9. mérés - Mikrokontroller I

Mérést végezte: Dobránszky Márk
Mérőtárs: Bartha András

Mérés dátuma: 2015. április 22.
Mérés helye: PPKE Információs Technológiai és Bionikai Kar

A mérés célja:[1]

Mikroszámítógépek gyakorlati megismerése, egyszerűbb műveletek, időzítések megvalósítása, valamint az analóg-digitális átalakító megismerése.

A mérés során használt eszközök:

- mikrokontroller-panel
- céláramkör
- JTAG adapter

Az eredmény meghatározásának körülményei:[2]

- A környezet üzembehelyezése
- A program szerkesztése

  A project-ban a demo.asm file tartalmazza az előkészített assembly forrást.

  A szerkesztő ablakban a számítógépen szokásos módon lehet változtatni a forrás file-t.

- A program fordítása és szerkesztése

  Workspace ablakban a project nevére jobb gombbal kattintva vagy a ikonsorban található a RebildAll vagy Make parancs.

  Az esetleges hibaüzeneteket a Messages ablakban lehet megtalálni. A hibák javítása után újra kell fordítani a project-et.

- A program futtatása

  Sikeres fordítás és linkelés után az ikonsorban található Debug segítségével tudjuk elindítani a nyomkövetési módot.

  Ebben az üzemmódban megjelenő ujabb ikonok biztosítják a végrehajtási módokat. (Reset; Break; Step Over; Step Into; Step Out; Next statement; Rum to cursor; Go; Stop debuging)

  A regiszterek tartalmas megtekinthető hexadecimális formában a Register ablakban.

[1] `http://users.itk.ppke.hu/~tihanyia/bevezet/MikrokontrollerII.pdf` (Hozzáférés: 2015. május 16.)
[2] `http://users.itk.ppke.hu/~tihanyia/bevezet/MikrokontrollerI.pdf` (Hozzáférés: 2015. május 16.)

A mérés ismertetése

1. feladat:

Végezzen el összeadást két 8 bites előjel nélküli szám között. Helyezze az összeadandókat mint konstansokat egy-egy regiszterbe, majd végezze el az adatok összeadását. Az eredményt ellenőrizze a Registers ablakban.

```
mov.b #15, R04
mov.b #10, R05
add.b R04,R05
```

Magyarázat: A 04-es regiszterbe beletesszük a 15-öt, a 05-ösbe a 10-et, majd összeadjuk őket. Az eredmény az 05-ös regiszterben lesz. A 8 bitességet a „.b" (byte) toldalék határozza meg az utasítás után.

```
mov.b #255, R04
mov.b #1, R05
add.b R04,R05
```

Magyarázat: A 04-es regiszterbe beletesszük a 255-öt, a 8 biten ábrázolható legnagyobb számot, az 05-ös regiszterbe pedig 1-et. Ezek összeadása után az eredmény 0-lesz a túlcsordulás miatt. Ezt jelzi a Carry bit 1 értéke is.

2. feladat:

Végezzen el összeadást két 16 bites előjel nélküli szám között. A művelet elvégzése során vizsgálja a carry bit értékét. A jegyzőkönyve csatolja az elkészített programokat, valamint az ellenőrzés eredményének értékelését is.

```
mov.w #255, R04
mov.w #10, R05
add.w R04,R05
```

Magyarázat: A 04-es regiszterbe beletesszük a 255-öt, az 05-ösbe a 10-et, majd összeadjuk őket. Az eredmény: 265. A 16 bitességet a „.w" (word) toldalék határozza meg az utasítás után.

```
mov.w #0xffff, R04
mov.w #0x0001, R05
add.w R04,R05
```

Magyarázat: A 04-es regiszterbe beletesszük a 65535-öt ($2^{16}-1$), a 16 biten ábrázolható legnagyobb számot, az 05-ös regiszterbe pedig 1-et. Ezek összeadása után az eredmény 0-lesz a túlcsordulás miatt. Ezt jelzi a Carry bit 1 értéke is.

3. feladat:

Végezzen el összeadást két 32 bites előjel nélküli szám között. A művelet elvégzése során vizsgálja a carry bit értékét. A jegyzőkönyvbe csatolja az elkészített programokat, valamint az ellenőrzés eredményének értékelését is.

```
mov.w #0xffff,R4
mov.w #0xffff,R5
mov.w #0x0002,R6
mov.w #0x0000,R7

add.w R4,R6
addc.w R5,R7
```

Magyarázat: a számok két regiszterben vannak tárolva, mert csak 16 bites adatbuszunk van. Az első szám kisebb helyértékű fele az R4-es, a nagyobb helyértékű fele az R5-ös regiszterben van. Az összeadás után az eredmény kisebb helyértékű fele a R6 regiszterben és nagyobb helyértékű fele az R7 regiszterekben vannak. Az összeadás után az eredmény 1, mivel az első összeadás carry bitjét figyelembe vettük a második összeadásnál.

4. feladat

A tanultakat ellenőrizze az 1;2;3; feladat megoldásával előjeles környezetben is. A jegyzőkönyve csatolja az elkészített programokat, valamint az ellenőrzés eredményének értékelését is.

```
1:
   mov.b #+15, R04
   mov.b #+10, R05
   add.b R04,R05
```

Magyarázat: ugyan az a szituáció mint az első feladatnál.

```
   mov.b #+127, R04
   mov.b #+1, R05
   add.b R04,R05
```

Magyarázat: A 04-es regiszterbe beletesszük a 127-et $(2^8/2-1)$, az előjeles 8 biten ábrázolható legnagyobb számot, az 05-ös regiszterbe pedig 1-et. Ezek összeadása után az eredmény -128-lesz $(-(2^8/2))$ a túlcsordulás miatt. Ezt jelzi a Carry bit 1 értéke is.

```
2:
   mov.w #+128, R04
   mov.w #+10, R05
   add.w R04,R05
```

Magyarázat: A 04-es regiszterbe beletesszük a 128-at, az 05-ösbe a 10-et, majd összeadjuk őket. Az eredmény: 138. A 16 bitességet a „.w” (word) toldalék határozza meg az utasítás után.

```
   mov.w #+0x7fff, R04
   mov.w #+0x0001, R05
   add.w R04,R05
```

Magyarázat: A 04-es regiszterbe beletesszük a 32767-et $(2^{16}/2-1)$, az előjeles 16 biten ábrázolható legnagyobb számot, az 05-ös regiszterbe pedig 1-et. Ezek összeadása után az eredmény -32768 $(-(2^{16}/2))$lesz a túlcsordulás miatt. Ezt jelzi a Carry bit 1 értéke is.

```
3:
   mov.w #+0xffff,R4
   mov.w #+0x7fff,R5
   mov.w #+0x0002,R6
   mov.w #+0x0000,R7

   add.w R4,R6
   addc.w R5,R7
```

Magyarázat: a számok két regiszterben vannak tárolva, mert csak 16 bites adatbuszunk van. Az első szám kisebb helyértékű fele az R4-es, a nagyobb helyértékű fele az R5-ös regiszterben van. Az összeadás után az eredmény kisebb helyértékű fele a R6 regiszterben és nagyobb helyértékű fele az R7 regiszterekben vannak. Az összeadás után az eredmény 1, mivel az első összeadás carry bitjét figyelembe vettük a második összeadásnál.

5. feladat

Végezzen el kivonást két 8 bites előjel nélküli szám között. Helyezze a kissebítendőt az egyik míg a kivonandót egy másik regiszterbe, majd végezze el az adatok kivonását. Az eredményt ellenőrizze a Registers ablakban. A program működését lépésenkénti futtatással lehet ellenőrizni. Ismételje meg a feladatot más konstansokkal is. Ellenőrizze, hogy mi történik akkor ha az eredmény túllép a számábrázolási határon. A jegyzőkönyve csatolja az elkészített programokat, valamint az ellenőrzés eredményének értékelését is.

```
mov.b #15, R04
mov.b #10, R05
sub.b R04,R05
```

Magyarázat: a 04-es regiszterbe beletesszük a 15-öt, az 05-ös regiszterbe pedig a 10-et. Kivonjuk a 04-es regiszterből az 05-öst, az eredmény pedig a 05-ösben fog megjelenni, és értéke 5.

```
mov.b #15, R04
mov.b #10, R05
sub.b R05,R04
```

Magyarázat: ezesetben az 05-ös regiszterből vonjuk ki a 04-est így alulcsordulást okozunk, és az eredmény a 04-es regiszterben 251 ($2^8 - 1 - 4$).

6. feladat

Végezzen el kivonást két 16 bites előjel nélküli szám között. A művelet elvégzése során vizsgálja a borrow bit értékét. A jegyzőkönyve csatolja az elkészített programokat, valamint az ellenőrzés eredményének értékelését is.

```
mov.w #15, R04
mov.w #10, R05
sub.w R04,R05
```

Magyarázat: a 04-es regiszterbe beletesszük a 15-öt, az 05-ös regiszterbe pedig a 10-et. Kivonjuk a 04-es regiszterből az 05-öst, az eredmény pedig a 05-ösben fog megjelenni, és értéke 5.

```
mov.w #15, R04
mov.w #10, R05
sub.w R05,R04
```

Magyarázat: ezesetben az 05-ös regiszterből vonjuk ki a 04-est így alulcsordulást okozunk, és az eredmény a 04-es regiszterben 65531 ($2^{16} - 1 - 4$).

7. feladat

Végezzen el kivonást két 32 bites előjel nélküli szám között. A művelet elvégzése során vizsgálja a borrow bit értékét. A jegyzőkönyve csatolja az elkészített programokat, valamint az ellenőrzés eredményének értékelését is.

```
mov.w #0x001,R4
mov.w #0x002,R5
mov.w #0x0002,R6
mov.w #0x0000,R7

sub.w R4,R6
subc.w R5,R7
```

Magyarázat: Hasonlóképpen történik minden mint a korábbi esetekben, itt az eredmény $2^{32} - 1 - 1$

8. feladat

A tanultakat ellenőrizze az 5;6;7; feladat megoldásával előjeles környezetben is. A jegyzőkönyve csatolja az elkészített programokat, valamint az ellenőrzés eredményének értékelését is.

```
5:
   mov.b #+15, R04
   mov.b #+10, R05
   sub.b R04,R05
```

Magyarázat: a 04-es regiszterből kivonjuk a 05-öst, az eredmény a 05-ös regiszterben 5.

```
   mov.b #-128, R04
   mov.b #+10, R05
   sub.b R04,R05
```

Magyarázat: Ez esetben a -128-ból vonjuk ki a 10-et. Ekkor az alulcsordulás miatt az eredmény $2^7 - 1 - 9$ azaz 118.

```
6:
   mov.w #-128, R04
   mov.w #+10, R05
   sub.w R04,R05
```

Magyarázat: A 04-es regiszterből kivonjuk az 05-ös regiszter értékét, az eredmény pedig az 05-ös regiszterben lesz és az értéke -138. Az alulcsordulást a Borrow bit 1 értéke jelzi.

```
7:
   mov.w #-484,R4
   mov.w #-352,R5
   mov.w #-334,R6
   mov.w #-478,R7

   sub.w R6,R4
   subc.w R7,R5
```

Magyarázat: Hasonlóképpen történik minden mint a korábbi esetekben, itt az eredmény R04=0xFF6A R05=0x007D. Az alulcsordulást a Borrow bit 1 értéke jelzi.