HWA 4. LABOR - FELKÉSZÜLÉS

Mikrokontroller labor 1.

6. javított változat, 2019. 11. 18. – 2019. 11. 28.

Huszár Csaba Benedek

Mi a szubrutin

- Kb. olyan, mint Pythonban a függvény, csak máshogyan kell megírni.
- A szubrutinban is lehet szubrutin (rekurzió, 16 a maximális „mélység”).
- A szubrutinok egymás után hívódnak meg, hacsak ezt meg nem változtatjuk, gátoljuk.

Változókezelés – nagyon körülményes

- Alapból egy változó 8 bites, így 0…255 közötti értékek lehetnek benne.
- Egyetlen konkrét értékkel megadható változó: W regiszter (akkumulátor)
- Új változót azért még lefoglalhatunk. (Pl: `valtozo RES 1`, a `MAIN_PROG_CODE` előtt)
- CSAK WREG-en keresztül lehet megadott számértéket (konstanst) más változókba beletölteni! (`MOVLW`-vel betöltjük W-be → `MOVWF`-fel átrakjuk egy másik változóba)
  - *PL:*
    - `MOVLW d'20'`
    - `MOVWF LATA` → ekkor LATA értéke 00010100 lesz binárisan
  - *Változóból változóba töltéskor nem teljesen ezt a módszert kell használni. (`MOVLW` helyett `MOVF` kell, de ez még a 4. laboron nem kerül elő)*
- A bemeneti-kimeneti portbiteket (meg minden más változó tetszőleges bitjét is) 0-1 között állíthatjuk.
  - *BCF kikapcsolja (Pl. `BCF LATA, 7`)*
  - *BSF bekapcsolja (Pl. `BSF LATA, 5`)*
  - ***A következő utasítás érkezésekor lépnek életbe***

Változókezelés – bitek sorszáma

A legnagyobb helyiértékű kapja a legmagasabb sorszámot!

7	6	5	4	3	2	1	0
$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$^0 = 1$
PI: LATA, 7	PI: LATA, 6	PI: LATA, 5	PI: LATA, 4	PI: LATA, 3	PI: LATA, 2	PI: LATA, 1	PI: LATA, 0

BE-és kimeneti regiszterek

- PORT(A,B,C,D)
  - *Bemenetetek (PL nyomógomb)*
- LAT(A,B,C,D)
  - *Kimenetek (PL LED)*
- TRIS(A,B,C,D)
  - *Irányok megadására, (bemenet, vagy kimenet)*
- ANSEL(A,B,C,D)
  - *Digitális vagy analóg portbitek beállítása*
  - *Az elején mindent digitálissá kell állítani.*
    - De ezt a mintaprojektek minden esetben megteszik helyettünk.
  - *ADC konverziót kiszedték a laborokról, így a potméterrel nem kell bajlódni.*

További ASM parancsok, amik kellenek a laborra

- `DECFSZ WREG,W` (W értékét eggyel csökkenti), ha az 0 lesz, akkor az eggyel alatta lévő utasítás kimarad.
- `DECFSZ VALTOZO,F` (VALTOZO értékét eggyel csökkenti), ha az 0 lesz, akkor az eggyel alatta lévő utasítás kimarad.

1. feladat: Szoftveres időzítő

- **A labor legfontosabb része**
- Többet is kell belőle írni
- Tudni kell egymásba ágyazni őket
- **Tipikus vizsgapélda (5-6 pontért, ami egészen sok)**
- **Minimális matek és számológép (vizsgán írásbeli számítások) kellenek hozzá**

Szoftveres időzítő PIC16F18875 esetén

34. feladat

Írjon assembly szubrutin (Delay10us), amely 10us időt késleltet szoftveresen. A processzor ciklusideje 125ns. Számítsa ki a késleltetéshez szükséges konstans értékét.

Megoldás:

```
Delay10us: ; 125ns/ciklus@Fosc=32MHz-->8MHZ instructionclock
     MOVLW   D'25'      ; 3
D10CK:
     DECFSZ  WREG,W     ; 1 /2
     GOTO    D10CK      ; 2
     RETURN             ; 2
```

Egy utasításciklus ideje: Tcy=1/8MHz = 125ns

Az előírt 10us késleltetéshez tehát összesen 10us/125ns = 80 ciklusnyi idő szükséges. A szubrutint meghívó CALL utasítás 2 ciklus, a hurok változó kezdőérték betöltése 1 ciklus. A hurok mag egy iterációja 3 ciklus, míg a ciklus végén a visszatérés további 2 ciklus.

Az összes ciklusok száma: $N = 3+(m\cdot3)+2 = 5 + (m\cdot3)$ vagyis:

$80 = 5+ (m\cdot3) \rightarrow m=25$.

Időmérés – a másodpercek alatt

Név	Másodpercben	Jelölés
Másodperc	**1 sec**	**sec**
Milliszekundum	**$1*10^{-3}$ sec**	**ms**
Mikroszekundum	**$1*10^{-6}$ sec**	**µs (us)**
Nanoszekundum	**$1*10^{-9}$ sec**	**ns**

Készítsünk 10 us késleltetést 1.

- A processzor egy utasítást 125 ns alatt dolgoz fel.
  - *Mert 8MHz az órajele*
- 10 us = 10*1000 = 10000 ns
- 10000/125 = 80 (Ha nem kerek a szám, kerekítsünk annak szabályai szerint)
- De mégsem 80 a megoldás.

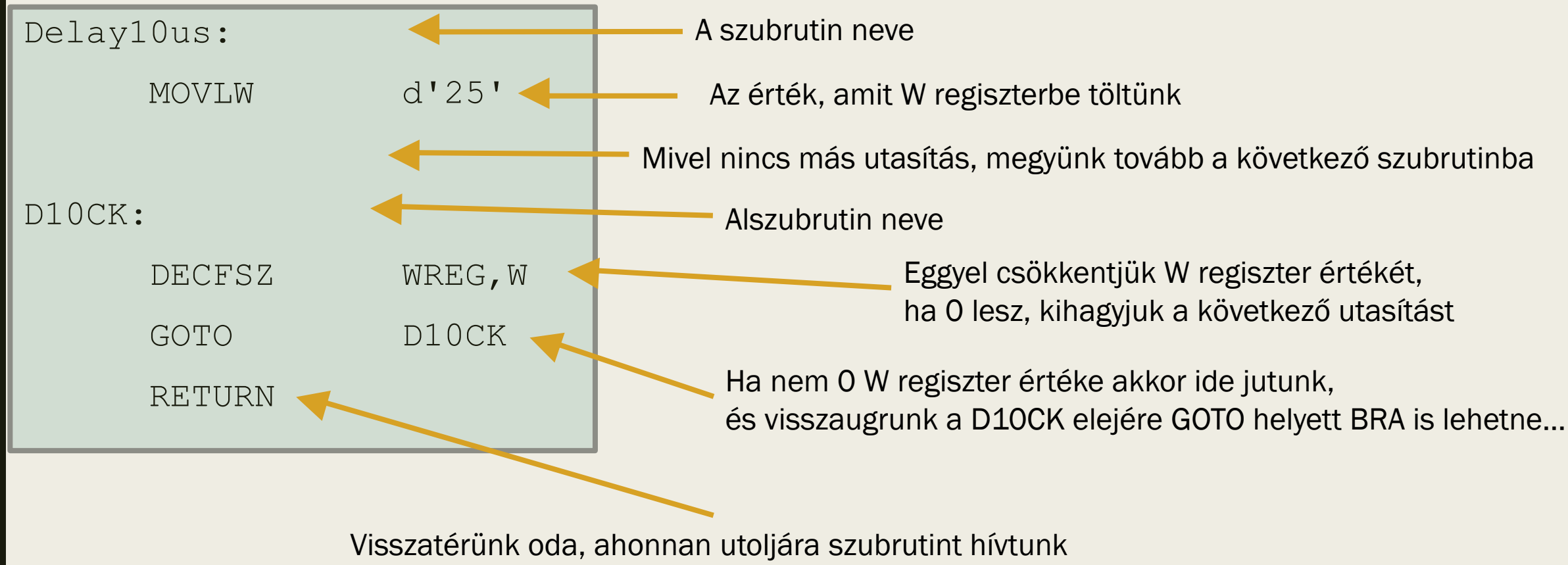
Készítsünk 10 us késleltetést 2.

- A szubrutin meghívásához (CALL Dealy10us) 2 utasításnyi időt kell használnunk (lásd 1 lapos segédlet). Így 80-2=78-nál tartunk.
- A szubrutinban először betöltjük, hogy mennyiszer szeretnénk a késleltető szubrutinban a tényleges belső késleltetést végrehajtani. Ez 1 utasítás, így 78-1=77-nél tartunk
- A szubrutinból való visszatérés további 2 ciklust vesz igénybe, így 77-2=75-nél járunk
- **Ezek az értékek mindig ennyi időt (2+1+2=5 órajelet) vesznek igénybe.**
- De ez (75) még mindig nem a jó válasz

Készítsünk 10 us késleltetést 3.

- A belső szubrutin lefutása hívásonként 3 órajelet vesz igénybe (ez is mindig így van)
- Így a megoldás: 75/3=25 értéket kell W regiszterbe mozgatnunk.

Mi ez a kód???


Delay10us:
MOVLW d'25'
D10CK:
DECFSZ WREG,W
GOTO D10CK
RETURN
A szubrutin neve
Az érték, amit W regiszterbe töltünk
Mivel nincs más utasítás, megyünk tovább a következő szubrutinba
Alszubrutin neve
Eggyel csökkentjük W regiszter értékét,
ha 0 lesz, kihagyjuk a következő utasítást
Ha nem 0 W regiszter értéke akkor ide jutunk,
és visszaugrunk a D10CK elejére GOTO helyett BRA is lehetne...
Visszatérünk oda, ahonnan utoljára szubrutint hívtunk

Hogyan hívjunk szubrutint?

- A kódban:

- **`CALL szubrutinnév`**
- A szubrutinban (vagy alszubrutinban) lefutó RETURN után ide jutunk vissza

Írjunk új késleltető (Delay1ms) szubrutint az előzőt felhasználva!

- 1 ms = 100*10us
- Itt már ne szórakozzunk a kerekítésekkel, levonásokkal
  - *minimális lesz a különbség.*
- Tehát 100-szor kell hívni a Delay10us szubrutint
- Ez még (100) kisebb, mint 255, tehát OK

Mi a hiba?

```
Delay1ms:
        MOVLW           D'100'
D100CK:
        CALL            Delay10us
        DECFSZ          WREG,W
        GOTO            D10CK
        RETURN
```

WREG-et másra is használjuk....
Visszatéréskor W értéke 0
Ha 0-ból kivonunk 1-et, akkor 255-öt kapunk.
Ami nagyon nem 99.

A megoldáshoz vezető út

- Változó bevezetése:
- A MAIN_PROG CODE előtt közvetlenül lévő helyre írjuk:
- `DELAY RES 1`
- **Az akkumulátorközpontú kezelés miatt továbbra sincsen egyszerű dolgunk…**

Ez már tényleg a (jó) megoldás

```
Delay1ms:
        MOVLW       D’100'
        MOVWF       DELAY
D100CK:
        CALL        Delay10us
        DECFSZ      DELAY,F
        GOTO        D100CK
        RETURN
```

A Delay10us alá írjuk

További időzítések

- A 250 ms-nél már egyszerűbb dolgunk van.
- Az előbb bemutatott módszert alkalmazzuk.
- Ez már ki is lehet próbálni az „igazi” fejlesztőpanelen

Delay250ms

```
DELI    RES 1
…
```

Ugyanúgy a MAIN_PROG_CODE elé írjuk

```
Delay250ms:
        MOVLW       D'250’
        MOVWF       DELI
D250CK:
        CALL        Delay1ms
        DECFSZ      DELI,F
        GOTO        D250CK
        RETURN
```

A Delay1ms alá írjuk

0. feladat: HIBÁK a mintaprojektben

- A mikrovezérlő típusa rosszul van megadva (PIC16LF18875 helyett PIC16F18875 kell)
  - *Bal oldalon középen csavarkulcs ikon → Device-nál az L-t kiszedni*
- `BSF  LED_D5_IRANY` →BSF HELYETT BCF kell, ahogyan minden más helyen is van…
- `MOVLW  D'10'` → 10 helyett 25 kell, korábban láttuk, miért
- `BTFSC  STATUS,C` → BTFSC STATUS,Z kell (Nem a Carry kell, hanem a Zero flag)
- `MOVLW  D'02'` → MOVLW D’03’ kell
  - *Ha először csökkentjük, akkor a 1 lesz, utána meg már 0 és visszamegy a MAIN-be, vagyis csak egy impulzus generálódik.*
- A LED kikapcsolása után 2X kell meghívni a Delay10us-t, nem csak egyszer.
- A szintaktikai hibás sort törölni kell

2. Feladat – adott idejű impulzus előállítása

- Lényegében készen vagyunk vele
  - *A Delay10us helyett a Delay1ms-t kell hívnunk. DE CSAK EGYSZER!*
    - Ez még egyszerűbb is. Sokkal.
      - *Var0-ban a BITCNT változót 1-gyel csökkenteni kell → 2 legyen!*
      - *PULS_LOOP a következő lesz: Ciklusváltozó csökkentése, LED BE, Várunk, LED KI, ugrás PULS_LOOP elejére*
  - *A START-ot lényegében ki kell szerveznünk BOADRINIT-té és a START elé be kell RAKNI.*
    - `BANKSEL ALSELA` (ettől a sortól)
    - .....
    - `CLRF STATUS` (eddig a sorig)
    - A végére kell egy `RETURN`
  - *A START a következővé alakul át:*
    - `CALL BOARDINIT`

2. Feladat – kód 1. rész

```
BOARDINIT:
    BANKSEL   ANSELA
    CLRF      ANSELA ; port digitális
    CLRF      ANSELB ; port digitális
    CLRF      ANSELC ; port digitális
    BANKSEL   LATA
    BCF       LED_D2    ; kezdoérték
    BCF       LED_D2_IRANY  ; kimenet
    BCF       LED_D3    ; kezdoérték
    BCF       LED_D3_IRANY  ; kimenet
    BCF       LED_D4    ; kezdoérték
    BCF       LED_D4_IRANY  ; kimenet
    BCF       LED_D5    ; kezdoérték
    BCF       LED_D5_IRANY  ; kimenet
    BSF       Gomb_S1_IRANY; bemenet
    CLRF      STATUS
    RETURN

START:
     CALL BOARDINIT
```

2. Feladat – kód 2. rész

```
MAIN_LOOP:
    BTFSS  Gomb_S1   ; alaphelyzetre vár
    GOTO   MAIN_LOOP
Var0:
    BTFSC  Gomb_S1   ; lefutó él
    GOTO   Var0
    MOVLW  D’02'
    MOVWF  BITCNT
PULS_LOOP:
    DECF   BITCNT,F  ; ciklus változó csökkentése
    BTFSC  STATUS,Z
    GOTO   MAIN_LOOP
    BSF    LED_D5 ; LED bekapcsolása
    CALL   Delay1ms
    BCF    LED_D5 ; LED kikapcsolása
    GOTO   PULS_LOOP
```

3. feladat: jobbra, bitmintával

- Milyen értéket kell betöltenünk (WREG-be) ?
  - *10000000-t, binárisan, hiszen az első 4 bit (7-6-5-4) lesznek a LED-ek bitjei*
- Ezt be kell töltenünk LATA-ba is. W→LATA
- Ezt követően kell jobbra shiftelni (LSRF LATA), amig 1 nem lesz az a bit, ami a jobb szélső LED portbitje (LATA, 4). Ekkor megint be kell tölteni az értéket LATA-ba.

3. feladat, jobbra, kód

```
PULS_JOBBRA:
      MOVLW         b'10000000’
      MOVWF         LATA
      CALL          Delay250ms
PULS_LOOP_JOBBRA:
      LSRF          LATA
      CALL          Delay250ms
      BTFSS         LATA, 4
      GOTO          PULS_LOOP_JOBBRA
      GOTO          PULS_JOBBRA
```

A MAIN_LOOP helyére írjuk. A Var0 és a PULS_LOOP is törölhető. Figyeljünk arra, hogy ha szimulátorban futtatjuk a programot, akkor a Dely10us-sel hívjuk.

3. feladat: balra, bitmintával

- Milyen értéket kell betöltenünk (DATAREG_L-be) ?
  - *00010000-t, binárisan, hiszen az első 4 bit lesznek a LED-ek bitjei, amelyek közül most a legjobboldalira van szükség.*
- Ezt be kell töltenünk LATA-ba is. W→LATA
- Ezt követően kell balra shiftelni (LSLF LATA), amig LATA(7) 1 nem lesz, akkor megint vissza kell tölteni a megfelelő értéket LATA-ba!

3. feladat: balra, kód

```
PULS_BALRA:
       MOVLW  b'00010000'
       MOVWF  LATA
       CALL   Delay250ms
PULS_LOOP_BALRA:
       LSLF   LATA
       CALL   Delay250ms
       BTFSS  LATA, 7
       GOTO   PULS_LOOP_BALRA
       GOTO   PULS_BALRA
```

A PULS_JOBBRA helyére írjuk.

3. feladat: Gomb kezelésével - végleges

```
PULS_JOBBRA:
    MOVLW  b'10000000'
    MOVWF  LATA
    CALL   Delay250ms
PULS_ELL_JOBBRA:
    BTFSC  Gomb_S1
    GOTO   PULS_LOOP_JOBBRA
    GOTO   PULS_BALRA
PULS_LOOP_JOBBRA:
    LSRF   LATA
    CALL   Delay250ms
    BTFSS  LATA, 4
    GOTO   PULS_ELL_JOBBRA
    GOTO   PULS_JOBBRA
```

```
PULS_BALRA:
    MOVLW  b'00010000'
    MOVWF  LATA
    CALL   Delay250ms
PULS_ELL_BALRA:
    BTFSS  Gomb_S1
    GOTO   PULS_LOOP_BALRA
    GOTO   PULS_JOBBRA
PULS_LOOP_BALRA:
    LSLF   LATA
    CALL   Delay250ms
    BTFSS  LATA, 7
    GOTO   PULS_ELL_BALRA
    GOTO   PULS_BALRA
```

Alapból jobbra megy, de amig nyomjuk az S1 gombot, addig balra
Mindkét oszlopban lévő kód kell, egymás után!

Kérdések a 3. feladathoz

- Miért nincs DATAREG változónk?
  - *Mert felesleges bonyolítás lenne, a programkód pedig csak hosszabb lenne tőle*
  - *Anélkül is megoldható, WREG-ből közvetlenül LATA-ba tesszük az adatot*
- W = W regiszter = WREG ?
  - *IGEN, mindhárom ugyanazt jelenti, de néha máshogyan kell rá hivatkozni.*
- Mi a BTFSC és a BTFSS Assembly parancsok jelentése?
  - *BTFSC: kihagyja az utána lévő utasítást, ha a megadott bit (PL: LATA, 7) 0-ás értékű*
  - *BFTSS: kihagyja az utána lévő utasítást, ha a megadott bit (PL: LATA, 7) 1-es értékű*