Hardverközeli programozás 1
4. gyakorlat

Kocsis Gergely

2019.03.04.

Lenyomott billentyűk megjelenítése visszafelé

```
            STA     [SIXLEDS]  # Write to port driving six LEDs
            BLDX    $0000      # index regiszter 0-ra állítása
##########################################################################
## End of initialization                                                ##
##########################################################################

##########################################
## Start of main program body
##########################################
# A képernyő letörlése
CLRDISP:    LDA         CLRCODE
            STA         [MAINDISP]
# Gomb beolvasása
GETKEY:     LDA         [KEYPAD]
            JN          [GETKEY]
            CMPA        $0F
            JC          [DISPSTUF]  # Ha az olvasott érték > $0F, megjelenít
            STA         [STORE,X]   # különben ACC a következő tömbhelyre
            INCX                    # index növelése
            JMP         [GETKEY]
# Tömb elemeinek megjelenítése
DISPSTUF:   LDA   [STORE-1,X]       # az utolsó érték menjen az ACC-be
            STA   [MAINDISP]
            DECX                    # index csökkentése
            JNZ   [DISPSTUF]        # ha nem 0-t írtunk ki, köv. irása
            JMP   [$0000]
##########################################################################
```

```
##########################################################
## Start of global data
##########################################################

TEMP8:      .BYTE             # 8 bites tároló pozíció
TEMP16:     .2BYTE            # 16 bites tároló
STORE:      .BYTE       *10   # 10*8 bites tárolótömb

##########################################################
## End of global data
##########################################################
```

Lenyomott billentyűk megjelenítése visszafelé

```
########## Verempointer inicializálás
           BLDSP   $EFFF          # Verem pointerbe $EFFF

########## Várakozás billentyűre

GETKEY:    NOP                    # “No operation”
LOOP:      LDA     [KEYPAD]       # Az input menjen az ACC-be
           JN      [LOOP]         # Ha nem volt gomb, vissza
           CMPA    $0F            # ACC >, mint $0F ?
           JC      [DISPSTUF]     # Ha igen, ugrás
           PUSHA                  # Amúgy ACC a verembe
           JMP     [GETKEY]       # Új gomb olvasása
DISPSTUF:  POPA                   # elem az ACC-be a veremből
           STA    [MAINDISP]      # ACC kiírása a kijelzőre
           JMP    [GETKEY]        # vissza az elejére
```

Gomb kódjának kiírása binárisan II

```
########## Get and display key codes in binary
## Wait for a key to be pressed
GETKEY:    LDA     [KEYPAD]    # Load ACC from the keypad
           JN      [GETKEY]    # Jump back if no key pressed
           STA     [TEMP8]     # Store ACC in temp location

## Clear display then display the '%' character
CLRDISP:   LDA     CLRCODE     # Load ACC with clear code
           STA     [MAINDISP]  # Clear main display
DISPPERC:  LDA     $25         # Load ACC with ASCII code for '%'
           STA     [MAINDISP]  # Write '%' to main display

## Initialize index register
LOADX:     BLDX    8           # Load X reg with 8

## Loop around writing
LOOP:      LDA     [TEMP8]     # Reload ACC from temp location
           SHL                 # Shift left 1 bit
           STA     [TEMP8]     # Store new value in temp location
           JC      [DISP_1]    # If carry = 1, jump to display a 1
DISP_0:    LDA     0           # ... otherwise load acc with 0
           STA     [MAINDISP]  # ... and store it to main display
           JMP     [DEALWX]    # ... then go and deal with the X reg
DISP_1:    LDA     1           # Load acc with 1
           STA     [MAINDISP]  # ... and store it to main display
DEALWX:    DECX                # Decrement the index register
           JNZ     [LOOP]      # If X not zero then do next bit
           JMP     [GETKEY]    # Go back and wait for new key
```