Mikrokontroller I. jegyzőkönyv

**Mérést végző személyek**: Hadnagy Levente, Ekart Csaba

**Mérés helye**: PPKE ITK 420 mérőlabor

**Mérés ideje**: 2017.04.27. 12:15-15:00

**Méréshez felhasznált eszközök**: MSP430 mikrokontroller, IAR Embedded Workbench programcsomag

# ***A mérés ismertetése***

A mikrokontroller csatlakoztatva volt a PC-nkhez, így először üzembe helyeztük az IAR Embedded Workbench fejlesztői környezetet, és az utasítások szerint létrehoztunk egy új projectet. A projektben a programkód megfelelő részében oldottuk meg a feladatokat, ellenőrzésképpen Debug módban futtattuk, ahol a Registry ablakban követtük az egyes regiszterek értékeit. A számolások eredményét először papíron is ellenőriztük, az adatok értelmezéséhez felhasználtuk a bit flagekről való rendelkezésünkre álló információkat.

# ***A mérési feladatok megoldása***

1. **8 bites összeadás előjel nélkül**

 ;8bites, előjel nélküli összeadás

 mov.b #8**,**R4

 mov.b #17**,**R5

 add.b R5**,**R4

Az R4-es regiszterbe beletesszük a 8-at, a R5-ösbe a 17-et a mov.b utasítással, majd az add.b paranccsal összeadjuk őket, és az eredmény az R4-es regiszterbe kerül. A műveletek végén a .b a byte-ot jelenti, e segítségével végezhetjük a műveletet 8 bites értékeken.

 ;8bites, előjel nélküli összeadás példa a túlcsordulásra

 mov.b #255**,**R6

 mov.b #1**,**R7

 add.b R7**,**R6

Tudhatjuk, hogy a 8 biten tárolható legmagasabb érték a $2^{8}-1=255$, ezért felmerül a kérdés, hogy mi történik ha az érték túlcsordul a tartományon, ezért betöltöttük az R6-os és az R7-es regiszterbe a 255, illetve az 1 értékeket, és a kettőt összeadtuk. Ezek összeadása 0-lesz a túlcsordulás miatt. Ezt tudhatjuk a carry bit 1-es értékéből is.

1. **16 bites összeadás előjel nélkül**

 ;16 bites előjel nélküli összeadás

 mov.w #130**,**R6

 mov.w #150**,**R7

 add.w R7**,**R6

A program az előzőhöz hasonlóan működik azzal a különbséggel, hogy a mov.w, illetve az ad.w parancsot használjuk a 16 bites számok összeadásához, melyben a „.w” a word rövidítése jelzi ezt számunkra.

 ;16 bites összeadás előjel nélkül, próba a túlcsordulásra

 mov.w #65535**,**R8

 mov.w #1**,**R9

 add.w R8**,**R9

Hasonlóan a 8 bites számok esetén, itt is szembe futhatunk a problémával, ami túlcsordulás esetén történik. Ennek kipróbálásához a 16 biten ábrázolható legnagyobb számot tekintettük, amely $2^{16}-1=65535$. Ha ehhez a számhoz hozzáadjuk az R9-es regiszterben tárolt 1 értéket, a túlcsordulás miatt 0-t kapunk, ezt jelzi a carry bit 1 értéke is.

1. **32 bites összeadás előjel nélkül**

 ;32 bites összeadás előjel nélkül:

 mov.w #0x3200**,**R4

 mov.w #0x12AC**,**R5

 mov.w #0x24A1**,**R6

 mov.w #0#9872**,**R7

 add.w R4**,**R5

 addc.w R6**,**R7

Mivel a mikrokontroller regiszterei csak 16 bites értékek tárolására képesek, ezért a 32 bites műveletek végre hajtásához minden 32 bites szám tárolásához 2x16 bitet alkalmaztunk. Az egyikben tároljuk a szám nagyobb helyiértékű tagját, a másikba pedig a kisebb helyiértékűt. A művelet végzése során először összeadjuk a kisebb helyiértékű tagokat, majd a carry bitet figyelembe véve a nagyobb értékeket is. Tehát, a példa szerint az első számot eltároltuk az R4-es és R5-ös regiszterben, a második számot pedig az R6-os, illetve R7-es regiszterekben. Először összeadjuk a étszám végét, azaz az R5-öt és R7-et, majd összeadjuk az elejét is.

1. **8 bites összeadás előjellel**

 ;8bites, előjel nélküli összeadás

 mov.b #**+**8**,**R4

 mov.b #**+**17**,**R5

 add.b R5**,**R4

Az előjeles összeadásnál az első példánkban a dolgok hasonlóan történtek, mint az első esetben előjelek nélkül. A helyzet akkor vált érdekessé, mikor a túlcsordulást teszteltük. Mivel a 8 bites számokat tekintjük és tudjuk, hogy az egyik bit előjel bit lesz, ezért csak 7 bittel gazdálkodhatunk. Ez azt jelenti, hogy az előjelnélküli helyzettel ellentétben itt a $2^{7}-1=127$ legnagyobb ábrázolható szám.

 ;8bites, előjel nélküli összeadás, példa túlcsordulásra

 mov.b #**+**127**,**R6

 mov.b #**+**1**,**R7

 add.b R7**,**R6

 A túlcsordulásra vonatkozó példakódunkra a vártak szerint az owerflow bit 1-re változott.

1. **16 bites összeadás előjellel**

 ;16 bites előjel nélküli összeadás

 mov.w #**+**130**,**R8

 mov.w #**+**150**,**R9

 add.w R9**,**R8

A 8 biteshez hasonlóan történnek a dolgok, itt 15 bit tárolja a számértéket, és az első bit értéke jelenti az előjelet, tehát a várható maximum érték $2^{15}-1=32767$. A túlcsordulásos példa kapcsán megfigyeltük, hogy az eredményünk negatívvá válik, és az overflow és a negatív bit értéke nullává változik.

 ;16 bites összeadás előjellel példa, túlcsordulásos

 mov.w #**+**32767**,**R4

 mov.w #**+**2**,** R5

 add.w R5**,**R4

1. **32 bites összeadás előjellel**

A számot két 16 bites regiszterben tároljuk, az alacsony helyiértékű részét előjel nélkül, a magasabb helyiértékű tagot viszont előjeles ábrázolásban. Ezek összeadásakor ADDC-t használunk.

 ;32 bites összeadás előjellel példa túlcsordulásra

 mov.w #0xEFFF**,**R4

 mov.w #0x0002**,**R5

 mov.w #0x0000**,**R6

 mov.w #0xFFFE**,**R7

 add.w R5**,**R7

 addc.w R4**,**R6

1. **8 bites kivonás előjel nélkül**

 ;8bites kivonás előjel nélkül

 mov.b #2**,** R4

 mov.b #4**,** R5

 sub.b R5**,**R4

A kivonás előjel nélküli számok esetén rendkívűl hasonlóan működik az összeadáshoz. Amennyiben kilépünk az intervallumról, tehát az eredményünk negatív lesz a számkezelés újraindul a maximumtól. A fenti példában az R5-ös regiszter értékéből vonjuk ki az R4-est, mivel az eredmény negatív, az előzőekben megfogalmazott jelenséget meg is tapasztaltuk.

1. **16 bites kivonás előjel nélkül**

 ;16 bites kivonás előjel nélkül

 mov.b #15**,**R4

 mov.b #10**,**R5

 sub.w R5**,**R4

Az előzőekkel hasonlatosan működött, amennyiben a 15-ből vontuk ki a 10-et a kivonás minden gond nélkül elvégezhető volt, amikor azonban a kettőt megcseréltük a 8 bites esethez hasonlóan a számozás újraindult, tehát $2^{16}-1-4=65531$.

 ;16 bites kivonás előjel nélkül, előröl indulás

 mov.b #15**,**R4

 mov.b #10**,**R5

 sub.w R5**,**R4

1. **32 bites kivonás előjel nélkül**

Az eddigiekhez hasonlóan a számokat két regiszterben tároltuk, a magasabb helyiértékek levonásánal a subc parancs segítségével a borrow bitet is figyelembe vettük. Az eredmény a vártak szerint ki is jött: $2^{32}-1-1=4,294,967,294$.

 ;32 bites előjeles kivonás

 mov.w #0x001**,**R4

 mov.w #0x002**,**R5

 mov.w #0x0002**,**R6

 mov.w #0x0000**,**R7

 sub.w R4**,**R6

 subc.w R5**,**R7

1. **8 bites kivonás előjellel**

 ;8 bites kivonás előjellel

 mov.b #**-**7**,** R04

 mov.b #**+**10**,** R05

 sub.b R04**,**R05

Az overflow bit értéke 0, tehát az eredmény előjele jó.

1. **16 bites kivonás előjellel**

 ;16 bites kivonás előjellel

 mov.w #−32**,** R4

 mov.w #17**,** R5

 sub.w R4 **,** R5

A rendszer hasonlóan működik, mint az előző esetben.

1. **32 bites kivonás előjellel**

 ;32 bites kivonás előjellel

 mov.w #**-**322**,**R4

 mov.w #**-**473**,**R5

 mov.w #**-**811**,**R6

 mov.w #**-**12**,**R7

 sub.w R6**,**R4

 subc.w R7**,**R5