Mikrokontroller jegyzőkönyv

Pákozdi Gergely Ferenc PRLAAJ
2019. 04. 30.
(Mérőpartner: Nagy Fanni és Zsolt Luca)
Pázmány Péter Katolikus Egyetem Információs Technológiai Kar
Budapest, Práter u. 50/A, 1083, Hungary
pakozdi.gergely.ferenc@hallgato.ppke.hu

***Kivonat*—A mérérseket elvégezte Pákozdi Gergely, Nagy Fanni és Zsolt Luca 2019. április 30-án, a Pázmány ITK 420-as termében. A mérésekről rövid leírást és értékelést olvashat ebben a dokumentumban. Méréshez az alábbi eszközöket használtuk fel: A Texas Instruments OLIMEX MSP-430-169LCD mikrokontrollere, IAR Embeded Workbench IDE**

***Keywords*-mikrokontroller, uC**

I. Mérési feladat

A mérés elején bemutatták nekünk az adott órai feladatot, miszerint 2-es, 3-as csoportokban leültünk egy olyan géphez, melyhez már előre csatlakoztatva volt egy mikrokontroller, melyhez egy szintén előre megírt hibásan működő ping-pong program kódját kellett értelmeznünk és javítanunk tudásunk és a dokumentáció alapján.

I-A. Ping-Pong

Egy labdát láttunk a mirokontroller kijelzőjén, mely a képernyő négy oldala között pattogott úgy, hogy a bal, illetve jobb oldalán kis ütőket jelképező vonalak mozogtak fel alá.

II. Mérés menete

Leültünk a számítógéphez, majd elindítottuk a az IAR Embeded Workbench nevű programot. Ezután beolvastuk a honlapról letöltött programkódot, a fordítás után pedig Debug módban vizsgáltuk a program működését először lépésről lépésre, majd pedig a számunkra kérdéses részeket külön külön esetleg bizonyos értékek megváltoztatásával, tesztelésével.

III. A program kódja

Assembly nyelven előre megírva kaptuk, feladatunk az volt, hogy értelmezzük, minden sor után kommentben írjuk oda az adott sor jelentését, valamint javítsuk az esetleges hibákat.
Részei:

1) Deklarálás
   A programban váltotókat definiál, melyekhez a mikrokontroller egy-egy registerét rendeli hozzá. Változók: x: labda x koordinátája, y: labda y koordinátája, dx: x irányú sebesség, dy: y irányú sebesség. Ezután értéket ad a program az egyes változóknak, x-et, y-t nullázza, dx, dy értéket 1-re állítja be.
2) Ciklus
   A ciklus leírja az ismétlődő részt. Legkülső kódrészében az új y koordinátáról dönti el, hogy érvényes-e, azaz, hogy benne van-e a kijelezhető tartományban. Ha lefelé kimennénk akkor Cimke001-re ugrunk, ha lefelé és felfelé sem mennénk ki, akkor a Cimke002-re ugrunk, hiszen ez egy megengedett lépés, különben a Cimke001 fut le.
3) Cimke001
   Megváltoztatja a dy-t tehát az függőleges sebesség irányát az aktuálissal ellentétesre.
4) Cimke002
   Mivel az új y-t megvizsgáltuk, oda is lépünk, majd y-et hasonlóképpen megvizsgáljuk Cimke003 és Cimke004 segítségével.
5) Cimke004 A megvizsgált x helyre lépünk, majd kirajzoljuk a labdát, és az ütőket. Majd kezdjük az egészet elölről

IV. A kód hibája

A kód hibája az volt, hogy az ütők bár az y tengely mentén mozogtak eredetileg mégis az x koordinátától függöt a mozgásuk, illetve a jobb oldali ütő ráadásul inverz mozgást végzett a helyes mozgáshoz képest. Tehát annyit javítottunk a programkódban, hogy a Cimke004 5. sorában x-et y-ra cseréltük, a 7. sorban a 40-et 0-ra változtattuk, illetve a 8. sorban a kivonást összeadásra változtattuk és x-et ismét y-ra cseréltük.

demo.asm

```
#include  <io430x16x.h>
#include  <msp430x16x.h>
      rseg code:CODE,2000h

      extern   LCDUpdate, LCDStr,LCDChrXY, DelayN, LCDPixmove
      extern    LCDUtoLXY, LCDUtoRXY, LCDLabdaXY
      public asmmain



////////////////////////////////////////////////
asmmain:
     ; ide írhatod az asm-programot

  ;innentől, pedig ne módosíts semmit!!
#define LeftValue   &ADC12MEM0
#define RightValue  &ADC12MEM1
   ; Kérem értsék meg!
#define x                       R10
#define y                       R6
#define dx                      R8
#define dy                      R9
  ;dx lesz x megváltozása
    clr.w   x ;a labda x koordinátájának memóriaterületét lenullázzuk
    clr.w   y ;a labda y koordinátájának memóriaterületét lenullázzuk
    mov.w   #1,dx ;az x irányú sebességet 1-re változtatjuk
    mov.w   #1,dy ;az y irányú sebességet 1-re változtatjuk
Ciklus:
    mov.w   y,R11   ;a labda y koordinátáját beletesszük, egy szabad regiszterbe
    add.w   dy,R11  ;a megváltozást hozzáadjuk ehhez az értékhez
    cmp.w   #40,R11 ;megvizsgáljuk, hogy y irányba lefelé kimegyünk-e a képernyőből (lehetséges y>=40)
    jge     Cimke001 ;ha kimennénk, akkor a Cimke001-et hívjuk meg
    cmp.w   #-1,R11 ;megvizsgáluk, hogy y irányba felfelé kimegyünk-e a képből (lehetséges y>=-1)
    jge     Cimke002 ;ha benne vagyunk a képbe akkor a Cimke002-őt hívjuk meg
Cimke001:
    xor #0xffff,dy;akár felfele, akár lefele mennénk ki a képernyőből negáljuk a sebességet
                  ;vagy FFFE-t kapunk, vagy 0000-át
    inc dy        ;majd megnöveljuk eggyel, tehát a túlcsordulás miatt, ha ezt hozzáadjuk egy számhoz, akkor az olyan,
                  ;mintha egyet kivonnánk, mivel ez a maximum (FFFF),
                  ;vagy ha FFFF volt, akkor 0001-et kapunk és ezzel növelünk
Cimke002:
    add.w   dy,y    ;tudjuk, hogy érvényes helyre akarunk lépni, oda is lépünk (y+/-1)
    mov.w   x,R11   ;az aktuális x koordinátát belemásoljuk egy szabad regiszterbe
    add.w   dx,R11  ;majd hozzáadjuk a megváltozást
    cmp.w   #77,R11 ;megvizsgáljuk, hogy x irányba jobbra kimegyünk-e a képernyőből (lehetséges x>=77)
    jge     Cimke003 ;ha kimennénk, akkor a Cimke003-et hívjuk meg
    cmp.w   #-1,R11  ;megvizsgáluk, hogy x irányba balra kimegyünk-e a képből (lehetséges x>=-1)
    jge     Cimke004 ;ha benne vagyunk a képbe akkor a Cimke004-et hívjuk meg
Cimke003:
    xor #0xffff,dx ;akár jobbra, akár balra mennénk ki a képernyőből negáljuk a sebességet
                  ;vagy FFFE-t kapunk, vagy 0000-át
    inc dx        ;majd megnöveljuk eggyel, tehát a túlcsordulás miatt, ha ezt hozzáadjuk egy számhoz, akkor az olyan,
                  ;mintha egyet kivonnánk, mivel ez a maximum (FFFF),
                  ;vagy ha FFFF volt, akkor 0001-et kapunk és ezzel növelünk
Cimke004:
    add.w   dx,x      ;tudjuk, hogy érvényes helyre akarunk lépni, oda is lépünk (x+/-1)
    mov.b   y,R13     ;a 13-as regiszterbe belemásoljuk az aktuális y koordinátát (a Labda függvény erre hívódik meg)
    mov.b   x,R12     ;a 12-as regiszterbe belemásoljuk az aktuális x koordinátát (a Labda függvény erre hívódik meg)
    call  #LCDLabdaXY ;a labda kirajzolásának függvénye
    mov.b   y,R12     ;itt változtattunk, mivel az y magasságába kell kiiratni az ütőt, nem az x szerint
    call  #LCDUtoLXY  ;a baloldali ütő kirajzolásának függvénye
    mov.b   #0,R12    ;itt változtattunk, mivel tükrözni kellett a mozgást, tehát nem 40-ből vontunk ki, hanem 0-hoz
                      ;adtunk hozzá
    add.b   y,R12     ;itt is változtattunk, kivonás helyett összeadás
    call  #LCDUtoRXY  ;a jobboldali ütőt kirajzoló függvény
    call  #LCDUpdate  ;a képernyő frissítése
   jmp    Ciklus      ;a kezdő ciklusra való ugrás
        ret
```

1. ábra. A program általunk kommentált és javított változata

HIVATKOZÁSOK

[1] Mikrokontroller dokumentáció, elérhető: http://www.ti.com/lit/ug/slau049f/slau049f.pdf