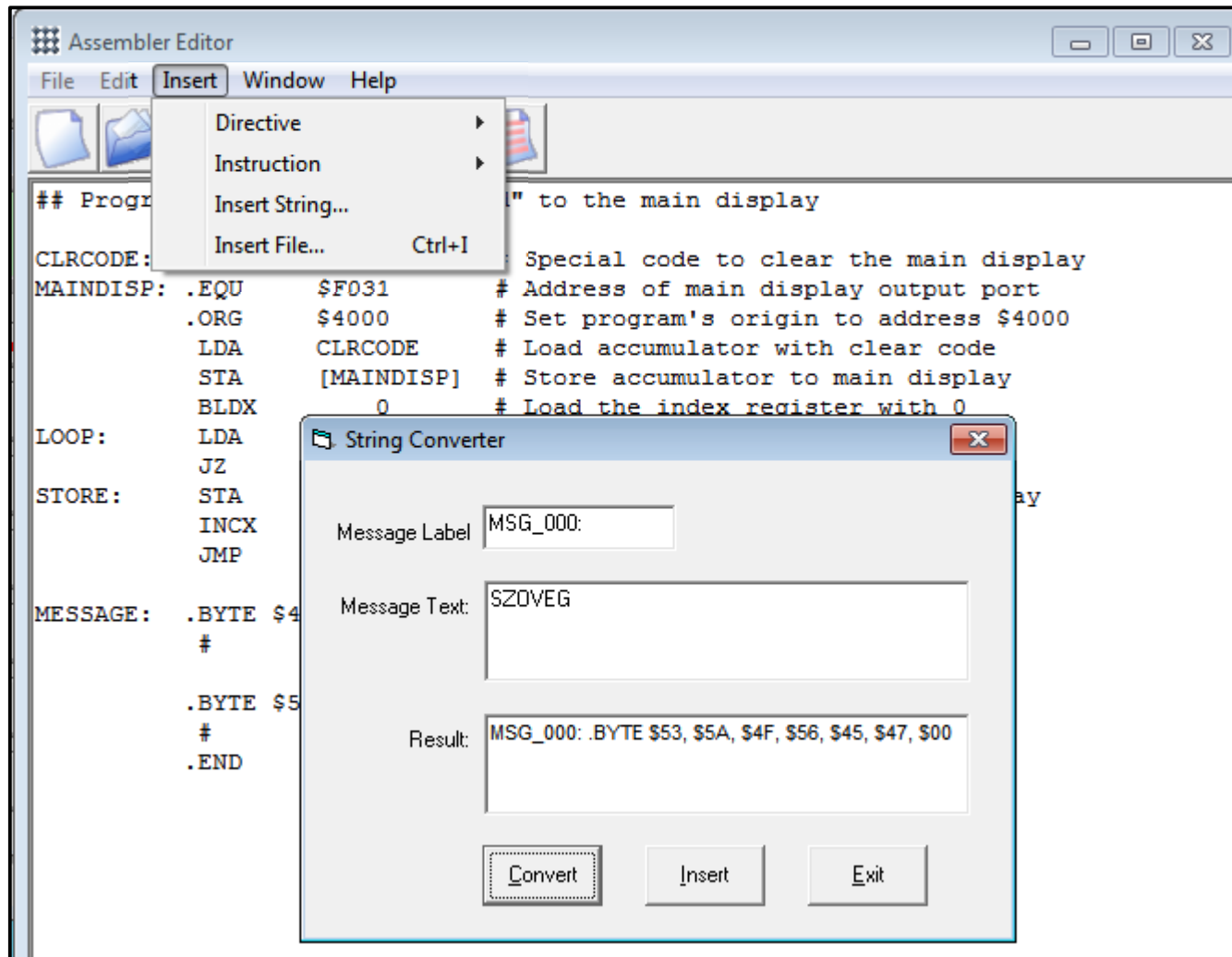
The image shows a software interface with two main windows. On the left is the 'Calculator Interface', which includes a large green display area, a keypad with various mathematical and logical functions, and a 'Step' button highlighted with a red box. Below the calculator are several control panels: 'O/P to Main Display' (showing \$F031 and \$10), 'O/P to LED's' (showing \$F032 and \$0), 'I/P from Buttons' (showing \$FF), 'Character etc' (with a 'Clear' button), 'Binary' (showing 000000), and 'Annotation' (showing A). On the right is the 'Message Window', which displays a sequence of system messages in a monospaced font. A red arrow points from the 'Step' button to the first line of the message window.

```
*****
Get Opcode
Program counter is $4000
Program counter -> address latch
  Latch points to target location
Increment program counter
  Doesn't change address latch
Read opcode
  Pointed to by address latch
Opcode = $90
Opcode -> instruction register
Decode Opcode
LDA (load accumulator) [imm]
  Next byte contains data
Get Data
Program counter is $4001
Program counter -> address latch
  Latch points to target location
Increment program counter
  Doesn't change address latch
Read data
  Pointed to by address latch
Data = $10
Execute opcode
Load data directly into the accumulator
Accumulator = $10
```

A programot léptetve minden egyes utasításról leírást kapunk a terminálban



Szöveg konstans beillesztése



The screenshot shows the Assembler Editor interface. The 'Insert' menu is open, showing options: Directive, Instruction, Insert String..., and Insert File... (Ctrl+I). The background code is as follows:

```
## Program to clear the main display
CLRCODE:      # Special code to clear the main display
MAINDISP:    .EQU    $F031    # Address of main display output port
             .ORG    $4000    # Set program's origin to address $4000
             LDA    CLRCODE    # Load accumulator with clear code
             STA    [MAINDISP] # Store accumulator to main display
             BLDX   0         # Load the index register with 0
LOOP:        LDA
             JZ
STORE:       STA
             INCX
             JMP
MESSAGE:     .BYTE $4
             #
             .BYTE $5
             #
             .END
```

The 'String Converter' dialog box is open, showing the following fields and buttons:

- Message Label: MSG_000:
- Message Text: SZOVEG
- Result: MSG_000: .BYTE \$53, \$5A, \$4F, \$56, \$45, \$47, \$00
- Buttons: Convert, Insert, Exit



Lenyomott billentyű kódjának kiírása

```
#####  
## Start of main program body ##  
#####  
# Gomb beolvasása  
GETKEY:    LDA        [KEYPAD]  
           JN         [GETKEY]  
           STA        [TEMP8]  
# A képernyő letörlése  
CLRDISP:   LDA        CLRCODE  
           STA        [MAINDISP]  
DISPDOLL:  LDA        $24      # $ jel kiírása a kijelzőre  
           STA        [MAINDISP]  
# az első 4 bit megjelenítése (a kód első hexa karaktere)  
DISPMSB:   LDA        [TEMP8]  
           SHR        # -XXXX---  
           SHR        # --XXXX--  
           SHR        # ---XXXX-  
           SHR        # ----XXXX  
           AND        %00001111  
           STA        [MAINDISP]  
# Az utolsó 4 bit megjelenítése  
DISPLSB:   LDA        [TEMP8]  
           AND        %00001111  
           STA        [MAINDISP]  
# vissza az elejére  
DONE:      JMP        [GETKEY]  
#####  
## End of main program body  
#####
```

```
#####  
## Start of global data  
#####  
TEMP8:     .BYTE      # 8 bites tároló pozíció  
#####  
## End of global data  
#####
```



Lenyomott billentyű kódjának kiírása II.

```
#####  
## Start of initialization                                     ##  
#####  
INIT:      LDA      CLRCODE      # Load accumulator with clear code  
           STA      [MAINDISP] # Write clear code to main display  
           LDA      BINMODE      # Load accumulator with BIN mode code  
           STA      [SIXLEDS]   # Write to port driving six LEDs  
#####  
## End of initialization                                     ##  
#####  
  
#####  
## Start of main program body                               ##  
#####  
# Gomb beolvasása  
GETKEY:    LDA      [KEYPAD]  
           JN       [GETKEY]  
           STA      [TEMP8]  
  
# A képernyő letörlése  
CLRDISP:   LDA      CLRCODE  
           STA      [MAINDISP]  
DISPPERC:  LDA      $25          # % jel kiírása a kijelzőre  
           STA      [MAINDISP]  
  
# az első 4 bit megjelenítése (a kód első hexa karaktere)  
DISPMSB:   LDA      [TEMP8]  
           SHR      # -XXXX--  
           SHR      # --XXXX--  
           SHR      # ---XXXX-  
           SHR      # ----XXXX  
           AND      %00001111  
           STA      [MAINDISP]  
  
# Az utolsó 4 bit megjelenítése  
DISPLSB:   LDA      [TEMP8]  
           AND      %00001111  
           STA      [MAINDISP]  
  
# vissza az elejére  
DONE:      JMP      [GETKEY]
```



Lenyomott billentyű kódjának kiírása II.

```
# A képernyő letörlése
CLRDISP:  LDA      CLRCODE
          STA      [MAINDISP]
DISPPERC: LDA      $25      # % jel kiírása a kijelzőre
          STA      [MAINDISP]
# A 7. bit kiírása
TEST7:   LDA      [TEMP8]
          SHL      # Az MSB a C-be kerül
          STA      [TEMP8]
          JC      [DISP7_1] # Ha c 1, akkor írjunk ki egy 1-est
DISP7_0: LDA      0        # Egyébként írjunk ki egy 0-t
          STA      [MAINDISP]
          JMP     [TEST6]
DISP7_1: LDA      1
          STA      [MAINDISP]
# A 6. bit kiírása
TEST6:   LDA      [TEMP8]
```

Az MSB (7. bit) esetére írt kódot kell ismételni egészen 0-ig (LSB)

```
DISP1_1:  LDA      1
          STA      [MAINDISP]
# A 0. bit kiírása
TEST0:   LDA      [TEMP8]
          SHL      # Az MSB a C-be kerül
          STA      [TEMP8]
          JC      [DISP0_1] # Ha c 1, akkor írjunk ki egy 1-est
DISP0_0: LDA      0        # Egyébként írjunk ki egy 0-t
          STA      [MAINDISP]
          JMP     [DONE]
DISP0_1: LDA      1
          STA      [MAINDISP]
          JMP     [DONE]
DONE:    JMP      [GETKEY]
#####
## End of main program body      ##
#####
```

