Mikrokontroller mérés

Mérést végezte: Paulicsek Ádám Imre
Mérés dátuma: 2021.03.11. 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. Első feladat

(Végezzen el összeadást két 8 bites előjel nélküli szám között.)

```
//1. feladat a)
    mov.b #6,R4
    mov.b #2,R5
    add.b R5,R4
```

Először létrehoztam egy R4-es nevű, 8 bites regisztert, aminek az értéke 6 lesz. Ezt úgy csináltam, hogy mov.b segítségével rendeltem a konstant, a regiszterhez. A move utáni .b jelenti, hogy a regiszter milyen hosszúságú, ami ebben az esetben 8. Ezután egy R5-os regiszterbe teszem a 2-es értéket. Futtatás közben látható a Registers ablakban, hogy az adott regiszterben milyen érték van, és ezek az értékek hexidecimálisan vannak megadva, amit onnan tudunk, hogy 0x kezdődnek a számok. Aztán a két regisztert az add.b segítségével adom össze. Fontos hogy két azonos hosszúságú regisztert adjunk össze, és akkor azonos hosszú végeredményt fogunk kapni. Az összeadott értékeket, a másodikként megadott regiszterbe fog kerülni, ami ebben az esetben az R4-es volt, annak az értékét írjuk felül, míg az R5-ösnek nem változik meg az értéke ebben az esetben, és továbbra marad a 2-es szám, míg fel nem írjuk utána, vagy nullázuk le.

```
//1. feldat b)
    mov.b #255,R4
    mov.b #17,R5
    add.b R5,R4
```

Tudjuk, hogy a 8 bites sorozatnál 0 ... 255 közötti számok lehet, mivel 2^8 darab ábrázolható szám van, és a legnagyobb ábrázolható szám pedig $2^8 - 1$. Ebben az esetben megnézzem, hogy mi történik, ha a 255 értékhez adok hozzá egy másikat, ami túlcsordulást fog eredményezni. Először az R4-hez rendelem hozzá a 255 értéket, ugyanúgy mint az előző feladatrészben, aztán az R5-höz a 17-et. Ezeket összeadva, ugyanúgy mint az előbb, az R4-be kerül az összeg, de kevesebb lett, mint 255. Ekkor a regiszter ablakban megfigyelhető a flag biteknél, hogy a carry bitnek az értéke 1 lett, vagyis túlcsordulást történt. Ekkor tudhatjuk, hogy pontosan miért is mutat 255-nél kevesebbet az összeadás után. Az R5 értéke viszont megint változatlan maradt.

B. Második feladat

(Végezzen el összeadást két 16 bites előjel nélküli szám között. A művelet elvégzése során vizsgálja a carry bit értékét.)

```
//2. feladat a)
    mov.w #155,R4
    mov.w #132,R5
    add.w R5,R4
```

Hasonló a feladat, mint az előzőben, de itt nem 8 bites, hanem 16 bites sorozatokat kell összeadni. Vagyis annyi különbség lesz a kódban, hogy nem .b, hanem .w fogok használni. Ugyanazt fogom tapasztalni, mint az előzőben, hogy a két regiszter összeadásakor, az R4-be fog kerülni az összeg, és az R5-ösnek nem fogja megváltoztatni az értékét.

```
//2. feladat b)
    mov.w #65535,R4
    mov.w #132,R5
    add.w R5,R4
```

Ebben az esetben is megnézem, hogy mi történik, ha túlcsordul az összeg. Itt az előzőhöz képest, ugyanazt a kódot fogom használni, de az értéke más lesz. Az ábrázolható számok száma 16 biten 2^{16}, a legnagyobb ábrázolható szám $2^{16} - 1$, vagyis 65535. Ekkor ugyanazt fogom tapasztalni a regiszter ablakban, hogy a carry flag bit értéke megválozik 1-re, ami jelzi ugyanúgy a túlcsordulást.

C. Harmadik feladat

(Végezzen el összeadást két 32 bites előjel nélküli szám között. A művelet elvégzése során vizsgálja a carry bit értékét)

```
//3. feladat
    mov.w #0x12AE,R4
    mov.w #0x3232,R5
    mov.w #0x4321,R6
    mov.w #0x1A1E,R7
    add.w R4,R5
    addc.w R6,R7
```

Mivel a mikrokontroller a regisztereibe maximálisan 16 bites soroztot tudd tárolni, ezért a 32 bites sorozatot két 16-os bit sotozatként fogjom kezelni, az egyikbe lesz a

kisebb helyiértékű sorozat, és a másikban lesz a magasabb helyiértékű sorozat. Mint az összeadásnál is, mindig a kisebb helyiértékeket adjuk először össze, hogy a túlcsordulást tovább tudjuk majd vinni. Aztán a magasabbat adjuk össze, és ehhez az előző művelet esetén keletkező túlcsordulást. Ezért a másodikat addc.w paranccsal fogom elvégezni. És így kaptam két darab 16-os bit sorozatot, aminek a kisebb helyiértékűt az R5-be, míg a nagyobbikat pedig a R7-be fog belekerülni. A többi regiszter értéke nem fog megváltozik.

D. Negyedik feladat

(A tanultakat ellenőrizze az 1;2;3; feladat megoldásával előjeles környezetben is.)

```
//4. feladat a)
    mov.b #+6,R4
    mov.b #+2,R5
    add.b R5,R4
```

Az előjeles összeadásnál hasonlóan történik, mint előtte előjel nélkül. Viszont mivel a legmagasabb helyiérték helyén most már az előjelet fogja jelölni, ezért a legnagyobb ábrázolható szám kevesebb lesz, viszont az ábrázolható számok majdnem a duplájára nőnek. Ebben az esetben $2^7 - 1$ lesz a legnagyobb ábrázolható szám, vagyis a 127. De akkor a legkisebb szám viszont nem a 0, hanem a -127 lesz. Ugyanúgy mint az első feladatban, a két szám összege az R4-be fog kerülni, és az R5-ös értéke nem fog megváltozni.

```
//4. feldat b)
    mov.b #+127,R4
    mov.b #+17,R5
    add.b R5,R4
```

Ebben az esetben nem a carry flag bitnek az értéke fog megváltozni, hanem az overflow flag bitnek az értéke lesz 1. Az összeadott szám ugyaúgy az R4-ben lesz, és az R5 megintváltozatlan lesz.

```
//4. feladat c)
    mov.w #+155,R4
    mov.w #+132,R5
    add.w R5,R4
```

Ugyanaz történik, mint az előjel nélküli összeadásnál. És itt is, mint a 8 biteshez hasonlóan 1 bittel kevesebb számot tudunk ábrázolni az előjel miatt. A legnagyobb ábrázolható szám $2^{15} - 1$, vagyis 32767 lesz.

```
//4. feladat d)
    mov.w #+32767,R4
    mov.w #+150,R5
    add.w R5,R4
```

Ebben az esetben, az overflow értéke 1 lesz, és emellett a negatív flag bit értéke is 1 lesz, mivel túlcsordulás történt. Az eredmény az R4-es regiszterbe kerül.

```
//4. feladat e)
mov.w #0xEFFF,R4
mov.w #0x0017,R5
mov.w #0xFFFE,R6
mov.w #0x0005,R7
add.w R5,R4
addc.w R7,R6
```

Ugyanúgy mint az előzőekben, ezt a 32 bites sorozatot is két részre bontjuk, és így adogatjuk össze, de itt ADDC-t használunk, aztán a végeredmény két regiszterbe kerül. Amikor az alsó helyiértékeket adom össze, akkkor a negatív flag bit értéke 1 lesz, míg amikor a felső helyiértékek szerint adom össze, akkor a carry flag bit értéke változik 1-re.

E. Ötödik feladat

(Végezzen el kivonást két 8 bites előjel nélküli szám között.)

```
//5. feladat a)
    mov.b #17,R4
    mov.b #7,R5
    sub.b R5,R4
```

A kivonáshoz a sub-ot fogom használni, másképpen mondhatjuk, hogy ez az összeadás műveletének az inverze. Ebben az esetben, a 17-ből fogjuk kivonni a 7-est. Ennek az értékét az R4-es regiszterbe fog kerülni, míg az R5-ösben marad a 7-es érték. Emellett a carry flag bit értéke 1 lesz, vagyis a borrow bit értéke az ellentetje, vagyis No.

```
//5. feladat b)
    mov.b #7,R4
    mov.b #17,R5
    sub.b R5,R4
```

Ebben az esetben felcseréltem az értékeket, és most azt vizsgálom meg, hogy mi történik amikor mínuszos számot kapnék, miközben nem használjuk az előjel bitet, amikor ilyen esetekben nagyon jól jönne. Viszont az történik, hogy a negatív flag bitnek az értéke fog megváltozni 1-re, és azt vehetjük még észre, hogy amikor kilépünk az értelmezett intervallumból, akkor kezdődik újra a legnagyobb ábrázolható számtól. Ekkor a carry flag bit értéke nem véltozik, marad 0. Ekkor az ellentetje a borrow bit értéke Yes lesz.

F. Hatodik feladat

```
//6. feladat a)
    mov.w #155,R4
    mov.w #132,R5
    sub.w R5,R4
```

Ebben az esetben is azt tapasztalhatjuk mint előbb, amikor nem lépünk ki az intervallumból. Viszont amikor elvégeztük a műveletet, akkor a carry flag bit értéke 1 lett, ami azt jelenti hogy a borrow bit értéke 0 lesz.

```
//6. feladat b)
    mov.w #132,R4
    mov.w #155,R5
    sub.w R5,R4
```

Ugyanaz történik, mint a 8 bites sorozatoknál, viszont itt nem lesz carry bit, hanem csak negatív flag bit. És az értéke ugyanúgy a legmagasabban ábrázolható számtól fog kezdődni, amikor elhagyjuk az intervallumot. És mivel a carry bit értéke 0, ezért a borrow bit értéke 1 lesz.

G. Hetedik feladat

(Végezzen el kivonást két 32 bites előjel nélküli szám között. A művelet elvégzése során vizsgálja a borrow bit értékét.)

```
//7.feladat
    mov.w #0x0004,R4
    mov.w #0x0002,R5
    mov.w #0x0000,R6
    mov.w #0x0008,R7
    sub.w R5,R4
    subc.w R7,R6
```

Mind az előző 32 bites esetben itt is a végeredményt két regiszterbe fogom tudni eltárolni. Ebben az esetben is ADDC-t értékként megadni őket. Az alsóhelyiértéknél carry flag bit értéke 1, ezért a borrow bit értéke 0. Míg a felsőhelyiértéknél a negatív flag bit értéke lesz 1, és a carry flag bit értéke 0, vagyis a borrow bit értéke ekkor 1 lesz.

H. Nyolcadik feladat

(A tanultakat ellenőrizze az 5;6;7; feladat megoldásával előjeles környezetben is.)

```
//8.feladat a)
    mov.b #-128,R4
    mov.b #+17,R5
    sub.b R5,R4
```

Ebben az esetben az overflow flag bit értéke 1 lesz, és a carry bit értéke is szintén 1 lesz. Vagyis a borrow bitnek az értéke 0.

```
//8.feladat b)
    mov.w #+23, R4
    mov.w #-156, R5
    sub.w R5,R4
```

Ebben az esetben a carry bit értéke 0 lesz, és a borrow bit értéke viszont 1 lesz. Az eredmény az R4-es regiszterbe kerül, a többi regiszter változatlan marad.

```
//8.feladat b)
    mov.w #-365,R4
    mov.w #-210,R5
    mov.w #-241,R6
    mov.w #-444,R7
    sub.w R5,R4
    subc.w R7,R6
```

Ebben az esetben is a 32 bites sorozatot két részre szedtem szét. A két résznek az eredménye az R4-es, és az R6-os regiszterbe fog kerülni. A carry bit értéke 1, ezzel szemben a borrow bit értéke 0 lesz.

REFERENCES

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