X86Prime - opsamling

Finn Schiermer Andersen,

Ekstern lektor, DIKU

x86prime - instruktioner

1. Aritmetik - kun register/register og konstant/register

    - 2-komplement aritmetik: addq, subq, mulq, imulq, sarq, shrq, salq
    - Bitvis logik: andq, orq, xorq
    - Addresse-aritmetik: leaq, all formater

2. Kontrol

    - sammenligning og betinget hop i EN instruktion: CBcc (jmp,cbe,cbne,cble,cbl,cbge,cbg)
    - funktionskald der bruger registre, ikke stakken: call, ret
    - stop verden! jeg vil af: STOP instruktion. Men også retur til adresse <= 0.

3. Data flytning (egentlig er det kopiering, men...)

    - konstant til register: movq $imm, %reg
    - register til register: movq %reg, %reg
    - register til lager (store): movq %reg, displacement(%reg)
    - lager til register (load): movq displacement(%reg), %reg

Med en regulær indkodning

Pointe: Instruktioner er også data.

```
00000000 00000000                    stop
00000001 0000ssss                    ret s
0001aaaa ddddssss                    register/register arithmic: op s,d
00100001 ddddssss                    movq s,d
00110001 ddddssss                    movq (s),d
00111001 ddddssss                    movq d,(s)
0100cccc ddddssss pp...32...pp       cb<c> s,d,p
01001110 dddd0000 pp...32...pp       call p,d
01001111 00000000 pp...32...pp       jmp p
0101aaaa dddd0000 ii...32...ii       imm/register arithmetic: op i,d
01100100 dddd0000 ii...32...ii       movq $i,d
01110101 ddddssss ii...32...ii       movq i(s),d
01111101 ddddssss ii...32...ii       movq d,i(s)
10xxxxxx                             leaq (various forms)
1111cccc dddd0000 ii...32...ii pp...32...pp cb<c> $i,d,p
```

dddd og ssss er registre. aaaa angiver aritmetisk operation

ii...32...ii er et 32-bit 2-komplement tal

pp...32...pp er en 32-bit adresse

Indkodning af leaq

```
10000001 ddddssss                    leaq (s),d
10010010 dddd0000 zzzzvvvv             leaq (,z,(1<<v)),d
10010011 ddddssss zzzzvvvv             leaq (s,z,(1<<v)),d
10100100 dddd0000 ii...32...ii         leaq i,d
10100101 ddddssss ii...32...ii         leaq i(s),d
10110110 dddd0000 zzzzvvvv ii...32...ii   leaq i(,z,(1<<v)),d
10110111 ddddssss zzzzvvvv ii...32...ii   leaq i(s,z,(1<<v)),d
```

zzzz angiver et register.

vvvv angiver hvor meget der skal skiftes.

Bemærk sammenhængen mellem hvilke operander der indgår i beregningen og de 3 mindst betydende bits i den første byte :-)

Bemærk også at enhver instruktions længde kan bestemmes alene fra de første 4 bits.

Indkodning af betingelse

0100cccc ddddssss pp...32...pp cb<c> s,d,p
1111cccc dddd0000 ii...32...ii pp...32...pp cb<c> $i,d,p

cccc: betingelse
0000 e 1000 a
0001 ne 1001 ae
0010 <reserved> 1010 b
0011 <reserved> 1011 be
0100 l 1100 <reserved>
0101 le 1101 <reserved>
0110 g 1110 <reserved>
0111 ge 1111 <reserved>

Et eksempel: sammenlign %r10 med %r11 og hop til 0x407, hvis %r10 er mindre end eller lig med %r11

01000101 10111010 00000111 00000100 00000000 00000000 cble %r10,%r11,0x407

Inkodning af aritmetisk operation

0001aaaa ddddssss register/register arithmic: op s,d
0101aaaa dddd0000 ii...32...ii imm/register arithmetic: op i,d

aaaa: aritmetisk operation
0000 add
0001 sub
0010 and
0011 or
0100 xor
0101 mul
0110 sar
0111 sal
1000 shr
1001 imul

Et par eksempler:

00010001 10011010 subq %r10,%r9
01010001 10010000 11111111 00000000 00000000 0000000 subq $255,%r9