Opgavenserie 7: Assembly

Systeemarchitectuur 1

1 april 2011

De uitwerkingen van deze opgaven graag inleveren uiterlijk op 11 april 2011 om 23.59 uur, per e-mail aan Wouter Geraedts (w.geraedts@student.ru.nl). Ik corrigeer alleen uitwerkingen in **platte tekst** of **PDF** die in een e-mail met onderwerp „[PnP] Opgave 7" verstuurd worden. Ik word extra vrolijk van uitwerkingen die met LaTeX zijn gemaakt.

Als je drie van de vijf opgaven goed hebt beantwoord, telt jouw uitwerking mee voor de bonus bij het tentamen.

Maak zonodig zinvolle aannames. Beredeneer je antwoord; laat tenminste zien dat je het antwoord niet hebt gegoogelt.

1. Een computer heeft een twee-level-cache. Neem aan dat 60 % van de RAM-verwijzingen al in de eerste level gevonden worden ("hit"), 35 % in de tweede level, en 5 % helemaal niet ("miss"). Voor toegang tot de eerste-level-cache is 5 nsec nodig, voor de tweede level 15 nsec, en voor de derde level 60 nsec. Hoe lang duurt een RAM-toegang gemiddeld? (Let op dat een toegang tot level 2 pas begint na een miss van level 1.)

2. Een computer met een pipeline van vijf stappen handelt voorwaardelijke sprongen af door drie stappen in de pipeline te wachten. Hoeveel performance gaat verloren als in een programma 20 % van alle instructies voorwaardelijke sprongen zijn? (Negeer alle andere bronnen van performanceverlies.)

3. Neem aan dat een bepaalde processor tot 20 instructies van te voren kan fetchen. Gemiddeld zijn echter vier van deze instructies (voorwaardelijke) sprongen. Stel dat de branch predictor 80 % van de sprongen correct afhandelt. Hoe groot is de kans dat de predictor de correcte route volgt?

4. (a) Bij een bepaalde processor heeft elke instructie 16 bits. De processor heeft 8 registers. Stel nu dat elke instructie n registers als operanden heeft. Hoeveel instructies kan de processor dan hebben, voor $n = 0; 1; 2; 3$?

   (b) Realistischer is het aan te nemen dat de processor voor elke n een aantal instructies heeft. Als de processor 32 instructies zonder en bovendien 32 instructies met 3 operanden heeft, hoeveel instructies met 2 operanden zou men dan nog kunnen toevoegen?

5. Beschouw het volgende programma:

```
for ( i = 0 ; i < 5 ; i++ )
        for ( j = 0 ; j < 5 ; j++ )
                for ( k = 0 ; k < 5 ; k++ )
                        a[j] = a[j] + i * k;
```

   Beschrijf hoe het principe van "locality" op deze code van toepassing is. Door de volgorde van de for-loops te wijzigen, hoe kun je deze code sneller laten uitvoeren?