Mikrokontoller alapú rendszerek - házi feladat dokumentáció

Készítette: Jójárt Bence - GFHSCH

A feladat:

Belső memóriában lévő 16 bites előjeles szám "gyorsszorzása" 10 hatványa szorzóval (1, 10, 100, 1000, 10000), a 10 kitevője bemenő paramétere a rutinnak. A gyorsszorzás azt jelenti, hogy kihasználjuk a szorzó speciális voltát (pl. 10=8+2, 100=64+32+4), univerzális szorzó használata nem felel meg a feladatnak! Az eredmény is 16 biten legyen ábrázolva, a túlcsordulás ennek figyelembevételével állítandó. Bemenet: szorzandó címe (mutató), szorzó kitevője, eredmény címe (mutató). Kimenet: 16 bites eredmény, OV

Az alkalmazott algoritmus alapvető működése:

Először is, írjuk fel a 10 hatványait, hogy miképpen állnak elő 2 hatványok összegeként:

1=1
10=8+2
100=64+32+4
1000=512+256+128+64+32+8
10000=8192+1024+512+256+16

Egy 2^x hatvánnyal való szorzás bináris környezetben ekvivalens azzal, mintha a szám bitjeit x-szer egymás után elforgatnánk balra. Tehát, ha pl. 10-zel kell szoroznunk egy számot, akkor csinálhatjuk azt, hogy először elforgatjuk balra a bitjeit háromszor, majd az eredeti szám bitjeit elforgatjuk egyszer, és a két eredményt összeadjuk. 10 minden hatványa előállítható az előbbiek szerint ilyen alakban, 2 hatványok összegeként. Ezen a tulajdonságon alapul a program működése.

A feladatot nehezíti, hogy a szorzandó szám 16 bites, nekünk pedig csak 8 bites regisztereink vannak. Ezt a problémát úgy oldjuk meg, hogy két-két regisztert (azaz regiszterpárokat) használunk a szorzandó eltárolására, az egyes forgatások véghezvitelére, illetve az eredmény eltárolására. A carry flaget is aktívan használjuk (azon keresztül forgatunk, RLC paranccsal), amely az alsó bájtról keletkezett átvitelt továbbítja a felső bájtra.

Továbbá a feladat szerint figyelni kell a túlcsordulásra, és ezt az OV flagben jelezni. Ehhez egy bitet használunk a bit direkt megcímzésével, amit alapértelmezetten 0-ra állítunk, és ha valamikor túlcsordulást tapasztalunk, véglegesen 1-be állítjuk. A túlcsordulás vizsgálata a következőképpen történik: ha az eredmény felső bájtja kezdetben 0, és valamilyen művelet (forgatás vagy összeadás) során 1-be állt, ill. fordítva, akkor pozitív->negatív ill. fordítva átmenet történt az eredményben, ami, mivel pozitív szám a szorzó, nem lehet. Tehát ez csak akkor következhet be, ha túlcsordult az eredmény. Ezt a legegyszerűbben a két érték XOR-olásával, majd a legfelső bit maszkolásával tudjuk megvizsgálni.

Az algoritmus bemutatása táblázatosan, egy példán keresztül:

Legyen a szorzandó szám a 273 = 0x0111, a szorzó kitevője pedig 2. Azaz a várt eredmény: 273*10^2=27300

Az egyes regiszterek tartalma:

- R0: szorzandó alsó bájtja
- R1: szorzandó felső bájtja
- R2: hányadik rotate ciklusnál tartunk
- R3: hány rotate ciklus kell még
- R4: itt forgatunk (alsó bájt)
- R5: itt forgatunk (felső bájt)
- R6: végleges eredmény (alsó bájt)
- R7: végleges eredmény (felső bájt)
- Bank1,R0 (ezt a továbbiakban R8-nak nevezem): az aktuális rotate ciklusban hányat kell még forgatni

Ezek közül R2, R3 és R8 igényel részletesebb magyarázatot. A mi példánkban:

100=64+32+4, tehát összesen kell 3 rotate ciklus. Az első ciklusban forgatnunk kell 6-ot, a másodikban 5-öt, a harmadikban 2-őt, majd az eredményeket összeadni.

A regiszterek tartalma időrendben a példánkban:

R0=00010001
R1=00000001 , ezen regiszterek értéke végig ennyi.

Az első rotate ciklus során: R2=1; R3=2

R5	R4	R8
00000001	00010001	6
00000010	00100010	5
00000100	01000100	4
00001000	10001000	3
00010001	00010000	2
00100010	00100000	1
01000100	01000000	0

Ezután a regiszterek értéke: R5=0x44; R4=0x40; és 0x4440, ami decimálisan 17472=273*64, tehát jó az eredmény.

A második rotate ciklus során: R2=2; R3=1

R5	R4	R8
00000001	00010001	5
00000010	00100010	4
00000100	01000100	3
00001000	10001000	2
00010001	00010000	1
00100010	00100000	0

Ezután a regiszterek értéke: R5=0x22; R4=0x20; és 0x2220, ami decimálisan 8736=273*32, tehát jó az eredmény.

A harmadik rotate ciklus során: R2=3; R3=0

R5	R4	R8
00000001	00010001	2
00000010	00100010	1
00000100	01000100	0

Ezután a regiszterek értéke: R5=0x04; R4=0x44; és 0x0444, ami decimálisan 1092=273*4, tehát jó az eredmény.

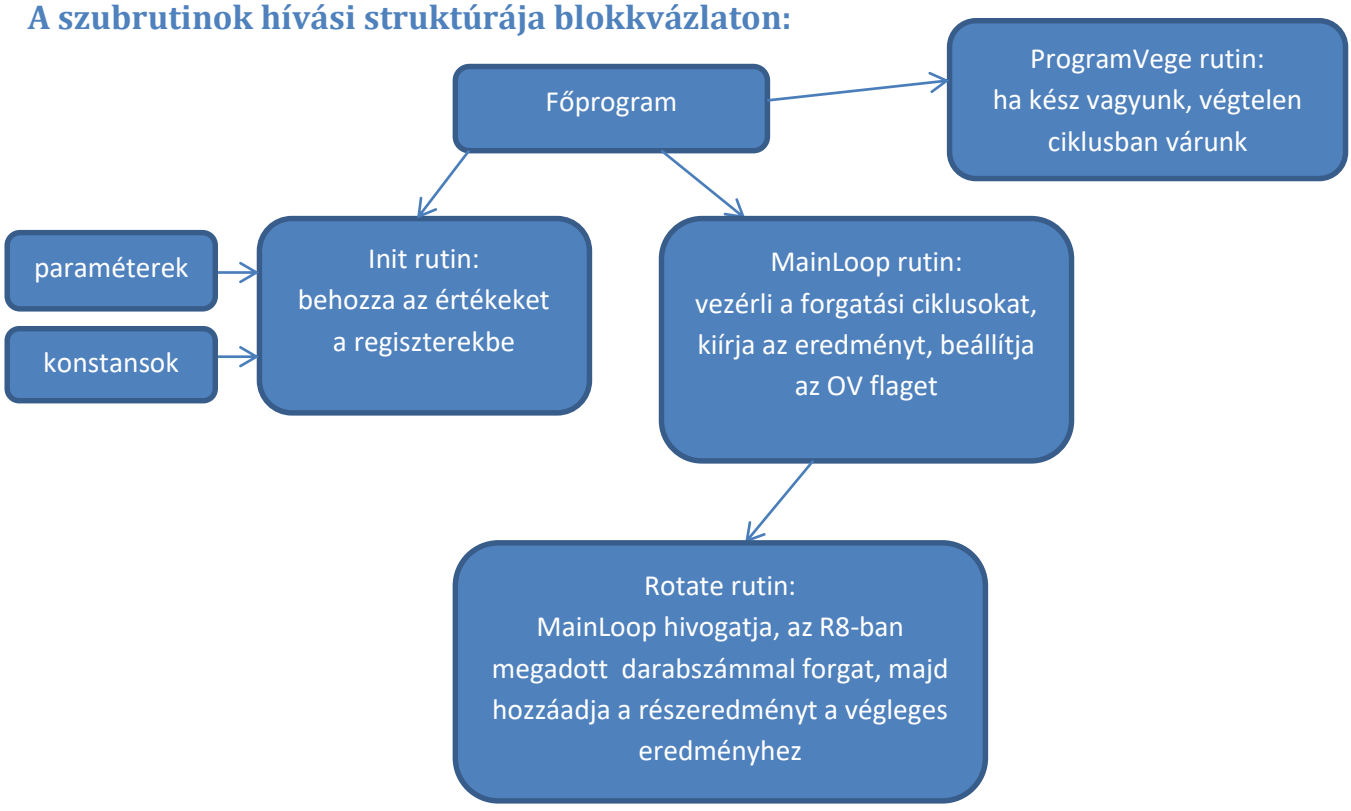
Legvégül: már csak annyi a feladatunk, hogy az egyes rotate ciklusok után az eredményeket összeadjuk. Ezt a következőképpen tesszük:

Kezdetben R6=R7=0. Minden rotate ciklus után:

R6+=R4
R7+=R5+átvitel

Így legvégül R6=0x40+0x20+0x44= 0xA4 lesz, átvitel nincs, R7=0x44+0x22+0x04=0x6A lesz. Ez összevonva 0x6AA4, ami decimálisan: 27300, tehát jó eredményt kaptunk.

A szubrutinok hívási struktúrája blokkvázlaton:



Ez a blokkvázlat az 1-gyel való szorzásra nem teljesül. A hátultesztelő ciklusok miatt (DJNZ) ebben a struktúrában a 0-val való elforgatás nem valósítható meg. Ezért 1-gyel való szorzásnál csak annyit csinálunk, hogy az Init rutin során kiírjuk az eredmény címére a szorzandó értékét, OV=0-ba állítjuk, majd rögtön a ProgramVege rutinra ugrunk.

Felhasznált források:

A tantárgy hivatalos jegyzeteiből, illetve a http://www.keil.com/ oldalon megtalálható Assembly utasítás-leírásokból dolgoztam.

Nyilatkozat:

Nyilatkozom, hogy a dolgozat teljes egészében a saját munkám.

Jójárt Bence, 2017.11.23.