Mikrokontroller vizsgálata

Erdélyi Áron
Pázmány Péter Catholic University, Faculty of Information Technology and Bionics
50/a Práter street, 1083 Budapest, Hungary
erdelyi.aron.janos@hallgato.ppke.hu

I. A mérés körülményei

A mérést 2018.03.14-én 16:00 és 19:00 között a PPKE ITK, 1083 Práter utca 50/a cím alatt található épületében, a 4. emeleti mérési laborban végeztem, Czirják Kristóf hallgatótársammal együtt. A mérés során az MSP430 mikrokontrollert, és a demo.asm programot vizsgáltuk.

II. A környezet felmérése

A mérést azzal kezdtük, hogy megfigyeltük, hogy az alap program mit csinál.

A kapott program működését a mikrokontroller LED kijelzőjén tudtuk nyomon követni. Indítás után egy labda pattog, a kijelző két szélén két vonal ment fel- és le, egymással ellentétesen.

Észre vehető volt hogy a program nem működik teljesen úgy, amilyenre tervezték, két dolog miatt:

- A labda az egyik oldalon túlment a falon, a másik oldalt meg még előtte 1 pixellel pattant le.
- A két egymással ellentétesen mozgó vonal a kijelző egyik végénél nem pattant vissza, hanem tovább ment egy ideig, ki a képernyőből.

III. A program értelmezése

Mérésünk azzal folytatódott, hogy megvizsgáljuk a program kódját. Az egyszerűség kedvéért az első lépésünk az volt, hogy minden sor mellé egyesével odaírtuk megjegyzésbe, hogy mit csinál, így átláthatóbb lett, és könnyebben ki tudtuk javítani a hibákat.

IV. Változók definiálása

```
#define x                R10
#define y                R6
#define dx               R8
#define dy               R9
```

A program elején, még mielőtt akármilyen műveletet végezne a program, a #define parancs segítségével hozzárendeljük a változóneveket különböző regiszterekhez.

Ebben az esetben láthatjuk, hogy 4 regisztert használunk föl a 4 változó deklarálásához: Az x változó az R10, az y az R6, a dx az R8, és a dy az R9 regiszterre mutat.

Ennek funkcionálisan semmi következménye nincsen. Ezt azért találták ki, hogy megkönnyítsék a programozók munkáját, hogy ne regiszterneveket kelljen megjegyezni, hanem elnevezhetik a változóikat.

```
;dx lesz x megváltozása
  clr.w   x ; x kiürítése
  clr.w   y ; y kiürítése
  mov.w   #1,dx ; dx:=1
  mov.w   #1,dy ; dy:=1
```

A következő kódrészletben alaphelyzetbe állítjuk a labdát, Az x és az y értékek kiürülnek - lenullázódnak. Ebben a helyzetben a labda koordinátái: (0,0).

Ezen kívül beállítódik a dx, és a dy, a labda x és y irányba felbontott mozgásvektora, 1-re. Mivel a számítógépes világban az y tengely a megszokotthoz képest fejjel lefelé van, ezért a két vektor összege egy 45 fokban lefelé irányuló vektort ad ki.

V. Ciklus

```
Ciklus:
    mov.w   y,R11 ; R11:=y
    add.w   dy,R11 ; R11:=R11+dy
    cmp.w   #41,R11 ; ÖSSZEHASONLÍT 41,R11 (javítva 40-ről)
    jge     Cimke001 ; HA R11>=40: GOTO Cimke001
    cmp.w   #0,R11 ; ÖSSZEHASONLÍT 0,R11 (javítva -1-ről)
    jge     Cimke002 ; HA R11>=-1: GOTO Cimke002
```

A következő kódrészlet felelős azért, hogy a labda ne menjen ki se a kijelző tetején, se az alján.

Első lépésként az R11-es regisztert egyenlővé teszi az y-al (jelen esetben 0). Ezután hozzáadja az R11-es regiszterben szereplő értékhez a dy-t, majd megnézi, hogy az így kapott érték nagyobb, vagy egyenlő-e mint 41 - azaz ha a képernyő szélén, vagy azon kívül van. Ha nagyobb, akkor ugrik a Cimke001 nevű programrészletre.

Ellenkező esetben, ha az R11-ben szereplő érték kisebb mint 40, akkor megnézi, hogy ez az érték nagyobb-e, mint 0, tehát ha a labda y értéke 0 és 41 között van. Ha igen, akkor ugrik a Cimke002 nevezetű kódrészletre.

Ha az R11 értéke kisebb, mint 0, azaz a pálya szélén, vagy a pályán kívül van, akkor tovább megy a következő részletre, a Cimke001-re.

VI. Cimke001

```
Cimke001:
    xor #0xffff,dy ; 1-es komplemens képzés
    inc dy ; 2-es komplemens képzés: dy:=-dy
```

A Cimke001-re hallgató programrészlet felelős azért, hogy a dy változó értékének vegye az ellentétét. Ezt binárisban úgy lehet a legegyszerűbben megcsinálni, hogy leképezzük az adott érték kettes komplemensét.

Először invertáljuk a bináris értéket, így leképezve az egyek komplemenst, majd ehhez az értékhez egyet hozzáadva megkapjuk a kettes komplemenst, ami egyenlő az eredeti érték ellentettjével.

VII. Cimke002

```
Cimke002:
    add.w   dy,y ; y:=y+dy
    mov.w   x,R11 ; R11:=x
    add.w   dx,R11 ; R11:=R11+dx
    cmp.w   #77,R11 ; ÖSSZEHASONLÍT 77,R11
    jge     Cimke003 ; HA R11>=77: GOTO Cimke003
    cmp.w   #0,R11 ; ÖSSZEHASONLÍT 0,R11 (javítva -1-ről)
    jge     Cimke004 ; HA R11>=-1: GOTO Cimke004
```

A következő kódrészlet első sorában láthatjuk, hogy miután a dy értékét meg tudtuk adni, az y változó értékéhez hozzáadja a dy értékét.

Ezután megismétli azt, ami a Ciklusban is történt, csak az x változóra nézve. Megnézi, hogy ha az x értékéhez hozzáadnánk a dx értékét, akkor a labda még bent lenne-e a pályán. Ha nem, akkor ugrik a Cimke 003 kódrészletre, ha a labda jó pozícióban lesz, akkor továbbmegy a Cimke004 nevezetű részletre.

VIII. Cimke003

```
Cimke003:
    xor #0xffff,dx ; 1-es komplemens képzés
    inc dx ; 2-es komplemens képzés: dx:=-dx
```

A Cimke003 nagyon hasonló a Cimke001-hez. Gyakorlatilag ugyan az történik, csak dy helyett a dx értékének veszi az ellentettjét, és ezt teszi egyenlővé dx-szel.

IX. Cimke004

```
Cimke004:
    add.w   dx,x ; x:=x+dx
    mov.b   y,R13 ; R13:=y
    mov.b   x,R12 ; R12:=x
    call  #LCDLabdaXY ; Labda KIRAJZOLÁS
    mov.b   y,R12 ; R12:=y (javítva x-ről)
    call  #LCDUtoLXY ; BAL ÜTŐ KIRAJZOLÁS
    mov.b   #40,R12 ; R12:=40
    sub.b   y,R12 ; R12:=R12-y (javítva x-ről)
    call  #LCDUtoRXY ; JOBB ÜTŐ KIRAJZOLÁS
    call  #LCDUpdate ; KÉPERNYŐ FRISSÍTÉS
   jmp    Ciklus ; GOTO Ciklus
        ret
```

A Cimke004 első sorában láthatjuk, hogy az x értékéhez a megfelelő dx értéket adja hozzá, így a labda x, és y pozíciója is a pályán belül marad.

Ezután a rogram az R13-a regiszter értékét megváltoztatja az y változó értékére, valamint az R12 értékét az x értékére. Ezt feltehetően azért teszi, mert a következő - máshol definiált - programrészletek ezeket a regisztereket használják.

A call parancs meghív egy másik programot, jelen esetben a labda kirajzolásáért felelős programot.

A következő sorokban a program az R12-es regiszter értékét egyenlővé teszi y értékével, majd meghívja a bal ütő kirajzolására szolgáló programot.

Miután kirajzolódott a baloldali ütő, az R12-es regiszter értéke megváltozik 40-y-ra, majd kirajzolja a jobboldali ütőt is. Ez azt a hatást kelti, mintha a két ütő tükrözve lenne a kijelző közepére.

Ezután frissíti a képernyőt, majd visszaugrik a Ciklus programrészletre, és ez a ciklus megy addig, amíg manuálisan le nem állítják.

X. Javítások

A program alapból nem működött tökéletesen, ezért több helyen belenyúltunk a programba, hogy azt a hatást érjük el, amiről azt gondoltuk, hogy jobb, mint az eredeti programban.

Ezeket a sorokat megjelöltük a sorok melletti megjegyzésekben.