

# HÁZI FELADAT DOKUMENTÁCIÓ

## FELADATKIÍRÁS:

Regiszterekben kapott 16 bites előjel nélküli szám négyzetre emelése, túlcscordulás figyelése. Az eredmény is 16 biten legyen ábrázolva, a túlcscordulás ennek figyelembevételével állítandó. Bemenet: 2 regiszterben a szám. Kimenet: 2 regiszterben az eredmény, CY

## MEGVALÓSÍTÁS:

A feladathoz a felhasznált regiszterek a felhasználó által a program elején található konstansokkal személyre szabhatók, így a szubrutin más programban is felhasználható (a konstansok elnevezése, illetve a hívás során módosított regiszterek részletesen megtalálhatók a .asm fájlban található dokumentációban, röviden összefoglalva: NUMLOW, NUMHIGH a bementet tároló és a kimenetet fogadó regiszterpár, míg RESLOW és RESHIGH a négyzetre emelés során az eredmény alsó, ill. felső byteját tartalmazza).

$$\begin{array}{r}
 0xFFFF \times 0xFFFF = 0xFFFFE\ 0001 \\
 \\
 0xFF \times 0xFF = 0xFE01 \\
 0xFF00 \times 0xFF = 0xFE\ 0100 \\
 0xFF \times 0xFF00 = 0xFE\ 0100 \\
 + \quad 0xFF00 \times 0xFF00 = 0xFE01\ 0000 \\
 \hline
 0xFFFFE\ 0001
 \end{array}$$

A megvalósítani kívánt feladat a specifikációból kiindulva tovább egyszerűsíthető, ehhez tekintsük a bemeneten előforduló legnagyobb számot (0xFFFF).

Az ábra alapján megállapítható, hogy a négyzetre emelés elvégezhető négy szorzás összegeként (alacsony-alacsony, alacsony-magas, magas-alacsony, magas-magas byteok szorzata), melyek közül kettő a szorzás kommutativitásából kifolyólag összevonható, az elkészített szubrutin is ennek megfelelően hajtja végre a négyzetre emelést.

Továbbá a példa alapján levonhatóak a következtetések arra vonatkozólag is, hogy milyen esetben történhet a számítás során túlcscordulás. Amennyiben az alsó byteot szorozzuk önmagával, akkor a regiszterbe tölthető felső korlát esetében sem tapasztalhatunk túlcscordulást, míg a többi esetben ez a bemeneti paraméterek függvényében változhat.

Ugyanis ha a magas és az alacsony byteokat szorozzuk, akkor csak az alsó 16 bit jelenik meg értékes jegyként a kimeneten, ekkor a túlcscordulást a következőképpen kaphatjuk meg: mivel a szorzatra kétszer is szükségünk van, így az RLC utasítás segítségével a CY flagen keresztül 1

bitet balra shiftelve elvégezhetjük a 2-vel történő szorzást, illetve ezzel a CY flaget ellenőrizve a túlsordulás ellenőrzését is egyszerűen megtehetjük - ehhez azonban még továbbá hozzá tartozik a MUL AB eredményeként A, B regiszterekben előálló szorzat felső bytejának ellenőrzése (B regiszter, ami a fenti példa alapján is látható, hogy csak a túlsorduláshoz járulhat hozzá, mivel nem fér el 16 biten), így B regiszter 0-val való komparálásával a példában is szemléltetett 8 bites shiftelést megspórolva a 16 biten tárolt eredmény kiszámításával egyidejűleg az overflow megtörténtét is ellenőrizhetjük.

A magas byte önmagával való szorzása esetében egyszerű dolgunk van, ugyanis a 16 bitnyi balra történő shiftelés miatt ez a részeredmény a 16 biten tárolt négyzet értékéhez nem adódik hozzá, így a szorzás eredményeként előálló A, B regisztereket összevagyolva megkapjuk, hogy ez a részeredmény overflow-t jelent-e.

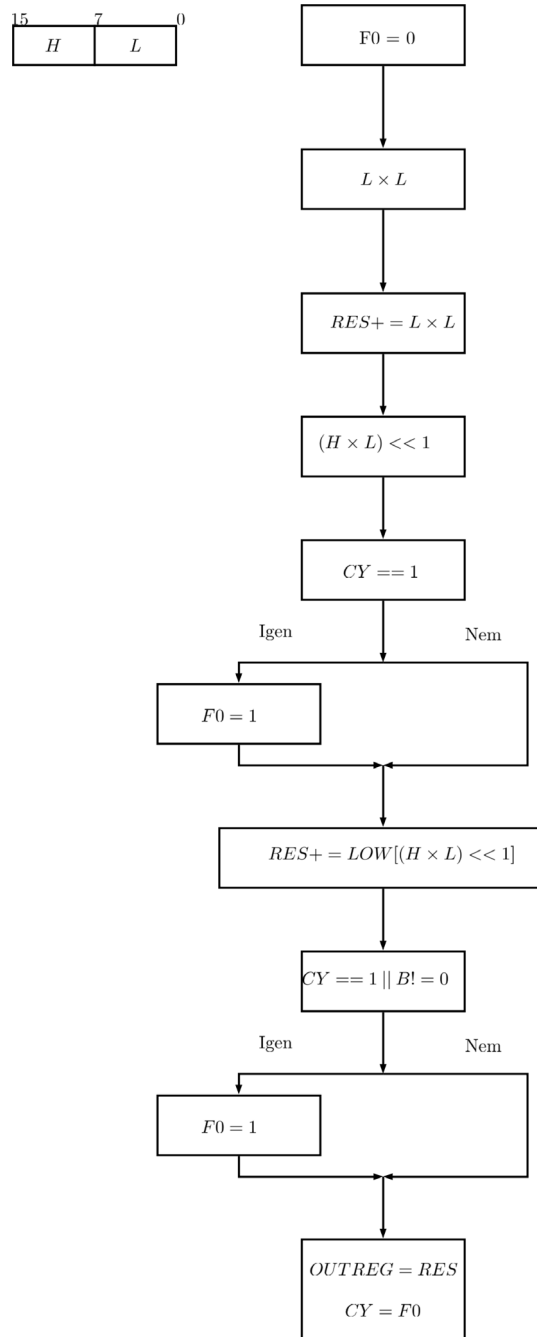
Ezzel a túlsordulás jelzését megvalósító algoritmus bemutatásra került, az implementációt tekintve azonban további megfontolásokat kell tenni, ugyanis a részeredmények kumulálása során az ADD műveletek a CY flaget módosítják, valamint a MUL utasítás is 0-ba állítja annak értékét, így ez nem lehet alkalmas arra, hogy a szubrutin alatt végig tárolja a szükséges flaget, ezért a mikrokontrollerben a felhasználó rendelkezésére bocsátott F0 flag szolgál a CY flag átmeneti tárolójaként, aminek módosításához, illetve mozgatásához rendelkezésre állnak a megfelelő utasítások, így a CY flaghez F0-t hozzávagyolva, majd az eredményt F0-ban eltárolva biztosítva van, hogy a megfelelő értékű flaget kapjuk meg.

A CY flag előállításának tárgyalása után a 16 biten tárolt négyzet kiszámításának a példa alapján nyilvánvaló lépései kerülnek részletesen is bemutatásra. Egyértelmű, hogy az alsó byteok szorzatát minden további nélkül eltároljuk a kódban az eredmény kiszámítására fenntartott regiszterpárban (mivel az ADD helyett a MOV utasítást alkalmazzuk, így nincs szükség a regiszterek 0-ra történő inicializálására). Az overflow vizsgálata során tárgyaltaknak megfelelően a felső byte önmagával alkotott szorzata nem járul hozzá a 16 biten tárolt négyzet számértékéhez, így ezzel jelen megvalósítás értelemszerűen nem foglalkozik.

A felső és az alsó byteok szorzata esetében az alkalmazott megoldás esetében a lehetőségeknek megfelelően a műveletek optimalizálására került sor, ennek következménye, hogy a kommutativitásból kiindulva a MUL AB utasítást követően az RLC utasítás segítségével az eredmény alsó byteját 2-vel megszorozzuk - ezt pedig minden további nélkül megtehetjük, ugyanis az eredmény szempontjából a kapott 16 bites szám valójában 24 bites, amit 8 darab 0 értékű bit alsó helyiértéken történő hozzáfűzésével kaphatunk meg (hangsúlyozandó, hogy ez a balra shiftelés fizikailag nem valósul meg). Így pedig elegendő a 2-vel megszorozott A regiszter tartalmát az eredményt tároló regiszterpár felső helyiértékéhez hozzáadni - ez a lépés pedig 8 órajel megtakarított, mivel nem kell 8 bitnyit balra shiftelnünk (viszont az összeadás miatt CY-t ismét ellenőrizzük).

Az eredmény, illetve a CY flag előállítása után a lokális regiszterekből az eredményt a bemeneti paramétereket is tartalmazó regiszterekbe másoljuk, majd az F0-ban tárolt CY flaget is

a PSW-ben található eredeti helyére mozgadjuk, ezt követően a rutin visszatér, a main ciklus pedig végtelen ciklusba kerülve várja a program leállítását - a főciklus a függvény futásának előkészítésén kívül nem rendelkezik egyéb funkcionalitással, a szubrutin visszatérte után végtelen ciklusba kerülve várakozik.



Alulírott, Reizinger Patrik (W5PDBR), polgári és büntetőjogi felelősségem tudatában kijelentem, hogy a Mikrokontroller alapú rendszerek (VIAUAC06) tárgy keretein belül számomra kitűzött házi feladatot önállóan, a tárgy során a hallgatók rendelkezésére bocsátott szakirodalom, illetve a <http://www.edsim51.com/> oldalon elérhető Intel 8051 szimulátor, illetve dokumentáció felhasználásával, nem megengedett segítséget/segédeszközt fel nem használva oldottam meg.

Budapest, 2017. 10. 24.

Reizinger Patrik