Mikrokontroller alapú rendszerek – Házi Feladat

Koren Zoltán-HOTQM1

2020.11.30.

Feladat kiírás:

Regiszterekben található 16 bites előjeles egész átalakítása 5 db BCD kódú számmá + előjellé. Az eredményt 3 regiszterben kapjuk vissza: az elsőben a legfelső bit az előjel, az alatta lévő 3 bit 0, ez alatt a legmagasabb helyiértékű digit. A második regiszterben a következő két digit, a harmadikban a legkisebb helyiértékű két digit. Bemenet: az átalakítandó szám 2 regiszterben, kimenet az átalakított szám 3 regiszterben.

Gondolatmenet:

A program elméleti felépítése:

- Előjel kiderítése
  - Az input felső bitjét az akkumulátor segítségével ki kell maszkolni, és elmenteni a legfelső bitet.
  - Amennyiben negatív szám volt, azt átváltani pozitív számba
    - Bitek negálása és +1 hozzáadása
  - Figyelni kell arra, hogy a túlcsordulás esetén a felső biteket tartalmazó regiszterben is folytassuk a hozzáadás.
- 16 bit esetén, legfeljebb 5 számjegyű szám lehetséges.
- A bemeneti számot 4x kell leosztani egymás után
  - A maradékok lesznek a „digitek”, illetve az utolsó osztás esetén maga a hányados
- Ezután a PackedBCD-k létrehozása a cél

A probléma:

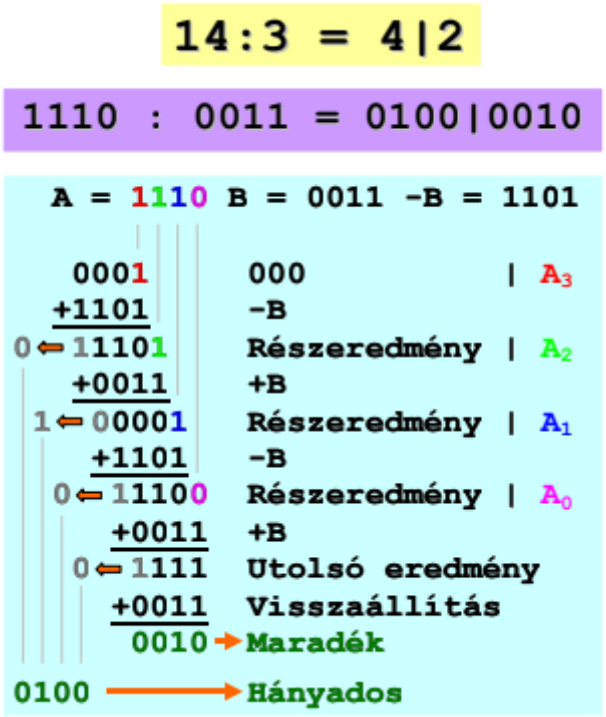
A 8051-es utasításkészlet nem támogatja a 8bitnél nagyobb számok osztását:

INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1

Így egy másik megoldást kell keresni, hogy az osztásokat elvégezzük.

Megoldás

A 10-zel való osztást elemi műveletekre kell lebontani (összeadás, kivonás). Ennek az algoritmusa a nem visszaállításos osztás:



(algoritmust bemutató ábra 4 bitre)

A házi feladatomban ezt az algoritmust valósítottam meg.

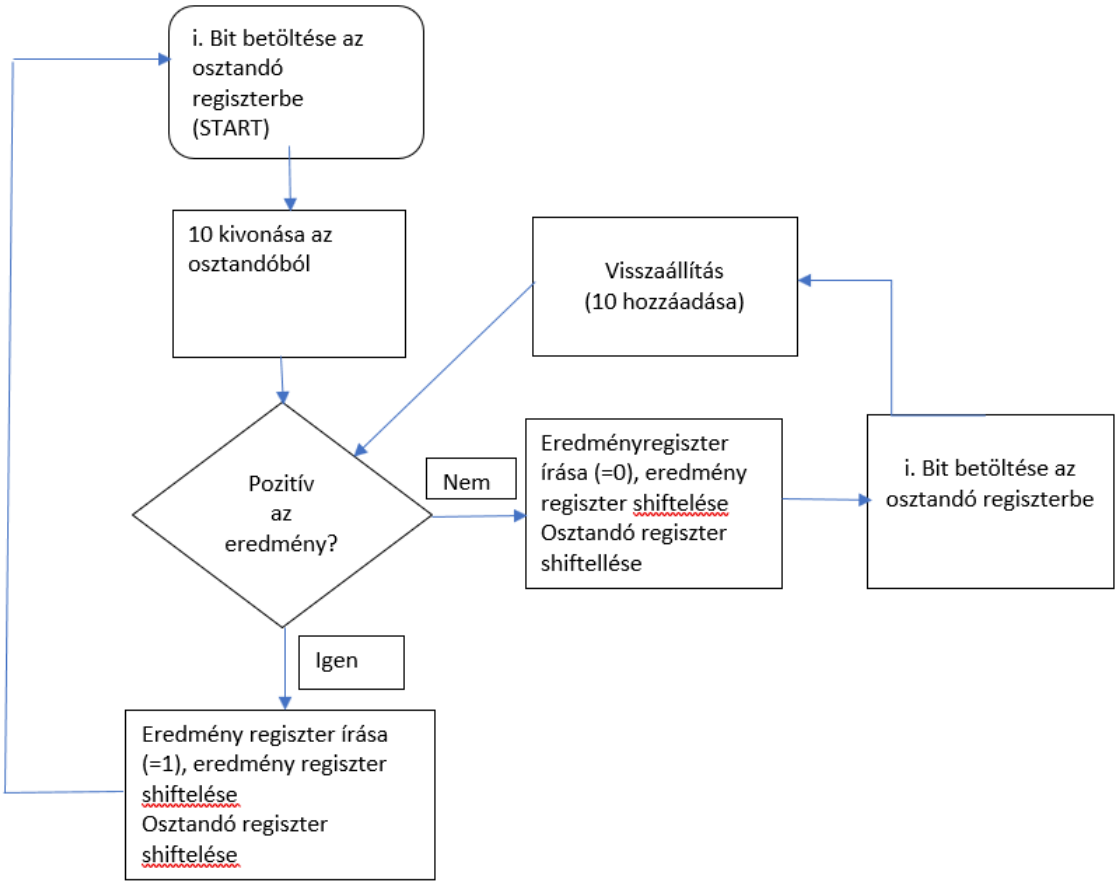
Működése:

Az algoritmus során kivonásokkal, összeadásokkal tudunk egy nagyobb számot elosztani valahány bites számmal. Az osztandó számunkat egy regiszter tartalmazza, aminek LSB-jére minden ciklusban betöltjük az input szám egyik bitjét (fentről lefele indulva sorban). Ebből következik, hogy 16 ciklusra van szükségünk ahhoz, hogy eredményt kapjunk az algoritmus végén.
Az első lépésben az osztandóból (a betöltés után) kivonunk 10-et (hozzáadunk -10-et). Az így kapott szám 5. bitjét megvizsgáljuk.
Ha:

- Az 5. bit 1, akkor nem volt meg benne a szám, a hányadost tartalmazó regiszterbe egy 0 bitet írunk a megfelelő helyre, és a következő ciklusban hozzáadni fogunk 10-et.
- Az 5. bit 0, akkor meg volt benne a szám, a hányadost tartalmazó regiszterbe egy 1 bitet írunk a megfelelő helyre, és a következő ciklusban kivonni fogunk 10-et.

Ezután az eredményt balra shifteljük, betöltjük a következő számot az inputból, és az így keletkezett szám lesz az új osztandónk. Ezután megkezdődik a következő ciklus (az előző eredményétől függően összeadással/kivonással).

Osztás vége:
Amennyiben véget ért az összes ciklus, megvizsgáljuk, hogy a hányadosba utoljára betett bit mi volt. Ha 0, akkor még az „osztandót” vissza kell állítanunk, így hozzáadunk 10-et. Az így kapott szám lesz a maradék.



Az algoritmus megáll, ha i eléri az osztandó bit bitszámát. Ha az utolsó kivonás után még mindig negatív az eredmény, az osztandó regisztert nem kell shiftelni, de vissza kell állítani, hogy a maradékot megkaphassuk.

Program

Az algoritmusról többet nem szándékozom írni (az egész el van rejtve a DIV_10 nevezetű szubrutinban).
A kezdeti inicializálások után, az input regiszter tartalmát az adatmemóriába elmentjük. Ez azért kedvező számunkra, mert így további 2 regiszter áll rendelkezésünkre a program folyamán, illetve az input regiszterhez alapvetően nem nyúlunk sokszor a futás során.
A program kezdetén eldöntjük, hogy milyen előjele volt az input számnak. Ezt a „tudást” eltároljuk, majd haladunk az osztások felé.
4 db osztást kell a program folyamán elvégezni. A 4 db osztás 4 db maradékot eredményez, illetve az 5. egy hányadost. Így állnak majd elő a szám digitjei.
Az osztások után az input regiszter tartalmát mindig felülírjuk (hiszen nincs rá szükségünk), és az új osztás során az előző osztás hányadosait használjuk majd inputként.
Miután az osztások lementek, a kimeneti regiszterek előállítása a cél. Ezt könnyű megtenni, hiszen minden egyes számjegy 4 biten elfér, és a számjegyeken kívül csak az akkumulátorra van szükségünk a regiszterek előállítására. Az előállítás során kihasználjuk azt, hogy az akkumulátor megtudja cserélni a felső/ alsó 4 bitjét.

Felhasznált anyagok:

- Tárgyhonlapról letölthető:
  - 8051 utasításkészlet
  - Mikrokontroller alapú rendszerek elektronikus jegyzet 1. fejezete
- Konzultáció Kovács Adorjánnal

Kijelentem, hogy a házi feladatot önállóan készítettem el, a házi feladat a saját munkám:

Koren Zoltán, 2020.11.30.