**Mikrokontroller alapú rendszerek**

**Sipos Roland**

**KARZPU**

**Házi Feladat**

**4 digites BCD számból Bináris szám készítése**

**Feladat kiírás:**

Regiszterekben található 4 digites BCD kódú szám átalakítása bináris számmá. Belépéskor az első regiszter tartalmazza a nagyobb helyiértékű két digitet, a másik regiszter a kisebb helyiértékű 2 digitet. Az eredmény 16 bites bináris szám 2 regiszterben.

**Bemenet:** átalakítandó BCD szám. (R0 és R1 regiszterek)

**Kimenet:** átalakított bináris szám. (R2 és R3 regiszterek)

**A megoldás egyszerű megfogalmazása:**

1. **Átalakítom a BCD számokat decimális számokká**, azaz a BCD digiteket 4 különböző regiszterbe másolom át, mindegyiknél az alsó 4 bitre, ezzel megkapva az 1-es (R0), 10-es (R1), 100-as (R2) és 1000-es (R3) helyiértékek decimális értékeit.
2. Az értékkészlet 0…9999-ig tart, amely tartomány ábrázolásához 14 bitre van szükség (999910=270F16). Az akkumulátor és a regiszterek mérete azonban csak 8bit, **tehát két részre bontom a decimális értékeket**:
   1. **R0 regiszter => „ALSÓ”:** 1-esek és 10-esek *(utóbbi megszorozva tízzel, hogy a valós értékéhez jussak, 0…99 tartomány)*
   2. **R1 regiszter => „FELSŐ”:** 100-asok és 1000-esek *(utóbbi megszorozva tízzel, hogy a valós értékéhez jussak, 0…99 tartomány, tehát 7 biten ábrázolható)*
   3. Az „ALSÓ” és „FELSŐ” számok közötti **átvitelt** az **R7 regiszterrel** oldom meg, ez tartalmazza aktuálisan a „FELSŐ” maradékát, amit hozzáadok az „ALSÓ” értékhez.
3. **A decimális számokból bináris számokat készítek** *(kézi módszer, ciklikusan végrehajtva):* 
   1. **Eredmény bit létrehozása:** Az „ALSÓ” szám legalacsonyabb helyiértéken lévő bitje adja meg, hogy melyik bit kerül az eredmény regiszterekbe *(R2 és R3).* Amennyiben ez az érték páratlan *(azaz 1),* kivonok az értékből 1-et, így 2-vel osztható lesz.
   2. **A bit tárolása:** *(maximálisan 14 bitet kell tárolni, ebből az alsó 8 bitet R2-ben, a felső 6 bitet R3-ban tárolom úgy, hogy az R3 legfelső 2 bitje az eredmény kiértékelésekor mindenképp 0 legyen)* 
      1. Az **R4 regiszterben** tárolom *(kezdőértéke 0x01h)* az aktuális **eredmény bit helyiértékét**, tehát ciklikusan shiftelem balra az 1-est.
      2. **Az F0 flag tárolja, az alsó vagy a felső eredmény regiszterbe kell írni** az eredményt. F0 értéke alapértelmezetten 0, ekkor az alsó regiszterbe írok (R2), amennyiben az R4 (eredmény bit helyiértékét tartalmazó regiszter) a futás során ismét a kezdőértékét tartalmazza, az F0 flag 1-es értéket kap, így a felső eredmény regiszterbe írok tovább.
      3. A tároláshoz az akkumulátorban lévő **eredmény bitet** megszorzom az R4 értékével, így az **aktuális helyiértékre shfitelem** azt, ezt pedig **hozzáadom** az **F0 által kiválasztott** **eredményregiszterek egyikébe** (alapértelmezetten 0x00h a kezdőértékük, tehát csak az adott helyiérték változhat az összeadás során.
   3. **Az átviteli érték meghatározása:** az ALSÓ és FELSŐ értékek közötti átvitelt az **R7** regiszterben tárolom. Ellenőrzöm, hogy a FELSŐ értéknek milyen a legalsó bitje, amennyiben páratlan *(azaz 1), az átvitel: 100/2 = +50 lesz (ellenkező esetben pl.: 200/2 = +100, amihez nem kell átvitel, tehát az átvitel 0).*
   4. **Értékek aktualizálása:** A FELSŐ és az ALSÓ értéket is elosztom kettővel *(az osztás után az ALSÓ-hoz hozzáadom az R7-ben tárolt átvitelt is)* , így felkészítve őket a következő ciklus műveleteire, ill. emiatt egyszer csak az értékeik 0-k lesznek, ami a ciklus végét, az eredmény létrejöttét jelzi.
4. **Egyszerűsítések a működésben:**
   1. **Csak 1. digit nem nulla:** ilyenkor R0 = R2 esetén előáll az eredmény.
   2. **Felső decimális szám nulla:** a kezelési ciklusát átlépi a függvény és előrről kezdi a ciklust.
   3. **Hiba esetén** *(BCD számok nem a megfelelő tartományban vannak):* kilép és **F0 flagen** és a **kimeneti regisztereken** hibát jelezve. *(HIBA: F0 = 1, R2,3 = 0xFFh, OK: egyébként)*

**Működési blokkdiagrammok:**

*(a nagyobb egységek működése a rájuk utaló szín alapján kapcsolható össze)*

R0L = BCD1

R0H = BCD2

R1L = BCD3

R1H= BCD4

PUSH

PSW, R(0,1,4,7),

ACC, B

Áthelyezés

(+ ellenőrzés)

BCD -> Dec.

R1 = Dec3+Dec4\*10

R0 = Dec1+Dec2\*10

R0 = BCD1

R1= BCD2

R2 = BCD3

R3 = BCD4

Dec. -> Bin.

POP

PSW, R(0,1,4,7),

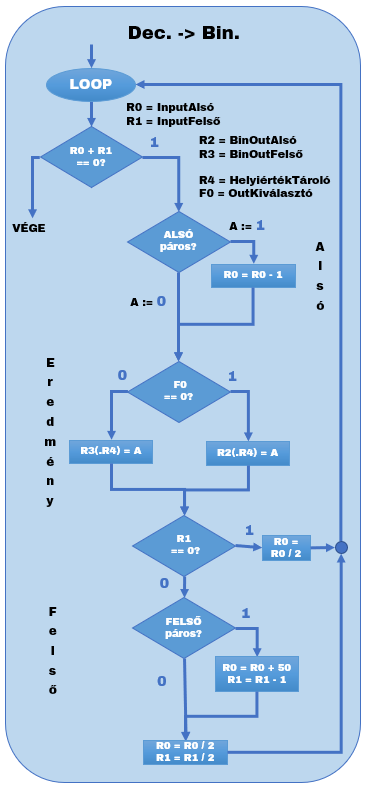
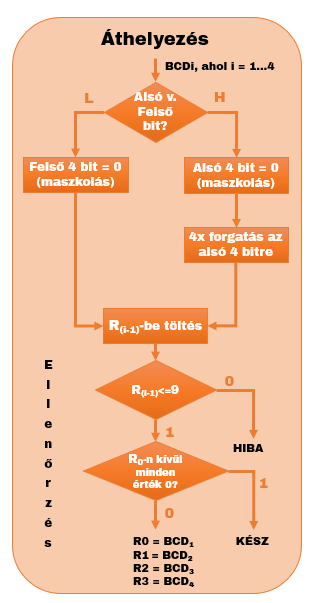
ACC, B

HIBA

KÉSZ

R2 = Bin0...7

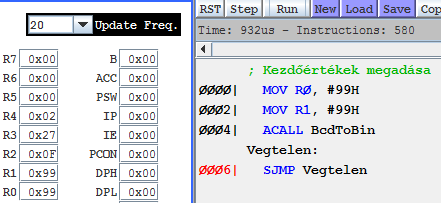
R3 = Bin8...15



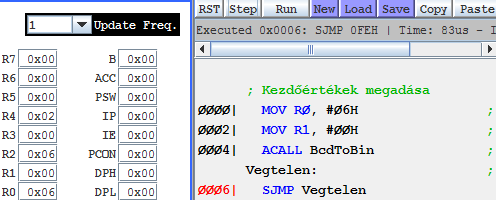
**Ellenőrzés:**

**Input:** 9999bcd **-> Output:** 0x270Fh

*(legnagyobb szám esetén 1ms a futás ideje)*

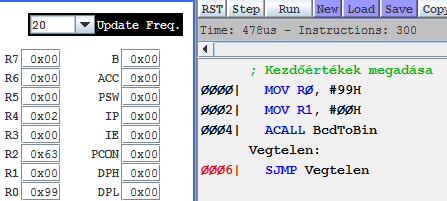


**Ha csak BCD1 nem nulla,** akkor nagy mértékben csökken a számítási idő: kb. 83usec

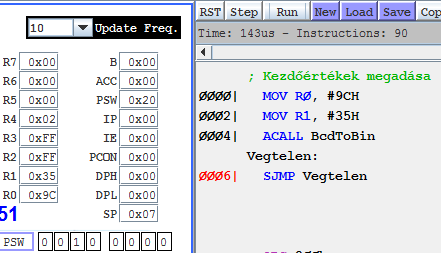


**A Felső helyiértékek elhagyásával** *(a Felsőre vonatkozó parancsok teljesen kimaradnak)*

kb. 05.ms a számítás ideje.



A **Hibás érték** kezelése 150usec alatt következett be a **megfelelő jelzési értékekkel.**



**A dokumentum beadásával nyilatkozom arról, hogy a dolgozat a saját munkám.**

**Források: http://www.keil.com/dd/docs/datashts/atmel/at\_c51ism.pdf**

**Sipos Roland**

**KARZPU**