Exjobbsförslag: Modellera processors exekveringshastighet

Håkan T. Johansson
f96hajo@fy.chalmers.se

16 februari 2003

1 Bakgrund

Som exjobb har jag studerat och implementerat delar av ett system för att läsa in, förstå, ändra i och skriva ut program för Intel x86-processorer [1]. Målet är att kunna modifiera programmen på ett sådant sätt att de kan köra fortare, d.v.s. beräknar precis samma sak, men med instruktioner som kan processas på kortare tid.

Slutsatsen var att ett sådant system borde vara möjligt att konstruera (svårigheterna ligger framför allt i att kunna "förstå" (analysera) ett program). Implementationen (kallad `exmhcg`[1]) har kommit så pass långt att den kan läsa in program, och fullt hantera (skriva ut nya varianter av) ett fåtal väldigt enkla program.

En viktig beståndsdel (som ännu ej finns) är naturligtvis trimmaren (optimeraren), utan vilken det inte är mycket mening med resten av systemet. För att trimmaren skall kunna göra goda bedömningar av hur snabbt en given instruktionsföljd kan exekveras av en processor krävs en ganska detaljerad modell av processorn (inte med avseende på resultaten av olika operationer, utan hur länge varje instruktion uppehåller sig i varje del av processorns pipeline). Utredning av relevanta problem och implementation är exjobbsförslaget.

2 Användningsområden

En processormodell kan användas dels som en del i `exmhcg`, d.v.s. feedback-funktion för en kod-trimmare. Dels kan en kodhastighetsberäknare användas för att ge en programmerare data för att bedöma effektiviteten och finna flaskhalsar hos ett stycke assembler-kod utan att behöva testköra koden i fråga.

[1] Släppt under GPL, se URL vid [1] för källkod och rapport.

3 Självständighet och avgränsning

Det är viktigt att det förslagna exjobbet inte blir beroende av någon utomstående okontrollerbar faktor, t.ex. att något vidare arbete utförs på `exmhcg`, och att omfånget är klart avgränsat.

3.1 Avgränsning

Om man väljer att basera en implementation på `exmhcg` så finns redan en rutin för att läsa in assembler-kod (i form av körbara program). Denna ger instruktionerna i ett internt format (objektorienterat, C++). Även någon form av utdata-presention kan vara lämplig. Mycket enkel (och ganska ful) finns, både i text-form och som en (pre-alfa) java-applet.

Arbetet avgränsas lämpligen också till att modellera en (eller flera snarlika) medlemmar i processorfamiljen. Naturligtvis bör man koncentrera sig på en, men samtidigt bör en eventuell implementation göras sådan att den inte hindrar utan direkt inbjuder till modellering av övriga syskon och kusiner (dvs även icke-Intel-konstruktioner). Ett basförslag är att 686-processorer studeras. I jämförelse är tidigare processorer antagligen busenkla att modellera och processorn använder många metoder som även senare modellers hastighet påverkas av, varför den inte är alltför gammal och omodern för att vara intressant.

3.2 Självständighet

Självständigheten kommer i att projektet över huvud taget inte behöver ha någon anknytning till `exmhcg`. Och om det skulle ha det, så är det inte nödvändigt att synka med eventuella senare versioner.

3.2.1 Tvådelat

Som beskrivs nedan finns dels en hel del strategiska problem att lösa, vilka inte kräver att någon körbar kod skrivs, vilket säkert kan vara ganska så mycket jobb. Dels (för att vara till nytta för en kod-trimmare) så är naturligtvis en implementation av processormodellen lämplig. Så arbetet kan delas in i en teoretisk och en med praktisk del, där den praktiska även kan vara valfri.

4 Problem

Utan att gå in på detaljer, kan en del lösa tankar beträffande en del av de problem som behöver lösas framkastas:

- För en processor med pipeline, hur avgörs hur många klockcykler varje instruktion tar? Knappast från det att den börjar avkodas tills att den är helt färdigbehandlad, eftersom detta överlappar med andra instruktioner.
- Ytterligare data angående programmets uppförande än enbart instruktionerna kan vara viktig för att avgöra exekveringshastighet. Till exempel är andelen felaktigt förutspådda villkorliga hopp förödande. Således kan sådan information vara bra att ha från utanförliggande program.
- …

Referenser

[1] Håkan T. Johansson, *Tuning Intel x86 Executables*, Exjobb CTH, `http://www.fy.chalmers.se/~f96hajo/exmhcg`, 2002.

[2] Agner Fog, *How to optimize for the Pentium family of microprocessors*, `http://www.agner.org/assem/pentopt.zip`, 2000.