Mikrokontroller II. mérés

Mérést végezte: Paulicsek Ádám Imre
Mérés dátuma: 2021.03.18. 11:15-14:00
Mérés helye: Budapest
paulicsek.adam.imre@hallgato.ppke.hu

Mérőeszköz adatai: IAR Embedded Workbench software

I. Mikrokontroller

A mikrokontroller vagy mikrovezérlő egyetlen lapkára integrált, általában vezérlési feladatokra optimalizált célszámítógép. A mikrokontroller egy mikroprocesszor kiegészítve az áramköri lapkájára integrált perifériákkal. Manapság sok hétköznapi használati eszközben mikrokontroller lapul a digitális hőmérőtől az autónkon át akár a gyorséttermi ajándék játékig.

Régebben mikroprocesszor-típusokat használtak a vezérlési feladatok elvégzésére. A mikroprocesszor használatakor a szükséges perifériák miatt további integrált áramköröket (IC) kellett beépíteni. Az áramköri technológia fejlődésével egyre több perifériát az IC-tokba lehetett integrálni, így alakult ki a mikrokontroller, nagyon tömör áramkört eredményezve.

II. Mérési feladatok

A. Nulladik feladat

Ebben a feladatban az volt a lényeg, hogy ha a két regiszterben eltárolt érték megegyezett, akkor az R15-ös regiszter értéke legyen egyenlő a feladat sorszámával.

```
//0. feladat
    mov #5,R4
    mov #5,R5
    cmp R4,R5
    jnz NoAction
    mov #0,R15
    NoAction:
```

Mint mindig, itt is a mov segítségével rendelek egy tetszőleges értéket, a regiszterhez. Először az R4-hez, aztán az R5-höz rendeltem hozzá az 5-ös értéket. Ezekután összehasonlítom a két értéket a cmp, és a jnz segítségével, ami azt jelenti, hogy jump if nonzero, ami annyit jelent, hogy a Z flag bitet nézi meg, hogy milyen értékkkel rendelkezik az összehasonlítás után, és ha 0, vagyis nincs használva, akkor kiugrik a 'ciklusból'. Ebben az esetben a Z flat bit értéke egy lesz, vagyis nem ugrik ki egyből, hanem amit megadunk neki az egyenlőség után, az lefut még le. Ebben az esetben az R15 regiszterhez rendelem hozzá a 0-ás értéket.

```
//0. feladat
    mov #5,R4
    mov #6,R5
    cmp R4,R5
    jnz NoAction
    mov #0,R15
    NoAction:
```

Ellenkező esetben, amikor a Z flat big értéke nem 1 lesz, amikor nem változik meg, és marad 0, akkor egyből kiugrik. És folytatodik tovább, amit a NoAction: után van.

```
//0. feladat
    mov #-5,R4
    mov #-5,R5
    cmp R4,R5
    jnz NoAction
    mov #0,R15
    NoAction:
```

Amikor előjeles biteket használok, akkor ugyanazt tapasztalom, mint amikor nem előjeles biteket használok. Viszont, ha olyan mínuszos értékeket adok meg, akkor az N flag bit értéke megváltozik 1-re.

B. Első feladat

Ebben a felatban az előző feladat tagadását kell megvalósítani, amikor a két regiszter nem egyezik meg. Ezt több féleképpen meg lehet valósítani.

```
//1. feladat
    mov #5,R4
    mov #4,R5
    cmp R4,R5
    jz NoAction
    mov #1,R15
    NoAction:
```

Az előző tagadása szerintem egyik legegyszerűbb módja, hogyha nem a jnz használom itt, hanem a jz. Ez az, amikor az értéke 1, akkor ugor ki, azért is nevezik jump if zero. Ha a regiszter értékei különbözőek, mint ebben az esetben az egyik 5, a másiknak pedig 4, ekkor a Z bit flag értéke 0 marad, akkor lefordul az utána lévő sor, ahol az R15-ös regiszter értéke 1 lesz.

```
//1. feladat
    mov #5,R4
    mov #5,R5
    cmp R4,R5
    jz NoAction
    mov #1,R15
    NoAction:
```

Ellenkező esetben, amikor a két regiszter értéke megegyezik, akkor a Z flat bit értéke 1 lesz, ekkor kiugrunk, és ami a NoAction: előtt van, azok nem fordulnak már le.

```
//1. feladat
    mov #5,R4
    mov #4,R5
    cmp R4,R5
    jz NoAction
    mov #1,R15
    NoAction:
```

Abban az esetbe, amikor előjeles bitekkel dolgozok, ugyanazokat tapasztalom, mint az előző feladatban, amikor előjeles bitet használtam.

C. Második feladat

Ebben a feladatrészben azt vizsgálom meg, hogy az R4-es regiszter szigorúan nagyobb az R5-ös regisztertől, akkor az R15-ös értéke legyen a feladat sorszámával megegyező.

```
//2. feladat
    mov #5,R4
    mov #4,R5
    cmp R4,R5
    jge NoAction
    mov #2,R15
    NoAction:
```

Mint az előzőekben, itt is a mov segítségével megadom a regisztereknek az értékeit, de az R4-es ebben az esetben nagyobb. A legegyszerűbb megoldás az, ha a jge használnám, ami azt jelenti, ha nagyobb, akkor lefut. Ekkor ha lefut és iagz, akkor az R15-ös regiszterhet hozzárendelem a 2-es értéket.

```
//2. feladat
    mov #5,R4
    mov #5,R5
    cmp R4,R5
    jge NoAction
    mov #2,R15
    NoAction:
```

Abban az esetben, amikor a két érték megegyezik, nem lesz igaz, ezért nem fog lefutni az R15-ös rész, hanem egyből kiugor.

```
//2. feladat
    mov #+6,R4
    mov #+5,R5
    cmp R4,R5
    jge NoAction
    mov #2,R15
    NoAction:
```

Nem tapasztalom az előekhez képest mást változást, mint amit eddig tapasztaltam.

D. Harmadik feladat

Az előző feladatnak a tagadása, mivel pont az ellentetje kelleni fog nekünk. Amikor az R4 regiszter helyett, az R5 regiszterben lesz szigorúan nagyobb érték, akkor legyen az R15-ös 3.

```
//3. feladat
    mov #4,R4
    mov #5,R5
    cmp R4,R5
    jl NoAction
    cmp R4,R5
    jz NoAction
    mov #3,R15
    NoAction:
```

Ahogy az előző feladatban szerettük volna a nagyobb-at használni, itt is a legegyszerűbb az lenne, és a szerencsénkre pont van ilyen, ami nem más mint a jl. Viszont ebből nincsen jle, ami majd a elkövetkező feladatokban lesz majd probléma. Mint az előzőekben ugyanúgy csinálok, mindent és mikor szigorúan kisebb az R4-es regiszter az R5-östől, akkor az R15 értéke megváltozik.

```
//3. feladat
    mov #5,R4
    mov #5,R5
    cmp R4,R5
    jl NoAction
    cmp R4,R5
    jz NoAction
    mov #3,R15
    NoAction:
```

Amikor a két érték megegyezik, akkor ugyanúgy lefutna ha csak az jl lenne benne, ezért még emellé oda kell írni egy másik elágazás, amit az első feladatban csináltunk, és akkor teljesen jó lesz, és csak akkor lesz az R15-ös értéke 3.

```
//3. feladat
    mov #+4,R4
    mov #+5,R5
    cmp R4,R5
    jl NoAction
    cmp R4,R5
    jz NoAction
    mov #3,R15
    NoAction:
```

Előjelesen megint nem tapasztalok semmi új dolgot, amit eddig nem tapasztaltam volna az előzőekben.

E. Negyedik feladat

Ebben a feladatrészben az R4-es regiszternek kell nagyobbnak lenni, mint az R5-ös. Viszont ebben az esetben nem szigorúan, hanem megegyezhet a két érték.

```
//4. feladat
    mov #5,R4
    mov #4,R5
    cmp R4,R5
    jge NoAction
    mov #4,R15
    NoAction:
    cmp R4,R5
    jnz NoAction1
    mov #4,R15
    NoAction1:
```

Ebben a részben, a jge segítségével elértem, hogy a nagyobb számokra jók legyenek. Ha kisebb lesz mint az R5, akkor nem fordul le az a rész, amikor az R15-nek az értéke 4 lenne.

```
//4. feladat
    mov #4,R4
    mov #4,R5
    cmp R4,R5
    jge NoAction
    mov #4,R15
    NoAction:
    cmp R4,R5
    jnz NoAction1
    mov #4,R15
    NoAction1:
```

Hogy amikor egyenlő a két szám, akkor is forduljon le, ezért a jnz kellett használni egy vagy kapcsolattal, és így teljes mértékben jól működik ez a rész is.

```
//4. feladat
    mov #-4,R4
    mov #-4,R5
    cmp R4,R5
    jge NoAction
    mov #4,R15
    NoAction:
    cmp R4,R5
    jnz NoAction1
    mov #4,R15
    NoAction1:
```

Előjeles bitekkel megint az tapasztalható, mint eddig. Ugyanúgy működik, mint az előjel nélküli.

F. Ötödik feladat

Az előző feladathoz hasonlít, csak itt az nem nagyobb vagy egyenlő kell, hanem kisebb, de abból is nem a szigorú, hanem megengedjük neki, hogy azonos érték esetén is leforduljon.

```
//5. feladat
    mov #5,R4
    mov #6,R5
    cmp R4,R5
    jl NoAction
    mov #5,R15
    NoAction:
```

Az jl segítségével tudjom elérni, hogy az R5 értéke nagyobb legyen az R4-nél. És ha ez igaz, akkor lefordul az a rész, amikor az R15-ösnek adok értéket, mégpedig 5-öt.

```
//5. feladat
    mov #5,R4
    mov #5,R5
    cmp R4,R5
    jl NoAction
    mov #5,R15
    NoAction:
```

Ebben a feladatrészben elég nekem az jl, és itt nem kell semmit sem korrigálni, mivel ekkor az egyenlőekre is ugyanúgy lefordul az a rész, mint amikor kisebb az R4, az R5-nél.

```
//5. feladat
    mov #+5,R4
    mov #+5,R5
    cmp R4,R5
    jl NoAction
    mov #5,R15
    NoAction:
```

Előjeles bit használatánál, nem tapasztalunk semmi eltérőt, mint eddig volt.

G. Hatodik feladat

Itt az R4-nek, és az R5-nek egyenlőnek kell lenni, és emelett az R6-nak kisebbnek kell lenni, mint az R7nek. Mindkettőnek teljesülnie kell, mert a két kifejezés között és kapcsolat van, vagyis ha már az egyiknél elbukik, akkor biztosan nem lesz igaz a kiértékelés.

```
//6. feladat
    mov #5,R4
    mov #5,R5
    mov #5,R6
    mov #6,R7
    cmp R4,R5
    jnz NoAction
    cmp R6,R7
    jl NoAction
    cmp R6,R7
    jz NoAction
    mov #6,R15
    NoAction:
```

Ebben a feladatban már nem csak két regiszterben tárolok értékeket, hanem négyben. Először azt nézzem meg a jnz segítségével, hogy az R4, és az R5 értéke megegyezik-e, ha igen, akkor nézem a többi 'feltételt', ha nem igaz, akkor kiugrok belőle. A következő az R6, és az R7 regiszter között nézem meg a szigorúan nagyobb-e az R6-nál, az R7. Ezt a jl és a jz segítségével érem el, ugyanúgy mint a harmadik feladatnál csináltam, ha ez a kettőre igaz lesz, akkor fordul le az a része a kódnak, amikor az R15-ös regiszterbe a 6-os értéket rendelem.

```
//6. feladat
    mov #-5,R4
    mov #-5,R5
    mov #+5,R6
    mov #+6,R7
    cmp R4,R5
    jnz NoAction
    cmp R6,R7
    jl NoAction
    cmp R6,R7
    jz NoAction
    mov #6,R15
    NoAction:
```

Előjelesnél megint nem tapasztalható semmi változás, mint az előttiekben.

H. Hetedik feladat

Hasonló mint az előző feladat, de itt nem az R6 kisebb, az R7-nél, hanem fordítva lesz, vagyis az R6-os regiszternek kell nagyobb lennie, mint az R7-es regiszterben lévő.

```
//7. feladat
    mov #5,R4
    mov #5,R5
    mov #7,R6
    mov #6,R7
    cmp R4,R5
    jnz NoAction
    cmp R6,R7
    jge NoAction
    mov #7,R15
    NoAction:
```

Az első fele a feladatnak teljesen ugyanaz, mint az előző feladatnak. Viszont mivel itt pont fordítva van a relációsjel, hogy az R6 az nagyobb-e, mint az R7, és akkor teljesül. A második rész, amiben eltér, hogy az R6, és az R7 összehasonlításánál, a jge-t használom, ami szigorúan nagyobbat jelent.

```
//7. feladat
    mov #4,R4
    mov #5,R5
    mov #7,R6
```

```
mov #6,R7
cmp R4,R5
jnz NoAction
cmp R6,R7
jge NoAction
mov #7,R15
NoAction:
```

Mivel és kapcsolat van a kettő kifejezés között, ezért már ha az első elbukik, akkor a második része, már le se fordul, és egyből kiugrik az egészből. Ebben az esetben is ez látható, mivel az első két regiszter különbözk.

```
//7. feladat
    mov #+5,R4
    mov #+5,R5
    mov #+7,R6
    mov #+6,R7
    cmp R4,R5
    jnz NoAction
    cmp R6,R7
    jge NoAction
    mov #7,R15
    NoAction:
```

Előjeles bitnél megint nem tapasztalhatunk semmi féle változást, az előbbiekhez képest.

I. Nyolcadik feladat

Az előző két feladattól, attól tér el ez, hogy itt majd nem és kapcsolat lesz, hanem vagy. Vagyis azzal változik minden, hogy elég csak az egyik kifejezésnek igaz lennie, és akkor R15-nek értéket fogok adni.

```
//8. feladat
    mov #5,R4
    mov #5,R5
    mov #5,R6
    mov #6,R7
    cmp R4,R5
    jnz NoAction
    mov #8,R15
    NoAction:
    cmp R6,R7
    jl NoAction1
    cmp R6,R7
    jz NoAction1
    mov #8,R15
    NoAction1:
```

Kódban nincsen sok változás a hatos feladathoz képest. Csak annyi változás van, hogy itt vagy kapcsolat van és helyett, vagyis az első elágazás teljesülés esetében az R15-ösnek az értékét megváltoztatom, és már mindegy hogy a második részben mi fog történni, elég csak az egyiknek teljesülnie, nem muszáj mindkét kifejezés teljesülése.

```
//8. feladat
    mov #4,R4
    mov #5,R5
    mov #5,R6
    mov #6,R7
    cmp R4,R5
    jnz NoAction
    mov #8,R15
    NoAction:
    cmp R6,R7
    jl NoAction1
    cmp R6,R7
    jz NoAction1
    mov #8,R15
    NoAction1:
```

Mint ebben az esettben, nem fordul le az első R15-ös regiszteres rész, viszont a második része igaz, ezért az le fog forduni, és át fogja adni az értéket az R15-ös regiszternek.

```
//8. feladat
    mov #+5,R4
    mov #+5,R5
    mov #-5,R6
    mov #-6,R7
    cmp R4,R5
    jnz NoAction
    mov #8,R15
    NoAction:
    cmp R6,R7
    jl NoAction1
    cmp R6,R7
    jz NoAction1
    mov #8,R15
    NoAction1:
```

Ebben az esetben is az előjel bitekkel, nem történik semmi új változás, amit leírhatnánk.

J. Kilencedik feladat

Az előző feladathoz képest, ebben csak a második kifejezés fog különbözni, hogy R6-os nagyobbnak kell lenni, mint az R7-esnek.

```
//9. feladat
    mov #5,R4
    mov #5,R5
    mov #6,R6
    mov #5,R7
    cmp R4,R5
    jnz NoAction
    mov #9,R15
    NoAction:
    cmp R6,R7
    jge NoAction1
    mov #9,R15
    NoAction1:
```

Itt is, mint a hetedik feladatban, ugyanúgy csinálom majdnem mindent, kivéve hogy a két kifejezés közt nem és, hanem vagy kapcsolat van. Ezt úgy orvosoljuk, hogy az első után teszek egy mov-ot, hogyha az igaz, akkor átadom neki az értéket, mivel a szükséges feltétel, hogy az egyik lefordul.

```
//9. feladat
    mov #4,R4
    mov #5,R5
    mov #6,R6
    mov #5,R7
    cmp R4,R5
    jnz NoAction
    mov #9,R15
    NoAction:
```

```
cmp R6,R7
jge NoAction1
mov #9,R15
NoAction1:
```

Ebben az esetben is látszik, hogy az első feltétel nem lesz igaz, ezért nem egyből kiugrik, hanem nézi tovább, és az utolsó igaz lesz, és R15-be beleteszi a feladat számát.

```
//9. feladat
    mov #+5,R4
    mov #+5,R5
    mov #+6,R6
    mov #+5,R7
    cmp R4,R5
    jnz NoAction
    mov #9,R15
    NoAction:
    cmp R6,R7
    jge NoAction1
    mov #9,R15
    NoAction1:
```

Mint minden esetben, itt se figyelhető meg, a többi előjeles biteshez képest változást.

K. Tizedik feladat

Ebben a feladatrészben két regisztert kell elosztani egymással, és ha oszthatóak, vagyis a maradéka az osztás után 0, akkor eloszthtató a két szám.

```
//10. feladat
    mov #8,R4
    mov #4,R5
    mov R4,R12
    mov R5,R14
    call #divide
    mov R5,R14
    call #multiply
    cmp R4,R12
    jnz NoAction
    mov #10,R15
    NoAction:
```

Ebben az esetben, már a szorzás, és az osztás már rendelkezésemre áll, nem kell nekem megírni. Meg van, hogy adott regiszterekbe kell betöltenem a számokat, mert arra van megírva. Először elosztom a két számot, és ahányszor el tudtam osztani, az lesz az eredménye. Ezután ezt az eredményt beszorzom a kisebbik számmmal, és megnézem a jnz segítségével, hogy amit kaptam eredmény megegyezik-e az R4-essel, mert ha igen, akkor maradék nélkül el tudtam osztanni. És az R15 regiszter értéke 10 lesz.

```
//10. feladat
    mov #8,R4
    mov #5,R5
    mov R4,R12
    mov R5,R14
    call #divide
    mov R5,R14
    call #multiply
    cmp R4,R12
    jnz NoAction
    mov #10,R15
    NoAction:
```

Ebben az esetben, nem fog teljesülni, mivel 8 osztva 5-tel az egyszer van meg, és ha ezt beszorzom 5-tel, akkor csak 5-öt fogok kapni, és az nem lesz egyenlő a 8-cal. Vagyis nem osztható el egészesen a szám.

```
//10. feladat
    mov #+8,R4
    mov #-4,R5
    mov R4,R12
    mov R5,R14
    call #divide
    mov R5,R14
    call #multiply
    cmp R4,R12
    jnz NoAction
    mov #10,R15
    NoAction:
```

Ebben az esetben, ha előjelesen csinálom meg mindezt, akkor bajban leszek, mivel egy pozitív számot osztok el egy negatívval, annak az eredménye negatív lesz. Azt beszorozva 4-gyel, megkapom hogy -8, vagyis az ellentetjét, és ugyanúgy nem fog lefutni.

L. Tizenegyedik feladat

Ebben a feladatrészben, két számnak a szorzatának az eredményét vizsgálom meg, hogy páros-e, és ha ez igaz, akkor az R15-ös regiszterbe kerül a feladatsorszáma.

```
//11. feladat
    mov #2,R4
    mov #5,R5
    mov R4,R12
    mov R5,R14
    call #multiply
    mov R12,R6
    mov #2,R14
    call #divide
    mov #2,R14
    call #multiply
    cmp R6,R12
    jnz NoAction
    mov #11,R15
    NoAction:
```

Először a két számot összeszorzom, és ennek az eredménye az R12-es regiszterbe kerül, amit majd tudunk tovább vinni, viszont el kell tárolni a benne lévő eredményt, mert majd a késiekben még kelleni fog. Aztán mint az előző feladatban, ugyanazt csinálom meg, hogy megvizsgálom, hogy 2-vel egészes maradékot kapok, és ha igen akkor páros szám, a két szám szorzata.

```
//11. feladat
    mov #3,R4
    mov #5,R5
    mov R4,R12
    mov R5,R14
    call #multiply
    mov R12,R6
    mov #2,R14
    call #divide
    mov #2,R14
    call #multiply
```

```
        cmp R6,R12
        jnz NoAction
        mov #11,R15
        NoAction:
```

Ebben az esetben látszik, hogy nem lesz a szorzat párosszám, hanem páratlan lesz. Ebben az esetben az R15-ös értéke nem lesz 11.

```
    //11.feladat
        mov #-2,R4
        mov #+5,R5
        mov R4,R12
        mov R5,R14
        call #multiply
        mov R12,R6
        mov #+2,R14
        call #divide
        mov #+2,R14
        call #multiply
        cmp R6,R12
        jnz NoAction
        mov #11,R15
        NoAction:
```

Amikor előjeles bitekkel csináljuk, akkor azt kell tapasztalni, hogy nem fog működni fog, mint az előző esetben, pontosan ugyanazért mint ott történt, hogy negatív szám, nem egyenlő a pozitívjával. Egy abszolút érték vonással, ez a hiba kiküszöbölhető lenne.

REFERENCES

[1] Mikrokontroller, https://hu.wikipedia.org/wiki/Mikrovez%C3%A9rl%C5%91