Komentarai „ant karštųjų“:

PDFo versijos laikas: ***2016-01-17 14:52***
Naujausią PDFo versiją (kol ji neišimta visai) rasite prie paties konspekto, t.y. :
https://docs.google.com/document/d/1UzYBaxLhF8pqvRaBMstj6RDrV5SsCjkfNLw0f4K-r7Y/edit

Koks šio failo tikslas?
Susirinkau iš jūsų idėjų, kuo reikėtų papildyti/pakeisti esamus konspektus, bet kadangi daug kas jau esate atsispausdinę kitus konspektus arba tiesiog per sunku neturint laiko ieškoti skirtumų „kažkur“ tai patogiausia „kažką naujo“ bus rašyti čia, vėliau (prieš perlaikymus) bus galima info išmėtyti kur ji iš tikrųjų priklauso.
Šis failas šiom dienom dar keisis.

Prie reikalo.

Turinys

Kaip apskaičiuoti tą OF flagą?

Darant prielaidą, kad kitus flag'us apskaičiuoti jau mokate, reiškia jau turite:
(pirmas skaičius dvejetainėje) +/- (antras skaičius dvejetainėje) = (rezultatas dvejetainėje).
Iš čia OF apskaičiuojamas taip:

OF Taisyklė: Jei pavertus abu skaičius ir rezultatą į dešimtainius skaičius su ženklu lygybė išlieka teisinga reiškia perpildymo nebuvo ir OF=0, jei neteisinga – perpildymas buvo, vadinasi OF=1.

Ką daryti? Tiesiog visus tris dvejetainius skaičius (2dėmenis, rezultatą) pasiversti į dešimtainę sistemą:

- Jei ženklo bitas yra 0, tai skaičius bus 0..127 ir teigiamas, jį galima pasiversti paprastai tiesiog sudedant vienetinių bitų svorius, pvz 00110001 būtų 64+32+1=97.
- Jei ženklo bitas yra 1, tai skaičius bus -128...-1 ir neigiamas, kadangi neigiamų skaičių gauti „nemokame“ variantas yra pasinaudoti ženklo keitimo galimybe (ženklas keičiamas invertuojant ir pridedant vieną) ir tada jau elgtis kaip tada, jei ženklo bitas yra 0, tik kadangi skaičius neigiamas – jam dar reikės užkabinti minusą. Tai 1111 1110 invertavus gaunasi 0000 0001, pridėjus vieną 0000 0010 (tai yra +2), vadinasi uždėjus minusą -2. Kas reiškia, kad 1111 1110 buvo -2.

Na o turėdami visus kaip dešimtainius su ženklu tiesiog patikriname lygybę ir pagal OF taisyklę jau žinome koks yra OF.

Taupant laiką ir turint „intuiciją ant skaičių“ (žr. PDFą prie **NEW STUFF** pavadinimu KodelTaipYra.pdf jame apie skaičius be ženklo ir su ženklu paaiškinta daugiau) galima elgtis ir gudriau:
Pavyzdžiui žinoti, kad jei vertėmės skaičių iš rėžių [-128;127] į dvejetainę, tai jau ir taip turime skaičių su ženklu (bet rezultato dar nežinojome, tai jį vis tiek reikės verstis).
Dar galima žinoti, kad iš [+128;+256] rėžių pereiti į skaičių su ženklu galima iš skaičiaus atėmus 256...
Na bet čia jau „euristikos“ (*heuristics*, žmogiškai tariant – gudrybės).
Iš tiesų kompiuteris skaičiuodamas OF elgiasi dar gudriau, skaičiuodamas Sign Flag ir Carry Flag pernešimus, bet šitas metodas, kokį aprašiau yra labiau paplitęs ir „laiko patikrintas“.

Kaip paversti dešimtainį 0.013 į kažkurį slankaus kablelio formatą?

Skaičių į dvejetainę verčiame taip, kaip ir bet kokį kitą:

Ženklo bitas: 0, nes teigiamas.

Sveikoji dalis: 0 (dvejetainėje 0).

Trupmeninė dalis: 0.013 per daugybas iš 16 gaunasi:

0.013 * 16 = **0**.208 (**0000**)

0.208 * 16 = **3**.328 (**0011**)

0.328 * 16 = **5**.248 (**0101**)

… dauginame tiek kiek reikia užpildyti mantisę, esmė, kad turime:

0, 0000 0011 0101 …

Normalizuojame, kablelį stumdami per 7 vietas į dešinę:

2^{-7}*1,10101 …

O toliau jau elgiamės įprastai. Eilė -7, į mantisę eina 10101...

Kodėl skaičiavimo sistemos „pozicinės“?

Nes kitaip nei romėniškoje skaitmens įtaka skaičiui priklauso nuo jo vietos skaičiuje. 12321 atveju kairiojo vienetuko pakeitimas į dvejetą turėtų 10000 kartų didesnę įtaką nei dešiniojo, t.y. skaitmens pozicija lemia, kokią įtaką skaičiaus vertei jis turi.

Ar valdymo perdavime segmento prefiksai turi įtakos?

Priklauso nuo to, ką skaičiuojate. Tarkim turime netiesioginį CALL, tuomet kur padėtas naujos komandos adresas (kokiame segmente) mums nurodo ne kas kitas, o operandas atmintyje pagal adresavimo baitą, tai vietai atmintyje rasti yra taikomos adresavimo baito taisyklės, bet naujai gautom CS:IP reikšmėm segmento pakeisti neišeina, nes baigęs vykdyti komandą procesorius sekančios komandos duomenis ima tiesiog iš ten kur dabar rodo CS:IP ir visa kita jo nebedomina.

Pažiūrėkite mano konspekto 15psl. lentelę, apie adresavimo atvejus.

Iš ko susideda mašininis kodas?

- Prefiksai lietuviškai tariant priešdėliai, kaip lietuvių kalbos žodyje (jų gali nebūti)
- Operacijos kodas (privalo būti)
- Adresavimo baitas (jo gali nebūti)
- Poslinkio baitai (nuo 0 iki 2 baitų)
- Betarpiškas operandas (nuo 0 iki 2 baitų)

Prefiksai iš esmės yra 3 tipų: LOCK, segmento keitimo, eilutinių komandų pakartojimo (REPai).
Operacijos kodas yra esminė komandos dalis, kuri identifikuoja, kokia tai tiksliai yra komanda.

95% atvejų užtenka žinoti tiek, kad:
Jei pirmas baitas nėra 26,2E,36,3E,F0,F1,F2 tai pirmas baitas yra operacijos kodas.
Jei yra kažkuris iš šitų, tai tada patikriname – gal antrasis irgi yra kažkuris iš šitų (gali būti daugiau nei vienas prefiksas, esant jų daugiau nei dviem prasideda neapibrėžtumai).

Kai jau sutinkame baitą, kuris nėra prefiksas tai žinome operacijos kodą, iš jo nustatome kokia tai yra komanda.
Jei tereikia rasti „operando atmintyje absoliutų/efektyvų“ adresą tai esminiai du atvejai:
Operacijos kodas yra A0,A1,A2,A3 ?
Jei taip – nėra adresavimo baito, iškart eina poslinkio baitai lyg tiesioginiame adrese (kai mod=00, r/m=110)
Jei ne – tai interpretuojame adresavimo baitą ir tada jau kiek reikia poslinkio baitų.

Betarpiškas operandas (paprasčiau tariant konstanta, angliškai: trumpinama imm, pilnas variantas: immediate).

MPL: Kodėl negalima rašyti A=MBR+0; B=MBR+0; C=MBR+0; D=MBR+0;

Nes ne į visus 4 registrus (A,B,C,D) eina magistralės iš sumatoriaus. MBR reikšmę į juos galima pasiųsti tiesiog A=MBR; B=MBR; C=MBR; D=MBR;

MPL: Ar galima pasiųsti kelias registrų reikšmes ne per sumatorių?

Taip, jei tik tai leidžia nutiestos magistralės, pavyzdžiui į A, B, D registrus MBR reikšmę galime pasiųsti nes tam turime magistrales, tai užrašoma, ir telpa net vienoje komandoje:

A=MBR; B=MBR; D=MBR;

MPL: Ką svarbiausia žinoti apie pociklius?

Kiekviena MPL komanda vyksta 3 pocikliai:

1. Reikšmių pasiuntimas tarp registrų ir į sumatorių vykdomi pirmame.

2. Reikšmių sumavimas vykdomas antrame.
3. Reikšmių apsikeitimas tarp MBR registro ir atminties vykdomas trečiame.

Svarbiausia dalis tai riba tarp pirmo ir antro pociklio. Būtent dėl jos negalima rašyti

X=15; MBR=X+1; vienoje komandoje, nes tiek pasiuntimas į sumatorių, tiek 15 → X bus vykdoma tam pačiam pociklyje ir tai sukelia neapibrėžtą situaciją, kai rezultatas MBRe priklausys nuo „race condition“ (kas pirmiau pateks?)