Hardverközeli programozás 1
5. gyakorlat

Kocsis Gergely

2019.03.11.

Index regiszter kiíratása

```
########## Display the value in the index register
DISPSTUF:  BSTX    [TEMP16]    # Store X reg into 2-byte temp location
           LDA     $24         # Load ACC with ASCII code for '$'
           STA     [MAINDISP]  # Write '$' to main display

########## Extract/display MS nybble of MS byte
DSMSNMSB:  LDA     [TEMP16]    # Load ACC with MS byte from X reg
           SHR                 # Shift right 1 bit (= 1 bit shift)
           SHR                 # Shift right 1 bit (= 2 bit shift)
           SHR                 # Shift right 1 bit (= 3 bit shift)
           SHR                 # Shift right 1 bit (= 4 bit shift)
           AND     %00001111   # Clear MS 4 bits
           STA     [MAINDISP]  # Copy result to main display

########## Extract/display LS nybble of MS byte
DSLSNMSB:  LDA     [TEMP16]    # Reload ACC with MS byte from X reg
           AND     %00001111   # Mask out (clear) MS nybble
           STA     [MAINDISP]  # Copy result to main display

########## Extract/display MS nybble of LS byte
DSMSNLSB:  LDA     [TEMP16+1]  # Load ACC with LS byte from X reg
           SHR                 # Shift right 1 bit (= 1 bit shift)
           SHR                 # Shift right 1 bit (= 2 bit shift)
           SHR                 # Shift right 1 bit (= 3 bit shift)
           SHR                 # Shift right 1 bit (= 4 bit shift)
           AND     %00001111   # Clear MS 4 bits
           STA     [MAINDISP]  # Copy result to main display
```

Lenyomott billentyűk megjelenítése visszafelé

```
            SHR                 # Shift right 1 bit (= 1 bit shift)
            SHR                 # Shift right 1 bit (= 2 bit shift)
            SHR                 # Shift right 1 bit (= 3 bit shift)
            SHR                 # Shift right 1 bit (= 4 bit shift)
            AND     %00001111   # Clear MS 4 bits
            STA     [MAINDISP]  # Copy result to main display

########## Extract/display LS nybble of MS byte
DSLSNMSB:   LDA     [TEMP16]    # Reload ACC with MS byte from X reg
            AND     %00001111   # Mask out (clear) MS nybble
            STA     [MAINDISP]  # Copy result to main display

########## Extract/display MS nybble of LS byte
DSMSNLSB:   LDA     [TEMP16+1]  # Load ACC with LS byte from X reg
            SHR                 # Shift right 1 bit (= 1 bit shift)
            SHR                 # Shift right 1 bit (= 2 bit shift)
            SHR                 # Shift right 1 bit (= 3 bit shift)
            SHR                 # Shift right 1 bit (= 4 bit shift)
            AND     %00001111   # Clear MS 4 bits
            STA     [MAINDISP]  # Copy result to main display

########## Extract/display LS nybble of LS byte
DSLSNLSB:   LDA     [TEMP16+1]  # Reload ACC with LS byte from X reg
            AND     %00001111   # Mask out (clear) MS nybble
            STA     [MAINDISP]  # Copy result to main display
            JMP     [$0000]     # Terminate the program
```

Szubrutinok használata

Írassuk ki a TEMP tömbben tárol 4 bájtot hexadecimális formában a kijelzőre. Az egyes bájtok megjelenítésére írjunk szubrutint.

```
#################################################
## Start of initialization ##
#################################################
INIT: LDA    CLRCODE     # törlő kód az ACC-be
      STA    [MAINDISP]  # Törlő kód a kijelzőre az ACC-ből
      LDA    HEXMODE     # HEXA mód kódja az ACC-be
      STA    [SIXLEDS]   # A LEDsor beállítása
      BLDSP  $EFFF       # Verempointer beállítása $EFFF -re
#################################################
## End of initialization   ##
#################################################

#################################
## Global data:
#################################
TEMP: .BYTE $23, $5A, $06, $4C
```

Szubrutinok használata

```
######## SZUBRUTIN KEZDETE

###### Szubrutin egy bájt megjelenítésére hexadecimálisan
DISPBYTE: PUSHA           # ACC mentése a verembe

###### A felső négy bit megjelenítése
DISPMSN: SHR              # Shift right 1 bit (= 1 bit shift)
         SHR              # Shift right 1 bit (= 2 bit shift)
         SHR              # Shift right 1 bit (= 3 bit shift)
         SHR              # Shift right 1 bit (= 4 bit shift)
         AND  %00001111   # A felső 4 bit nullázása
         STA  [MAINDISP]  # Kiíratás

###### Alsó négy bit kiírása
DISPLSN: POPA             # Az eredeti bájt kiolvasása
         AND  %00001111   # A felső 4 bit nullázása
         STA  [MAINDISP]  # Kiíratás
         RTS              # Visszatérés a szubrutinból

######## SZUBRUTIN VÉGE
```

Szubrutinok használata

```
###### Start Program Body - Főprogram
    LDA   [TEMP]     # Az első bájt beolvasása
    JSR   [DISPBYTE] # Az első bájt kiírása
J1: LDA   [TEMP+1]   # A második bájt beolvasása
    JSR   [DISPBYTE] # A második bájt kiírása
J2: LDA   [TEMP+2]   # A harmadik bájt beolvasása
    JSR   [DISPBYTE] # A harmadik bájt kiírása
J3: LDA   [TEMP+3]   # A negyedik bájt beolvasása
    JSR   [DISPBYTE] # A negyedik bájt kiírása
    JMP   [$0000]    # Vége
###### End Program body
```

A JSR az ugráson kivül a verembe teszi a JSR után következő utasitás LS, majd MS bájtját. Az RTS a verem tetejéről olvasott két bájtot értelmezi visszatérési cimként MSB,LSB sorrendben.

Szubrutinok használata

Cseréljük ki a szubrutint, több szinten ágyazott szubrutinnal

```
########## Szubrutin egy bájt megjelenítésére
DISPBYTE:  PUSHA            # ACC a verembe
           JSR [DISPMSN]    # Felső 4 bit megjelenítése
           POPA             # Eredeti bájt ACC-be a veremből
           JSR [DISPLSN]    # Alsó 4 bit megjelenítése
           RTS              # Visszatérés a hívóhoz
########## Szubrutin felső 4 bit megjelenítésére
DISPMSN:   SHR              # Shift jobba (1)
           SHR              # Shift jobba (2)
           SHR              # Shift jobba (3)
           SHR              # Shift jobba (4)
           AND %00001111    # MS 4B kinullázása
           STA [MAINDISP]   # Eredmény a kijelzőre
           RTS              # Visszatérés
########## Szubrutin alsó 4 bit megjelenítésére
DISPLSN:   AND %00001111    # Felső 4 bit maszkolása
           STA [MAINDISP]   # Eredmény a kijelzőre
           RTS              # visszatérés
```

Rekurzív szubrutinok használata

A rekurzív program törzse:

```
        BLDX   0         # Az index regiszter legyen 0
        BLDSP  $EFFF     # A verempointer mutasson a RAM végére
OUT:    JSR    [REV]     # Ugrás a szubrutinra
VEGE:   JMP    [$0000]   # Befejezés
```

A szubrutin:

```
########## Recursive subroutine
REV:            LDA    [PHRASE,X] # A következő karakter az ACC-be
                JNZ    [GOIN]     # Ha a karakter nem NUL, beljebb
RETURN1:        RTS               # Visszatérés a szubrutinból
########## Store the character on the stack and go further in
GOIN:           PUSHA             # A karakter a verembe
                INCX              # Az indexregiszter növelése
INSIDE:         JSR    [REV]      # Rekurzív hívás
########## Retrieve and display
COMEOUT:        POPA              # Karakter a veremből ki
                STA    [MAINDISP] # A karakter megjelenítése
RETURN2:        RTS               # Visszatérés a szubrutinból
```