Hardverközeli Programozás 1
MINTA zárthelyi dolgozat (2015) **MINTA**

Név: _____ Neptun: _____

1.) Végezd el a következő számításokat számológép használata nélkül és add meg az eredményt **hexadecimális** formában. Az alkalmazott számrendszerek jelölésére a feladatokban az órán is használt jelölésrendszer érvényes. A megoldások során maximum 8 biten ábrázolható számokat feltételezz. A műveletek működésénél a DYC működési elvét feltételezd. (2 pont).

$3A + %10011001 = 3*16 + 10 + 9*16 + 9 = 12*16 + 19 = 12*16 + 16 + 3 = 13*16 + 3 = **$D3**

SHL($5C) = SHL(%0101 1100) = %1011 1000 = **$B8**

$A5 - 65 = $A0-60 = 160 – 60 = 100 = 6*16 + 4 =**$64**

SHR(2 * $93) = SHR(SHL(%1001 0011)) #8 biten kell elvégezni!!!

SHR(%0010 0110) = %0001 0011 = **$13**

2.) Mi lesz az **akkumulátor** tartalma a következő DIY Calculator Asembly kódrészlet megjelölt pontjain (4 pont)?

```
            BLDX    $0001          X: 0001
            BSTX    [TEMP16]       X: 0001 TEMP16: $00 T.+1: $01
            LDA     [TEMP16,X]     # % 0 0 0 0 0 0 0 1
            SHL                      % 0 0 0 0 0 0 1 0
            ADD     126              % 1 0 0 0 0 0 0 0
            JNN     [TOHERE]           → nem ugrik, mert N=1
            DECA                     % 0 1 1 1 1 1 1 1
TOHERE:     DECA                     % 0 1 1 1 1 1 1 0
            PUSHA                  # % 0 1 1 1 1 1 1 0 + Verembe
            LDA     $AE              % 1 0 1 0 1 1 1 0
            JN      [TOHERE2]          → ugrik, mert N=1
            INCA
TOHERE2:    INCA                   # % 1 0 1 0 1 1 1 1
            POPA                     % 0 1 1 1 1 1 1 0
            SHR                      % 0 0 1 1 1 1 1 1
            SHL                      % 0 1 1 1 1 1 1 0
            STA     [$F031]        # % 0 1 1 1 1 1 1 0
```

3.) Írj olyan DIY Calculator Assembly programot, melyben bekérsz a billentyűkről ($F011) két hexadecimális számot 0 és 9 között, majd a képernyőre irod az összes decimális számjegyet a két beolvasott között, beleértve a beolvasott két számot is. Előre nem tudod, melyik szám lesz a nagyobb. (6p)

```
Példa:
Be: 8, 4
Ki: 4 5 6 7 8
```

```
.ORG $40000
                LDA       $10
                STA       [$F031]
    BE:         LDA       [$F011]   #elso szam beolvasasa
                JN        [BE]
                CMPA      $09
                JC        [BE]
                STA       [SZ1]
    BE2:        LDA       [$F031]   #masodik szam beolvasasa
                JN        [BE2]
                CMPA      $09
                JC        [BE2]
                STA       [SZ2]
                CMPA      [SZ1]     #SZ2 >? SZ1 (mivel ACC=SZ2)
                JC        [KIIR]
    CSERE:      STA       [TEMP]    #temp=sz2
                LDA       [SZ1]
                STA       [SZ2]     #sz2=sz1
                LDA       [TEMP]
                STA       [SZ1]     #sz1=temp
    KIIR:       LDA       [SZ1]
    IDE:        STA       [$F031]
                INCA
                CMPA      [SZ2]
                JNC       [IDE]
                JMP       [$0000]
    TEMP:       .BYTE
.END
```

4.) Írj olyan DIY Calculator Assembly programot, mely az enter gomb lenyomásáig billentyűkódot olvas a workbench-hez tartozó billentyűzetről ($F008), majd az enter ($05) beolvasása után kiírja a beolvasott karaktersorozatot visszafelé a beolvasás sorrendjében ($F031). (5p)

```
.ORG $4000
            BLDSP    $EFFF
            LDA      0          #vegjel a verembe
            PUSHA
     BE:    LDA      [$F008]
            JZ       [BE]       #$00 esetén nem volt gombnyomás
            CMPA     $05
            JZ       [KI]       #enter eseten nem olvas tovabb
            JMP      [BE]
     KI:    POPA                #kiiras
            JZ       [VEGE]     #vegjelnel alljon meg
            STA      [$F031]
            JMP      [KI]
     VEGE:  JMP      [$0000]
.END
```

5.) Írj olyan DIY Calculator Assembly **<u>szubrutint</u>**, mely induláskor a veremben egy nyolcbites értéket vár. A szubrutin cserélje ki a kapott érték első négy bitjét az utolsó néggyel, s ezután rakja vissza a verembe, majd térjen vissza. (7p)

```
Példa:
Híváskor a verem tetején: $F5 (vagy: %11110101)
Visszatérés után a verem tetején: $5F (vagy: %01011111)
```

```
SUBR:     POPA
          STA  [S_MSB]
          POPA
          STA  [S_LSB]

          POPA           # ACC: $XY
          PUSHA          # verembe vissza a $XY
          SHR
          SHR
          SHR
          SHR            # ACC: $?X
          AND  $0F       # ACC: $0X
          STA  [S_TEMP]

          POPA           # ACC: $XY - ezert mentettuk el
          SHL
          SHL
          SHL
          SHL            # ACC: $Y0
          AND  [S_TEMP]  # $YX

          PUSHA
          LDA  [S_LSB]
          PUSHA
          LDA  [S_MSB]
          PUSHA
          RTS
          S_LSB:    .BYTE
          S_MSB:    .BYTE
          S_TEMP:   .BYTE
```

6.) Írj lyan DIY Calculator Assembly programot, mely folyamatosan egy hexadecimális karaktert (0-F) ismételget a kijelzőn ($F031). A program végig ellenőrizze érkezik-e megszakítás és ha igen cserélje le a megjelenitendőt a nála eggyel nagyobb értékűre -- periodikus határfeltétellel. (0->1, 1->2, 9->A, **F->0**) Vigyázz, hogy a megszakítás kezelésekor az ACC és a státusztregiszterek értéke ne sérüljön (azaz a megszakításkezelő szubrutin végén álljon vissza az eredeti állapotába). (6p)

```
Példa futás:
CCCCCCCCDDDDDEEEEEFFFFFFF000000
       ↑    ↑    ↑      ↑
     m e g s z a k í t á s o k
```

```
A feladat szövege sajnos hibás. Az ACC-t muszáj módosítani a
szubrutinban, hogy azt csinálja, amit kell. A környezet
visszaállítása viszont számon lehet kérve más feladatban!


.ORG $4000
          BLDSP          $EFFF     # a szubrutin vermet hasznal
          BLDIV          KEZELO
          SETIM

          LDA  0
     KI:  STA  [$F031]
          JMP  [KI]

     KEZELO:   PUSHSR
               INCA
               CMPA       $0F
               JC         [K_VEGE]
               LDA        0
     K_VEGE:   POPSR
               RTI
.END
```