Mikrokontroller II. mérési jegyzőkönyv

Mátyás ANTAL
(Supervisor: Attila TIHANYI)
Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
50/a Práter street, 1083 Budapest, Hungary
antal.matyas.gergely@hallgato.ppke.hu

***Abstract*—A mérés célja volt a mikrokontrollerek gyakorlati megismerése, az MSP 430-196 mikrokontroller használata, egyszerű alapműveletek végrehajtása, ezzel a flag bitek működésének gyakorlati vizsgálata.**

I. Mérendő objektumok

A mérés során az MSP 430-196 mikrokontrollert, valamint egy számítógépet és az ezen futó programozási környezetet használtunk, ennek segítségével végeztünk a mikrokontrolleren egyszerű alapműveleteket, összeadást és kivonást, ezzel ismerkedve a műveletvégzéssel, valamint a flagek használatával. A mérés során a műveletek elvégzésére használt programrészleteket a jegyzőkönyvbe illesztem, valamint az elkészült programot az emailhez csatolom.

II. LED kigyújtása

Az első feladat megvalósítására a mérési utasításban kapott példaprogram alkalmas volt, így ennek működését vizsgáltuk. A kódrészlet első két parancsa nullázza a P2Out és a P2DIR 1-es bitjét, ezzel kikapcsolva a LED-et. A bit.b parancs a carry bitbe tölti a joystick P2IN gombjának helyzetét. Gombnyomás esetén ennek értéke 0, különben 1. A jc paranccsal visszaugrunk a program elejére, ha a carry 1, vagyis nem történt gombnyomás. A parancs miatt a program csak akkor jut tovább, ha az előző feltétel hamis volt, azaz a gombot megnyomtuk, ekkor a LED kigyullad. Ezután a program visszaugrik az elejére, várva a következő gombnyomást.

```
asmmain:
     ; Ide írhatod az asm-programot!
       mov.b #0x0034, R12
       call #hexdraw

minta:  bic.b #STAT2
        bic.b #STAT
        bit.b #BUTTON
        jc minta
        bis.b #STAT2
        jmp minta
```

III. Szám növelése gombnyomásra

A feladat megoldásához az R4 regiszterbe töltöttük a kezdőértéket, ezt fogjuk később növelni. A feladatban az értéket az R12 regiszterbe töltöttük, majd a $hexdraw$ függvény segítségével a képernyőre írattuk. Az előző feladathoz hasonlóan minden ciklusonban ellenőriztük a nyomógomb értékét a carry flagbe töltve. Amennyiben nem történt gombnyomás, a jc paranccsal a program elejére ugrottunk. Amennyiben a gombot megnyomtuk, az inc.b parancs segítségével az R4 értékét eggyel megnöveltük, majd az R12 regiszterbe töltve, a $hexdraw$ függvény ismételt használatával kirajzoltattuk.

Mivel egy gombnyomás ideje alatt több ciklus is lefuthatott, nehéz volt a számot egyesével növelni. Ennek megoldására hoztuk létre a $nyomom$ függvényt, mely a gomb elengedéséig nem engedte a program újraindulását. Ezzel értük el, hogy egy gombnyomás csak egy számnövelést eredményezzen.

```
asmmain:
     ; Ide írhatod az asm-programot!
     mov.b 1, R4
feladat1:  bic.b #STAT2
        bic.b #STAT
        bit.b #BUTTON
        jc feladat1
        bis.b #STAT2
        inc.b R4
        mov.b R4,R12
        call #hexdraw

nyomom:  bit.b #BUTTON
        jnc nyomom
        jmp feladat1

      ret
```

IV. Decimális kiíratás

A tízes számrendszerbeli számkiíratáshoz az előző programot csak kevéssé kellett módosítanunk. Ehhez a BCD formátumot, és a $dadd$ műveletet használtuk. A parancs úgy végzi el az összeadást, hogy a megjelenő eredmény tízes számrendszerbelinek tűnjön. A papíron történő átváltáshoz hasonlóan a számokat elosztjuk tízzel, majd a maradékokból keletkeznek az átalakított szám számjegyei a hozzájuk tartozó hatványértékekkel. Ha ezeket összeadjuk, megkapjuk a szám decimális értékét.

```
asmmain:
     ; Ide írhatod az asm-programot!
     mov.b 1, R4
feladat2:
        mov.b R4, R12
        call #hexdraw
        bic.b #STAT2
        bic.b #STAT
        bit.b #BUTTON
        jc feladat2

        bis.b #STAT2
        dadd 1, R4
        mov.b R4,R12
        call #hexdraw

nyomom:  bit.b #BUTTON
        jnc nyomom
        jmp feladat2

      ret
```

V. Joystick irányítás

A joystick segítségével való irányításhoz az előre definiált irányokat használtuk. A korábbiakhoz hasonlóan vizsgáltuk az egyes irányokba való elmozdítást, ezeknek értékét a carry bitbe tölöttük, majd a jnc parancs segítségével, amennyiben adott

irányba való elmozdítás történt, az erre létrehozott mozgatási programrészletekhez ugrottunk.

```
asmmain:
    ; Ide írhatod az asm-programot!
    mov.b 7, R5 ; x coordinate
    mov.b 2, R4 ; y coordinate
    mov.b #0x4F, R6 ; O char
    call #draw
feladat3:
       call #clear
       bit.b #LEFT
       jnc movel
       bit.b #RIGHT
       jnc mover
       bit.b #UP
       jnc moveu
       bit.b #DOWN
       jnc moved
       jmp feladat3


      movel:
              dec R5
              call #draw
              break1:
              bit.b #LEFT
              jnc break1
              jmp feladat3

      mover:
              inc R5
              call #draw
              break2:
              bit.b #RIGHT
              jnc break2
              jmp feladat3

      moveu:
              dec R4
              call #draw
              break3:
              bit.b #UP
              jnc break3
              jmp feladat3

      moved:
              inc R4
              call #draw
              break4:
              bit.b #DOWN
              jnc break4
              jmp feladat3
            ret
```

További függvényeket definiáltunk még - draw és clear. A draw függvény meghívásakor a joystick-al módosítható x és y koordinátákat tartalmazó R4 és R5 regiszterek értékét az R12 és R13 regiszerekbe töltöttük, majd az 'o' karaktert tartalmazó R6 regiszter értékét az R14-be, majd az $LCDChrXY$ valamint az $LCDUpdate$ parancsok hívásával adott pozícióra kiírattuk az LCD-re. A clear függvény arra szolgált, hogy az előző kirajzolt karaktert felülírja egy szóközzel, így elérve, hogy egy időpontban csak egy kör legyen kirajzolva.

```
draw:
      mov.b R5, R12
      mov.b R4, R13
      mov.b R6, R14
      call #LCDChrXY
      call #LCDUpdate
      ret

clear:
      mov.b #0x20, R14
      mov.b R4, R13
      mov.b R5, R12
      call #LCDChrXY
      ret
```