Mikrokontroller alapú rendszerek
Házi feladat programozói dokumentáció
Papp Dominik Edvárd EAT3D9

Feladatkiírás

„Belső memóriában lévő 16 bites előjeles szám "gyorsszorzása" 10 hatványa szorzóval (1, 10, 100, 1000, 10000), a 10 kitevője az egyik bemenő paramétere a rutinnak (0...4). 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 bites előjeles szám legyen, a túlcsordulás ennek figyelembevételével állítandó. Bemenet: szorzandó címe (mutató), szorzó kitevője (érték), eredmény címe (mutató). Kimenet: 16 bites eredmény (a kapott címen), OV”.

Megvalósítás

A program Little Endian bytesorrendet alkalmaz, azaz a memória alacsonyabbik címén az adat legalacsonyabb helyiértékű bytjea, az LSB van. A legmagasabb helyiértékű byte megnevezése MSB. A program C nyelvi szokásokat követ, avagy függvényekre, szubrutinokra támaszkodik. A program 6 szubrutint definiál és használ. Ezek elnevezési: Main, QuickMultiply, ShiftLeftR6Times, Add2Mem, SaveOV, Complement2. A program kihasználja, hogy 10 felírható 10 = 8 + 2 ként és 10 bármely hatványa előállítható önmagával egész számú szorzás után.. A feladatot megvalósító algoritmusról részletesebb leírás a „QuickMultiply” szubrutin bemutatásánál található.

Bemenő paraméterek

1. R0 – Ez a regiszter tárolja a szorzandó (továbbiakban operandus) címét
2. R1 – Ez a regiszter tárolja az eredmény címét
3. R2 – Ez a regiszter tárolja a szorzó kitevőjét

Kimenő paraméterek

1. R1 – Ez a regiszter tárolja az eredmény címét
2. PSW – Ebben a regiszterben az OV bit helyén jelenik meg, hogy történt-e túlcsordulás. Amennyiben történt, az OV bit értéke 1, egyébként 0.

Egyéb használt regiszterek

- R3 – Az operandus LSB-jét tároljuk itt. Első betöltéskor kerül ide az operandus LSB-je, a program futása közben ez folyamatosan változik!
- R4 – Az operandus MSB-jét tároljuk itt. Első betöltéskor kerül ide az operandus MSB-je, a program futása közben ez folyamatosan változik!
- R6 – Ciklusváltozó „ShiftLeftR6Time" szubrutinhoz. Ennek a regiszternek az értéke mondja meg hogy hányszor shiftelődjön balra az R3 és R4-ben tárolt 16 bites érték
- R7 – Ez a regiszter tárolja, hogy mely bitek voltak valaha 1-esek a PSW R7-be mentésekor

Bemenő paraméterek megadása

- R0 – Az operandus címe a program 38., LSB értéke a 42., MSB értéke a 44. sorában adható meg.
- R1 – A kimenet címe program 39. sorában adható meg.

- R2 – Értéke a program 40. sorában adható meg.

Szubrutinok

Main

A „Main" szubrutin feladata a bemenő paraméterek megadásam elmentése, a feladatot megvalósító szubrutin meghívása és a mikrokontroller végtelen ciklusban való tartása a program lelövéséig. Az program eredménye a 49. sorban érvényes.

SaveOV

A „SaveOV" szubrutin feladata, hogy elmentse az R7 regiszterbe a PSW 1-es bitjeit. A szubrutin logikai vagy-olja a PSW tartalmát R7-hez, így R7-ben azt látjuk, hogy mely bitek voltak valaha 1-esek PSW-ben a mentésekor. Ezt használjuk fel arra, hogy megnézzük történt-e túlcsordulás a programunkban.

ShiftLeftR6Times

A „ShiftLeftR6Times" szubrutin feladata az, hogy az R3(LSB) és R4(MSB) regiszterekben található 16 bites értéket R6-ban meghatározott értékszer balra forgatja úgy, hogy a legkisebb helyiértékű bitre mindig 0 kerül. Ez a szubrutin meghívja „SaveOV" szubrutint, hogy később visszanézhető legyen, a shifteléskor került-e ki 1-es a 16 bites értékből.

Add2Mem

Az „Add2Mem" szubrutin feladata az R3(LSB) és R4(MSB) regiszterekben található 16 érték hozzáadása az R1 regiszter által mutatott memóriacímen lévő 16 bites értékhez.

Complement2

A „Complement2" szubrutin az R3(LSB) és R4(MSB) 16 bites értékből kettes komplemens számot képez.

QuickMultiply

A „QuickMultiply” szubrutin a „fő” szubrutin. Ő vezérli az egész lefolyást és valósítja meg az algoritmust, mely az alábbi:

Azt a tényt használjuk ki, hogy 10 = 8 + 2. Azt is kihasználjuk, hogy a 2 valamely hatványával való szorzás annyit jelent, hogy az adott bináris értéket balra shifteljük annyiszor, amennyi a kitevő értéke és a beshiftelt alsó helyiértékű bitek helyére 0-át írunk. A szubrutin meghívásakor az első, amit csinálunk, hogy az eredmény címén(R1) lévő értéket „kinullázzuk”. Erre azért van szükség, mert minden 10el való szorzás után az értéket el fogjuk tárolni a kimeneten. Ez után kimentjük az operandust az R3(LSB) és R4(MSB) regiszterekbe. Továbbiakban az operandus alatt R3 és R4 16 bites értékét értjük és megnézzük, hogy a kitevő nulla-e. Ha igen, akkor kimentjük az operandust a kimenetre és visszatérünk. Következő lépésként el kell dönteni, hogy az operandus negatív szám-e, mert előjel nélküli szám esetében könnyebb eldönteni, hogy történt-e túlcsordulás. Amennyiben negatív az operandus, azt kettes komplemenseljük. A „Cycle” címke kódja egy hátultesztelő ciklust valósít meg. Ahogy már említettem, egy bináris szám kettővel való szorzása annyit jelent, hogy a számot balra shifteljük eggyel és a legkisebb helyiértékre 0-át írunk. Ez úgy tudjuk megvalósítani, hogy az R6 regiszterbe megadjuk a 2 mely hatványával szeretnénk megszorozni az operandus és a „ShiftLeftR6Times” szubrutin hívásával az operandus annyival meg is szorzódott. Fontos, hogy ha tovább szeretnénk szorozni az operandust, akkor az előző szorzáshoz relatívan kell tenni. Például, ha megszoroztam egyszer 2-vel az operandust, és most meg szeretném szorozni az EREDETI operandust 4-gyel, akkor már csak egyszer kell shiftelni mert az operandus (R3 és R4) az EREDETI operandus 2-vel szorzott változatát tárolja! A ciklus első lépésben nullázza a kimenetet. Második lépésen megszorozza a kimenetet 10-el és végül elmenti a kimenet 10-el szorzott értékét az R3 és R4 regiszterekben. Végsősorban csökkenti R2 értékét 1-el, és amennyiben szükséges, megszorozza a kimenetet még 10-el. Miután sikeresen meghatároztuk az eredményt megnézzük, hogy az EREDETI operandus negatív volt-e, mert ha igen, az eredményt kettes komplemenselni kell a helyes eredmény elérésének érdekében. Ez után már csak annyi dolgunk van, hogy megnézzük volt-e túlcsordulás. Túlcsordulás akkor történt, ha az EREDETI operandus és az eredmény legnagyobb bitje nem egyezik, vagy az R7 7. bitje 1-es. Ha az R7 7. bitje 1-es az azt jelzi, hogy valamely shiftelésnél CY történt, azaz kicsúszott egy 1-es bit a 16 bitből, így helytelen eredményt kaptunk! Amennyiben bármelyik feltétel fennáll, túlcsordulás történt és ezt a PSW 3. bitjének beállításával jelezzük.

Felhasznált források:

Dr. Tevesz Gábor c. egyetemi tanár: Mikrokontroller alapú rendszerek**,** Elektronikus jegyzet,1. fejezet

https://www.refreshnotes.com/2016/03/8051-addition-of-16-bit-numbers.html

https://techetrx.com/8051-microcontroller-tutorials/8051-assembly-language/

http://what-when-how.com/8051-microcontroller/loop-and-jump-instructions-in-8051/

https://www.keil.com

Az utolsó oldalon diagram található!!

Az algoritmus egyszerű blokkvázlata