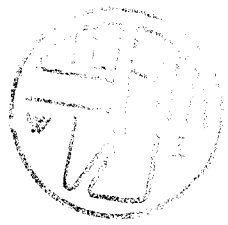
Computer organisatie
21 december, 2005, 9:30-12:30



- Dit examen bestaat uit 5 vragen met subvragen. Elke subvraag is 10 punten waard.
- Begin elke vraag op een nieuwe bladzijde. Zet overal je naam en studentnummer boven.
- Gebruik van boeken of ander studiemateriaal is niet toegestaan, een rekenmachine wel.
- Houd je antwoorden bondig. Schrijf niet met potlood of rode inkt.

Vraag 1

(a) Druk de CPU-tijd (in seconden) uit in CPI en IC (waarbij IC = 'het aantal uitgevoerde instructies').

(b) Een programma-executie op een 1GHz processor heeft een CPU tijd van t seconden en een CPI van c. Is het mogelijk om precies te achterhalen hoeveel instructies tijdens deze executie werden uitgevoerd? Zoja, hoeveel dan? Zo nee, waarom niet?

Een processor heeft een kloksnelheid van 1 GHz en de specificatie geeft de volgende timing informatie:

Type	CPI
ALU operations	1
Floating point (FP) operations	x (onbekend)
Branches	3
Loads and stores	2

(c) In een bepaald programma zijn 30% van de operaties ALU instructies, 5% FP instructies, 25% Branches, 20% Loads en 20% Stores. Geef een formule (uitgedrukt in x) om de gemiddelde CPI te berekenen uit de CPIs uit de tabel.

Het volgende (andere) programma wordt nu ook uitgevoerd op deze machine (gegeven in pseudo-assembly):

```
Code                      Description
-----------------------------------------------------------------
1.      LOAD R1,#10       ; laad in register R1 de waarde 10
2.  L1: DIV F1,F2         ; deel (floating point register) F1 door F2 en zet resultaat in F1
3.      SUB R1,R1,#1      ; trek 1 af van de inhoud van R1 - resultaat wordt in R1 gezet
4.      BNEZ R1,L1        ; als inhoud van R1 niet gelijk is aan 0, spring naar L1
```

Neem aan dat F1 initieel de waarde 100.0 bevat en F2 gelijk is aan 2.0.

(d) (i) Geef van *elk* van de instructies aan hoe vaak deze wordt uitgevoerd. (ii) Beschrijf *in één zin* wat dit programma doet (d.w.z. welke berekening wordt uitgevoerd?).

We zijn nieuwsgierig naar x, de waarde van de CPI voor de FP operaties. Ga er vanuit dat DIV de enige FP instructie is. Neem tevens aan dat bovenstaande code (de pseudo-assembly) wordt uitgevoerd op een processor met een kloksnelheid van 1 GHz. Om x te vinden meten we de CPU time van bovenstaand programma. Deze blijkt T seconden te zijn.

(e) Bepaal, gebruikmakend van uw eerdere antwoorden, een waarde voor x (uitgedrukt in T).

Vraag 2

(a) Geef een expressie in termen van AND en OR voor wat wordt uitgedrukt in de volgende waarheidstabel.

A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(b) Geef ook een digitaal circuit voor deze waarheidstabel.

(c) In een bepaalde machine wordt de integriteit van codewoorden bewaakt door middel van de Hamming code die in het college is behandeld. We lezen het woord '0 0 1 0 1 1 1 0 0 0'. Laat zien of er een (single) bit error is opgetreden (uitgaande van even pariteit).

Vraag 3

Geef van de volgende uitspraken aan of ze goed of fout zijn:

(i) Een architectuur met een 32 bits memorybus kan meer dan 4GB adresseren, mits de 'kleinst adresseerbare eenheid' groter is dan een byte.

(ii) Uniform Memory Access (UMA) machines zijn makkelijker te bouwen voor grote aantallen processoren dan Non-Uniform Memory Access (NUMA) machines.

(iii) Om een ISA instructie uit te voeren heb je *altijd* meer dan 1 microinstructies nodig.

(iv) Als het aantal datalijnen naar de memorychips verdubbelt, wordt de latency naar geheugen verminderd.

(v) Om echt baat te hebben van DMA, is een cache nodig bij de CPU.

(vi) Als het aantal datalijnen naar de memorychips 64 is, is een cache line van minder dan 8 bytes geen zinnige keuze.

(vii) Als het aantal datalijnen naar de memorychips 64 is, is een cache line van meer dan 8 bytes geen zinnige keuze.

(viii) Een goedkope manier om een NUMA machine om te bouwen tot een UMA is door aan elk van de CPUs een grote cache toe te voegen, zodat alle data lokaal kan worden gevonden.

(ix) Sequential consistency is makkelijker te garanderen in on-chip multiprocessing dan in distributed memory multicomputers of grid computers.

(x) We 'saven' de *program counter* op de stack bij een functie-aanroep, zodat we bij terugkeer van de functie aanroep (return) weten waar de *parameters* en *lokale variabelen* van de aanroepende functie zich bevinden op de stack..

Vraag 4

(a) Welke cache heeft gemiddeld een hogere missrate: een direct-mapped cache of een 2-way set associative cache? Waarom?

We ontwerpen een cache van 64 KByte voor een byteadresseerbare machine met 32-bits adressen en 8 bytes in een woord. De cache is direct mapped. We moeten alleen nog beslissen wat de grootte van een cacheline wordt: 16 woorden, of 32 woorden.

(b) Stel dat de cacheline 16 woorden bevat. Op welke cacheline komt dan het byte dat zich op hexadecimaal adres `00 00 ca fe` bevindt? Wat is de tag?

We hebben gemerkt dat het programma dat op deze machine draait heel vaak een serie geheugenlocaties leest die er als volgt uitziet (alle adressen *decimaal*):

`768, 131970, 770, 131972, 772, 131974, 774, 131976, 776, 131978, ...`

(patroon: steeds $768 + 2n$ gevolgd door $131970 + 2n$ voor $n = 0, 1, 2, ...$

(c) Op basis van (alleen) deze gegevens, kies je voor 16 of 32 woorden in een cacheline, of maakt het niet uit? Motiveer je antwoord.

(d) Zou je keuze hetzelfde zijn geweest als het een two-way set associative cache betrof (zelfde cachegrootte en woordgrootte)? Licht je antwoord toe.

Vraag 5

(a) In superscalars wordt veel gebruik gemaakt van register renaming. Leg het principe van register renaming uit en laat met behulp van een voorbeeld zien hoe het kan leiden tot meer parallelisme en kortere executietijd.

(b) Register renaming is niet geschikt voor het verwijderen van stalls die het gevolg zijn van RAW (read after write) hazards. Noem een techniek (behandeld in het college) die soms helpt om RAW stalls te voorkomen. **Deze vraag is bonus. Foute beantwoording levert geen punt aftrek op. Juiste beantwoording is goed voor extra punten.**