Tentamen Computerarchitectuur

Woensdag 18 januari 2017, 10:00 - 13:00 uur

Examinator: dr. K. F. D. Rietveld

- Het tentamen is **gesloten boek**, dus het is niet toegestaan om het tekstboek, slides of eigengemaakte aantekeningen te gebruiken.
- Alleen rekenmachines waarin geen teksten kunnen worden opgeslagen zijn toegestaan. Het gebruik van grafische rekenmachines is **niet** toegestaan.
- De vragen mogen worden beantwoord in het Nederlands en Engels.
- Beargumenteer al uw antwoorden. Aan antwoorden zonder uitleg of uitwerking worden geen punten toegekend.

- Bij het inleveren van het gemaakte werk zal u worden verzocht uw naam, studentnummer en aantal ingeleverde bladen te noteren op de presentielijst.

- Het aantal opgaven is 5 met een totaal van 21 onderdelen. Het tentamen bestaat uit 4 pagina's.
- Bij elk onderdeel staat tussen vierkante haken het aantal te behalen punten aangegeven. Het totaal aantal te behalen punten is 100.

Opgave I – Performance [15 punten]

We gaan uit van een RISC machine met een klokfrequentie van 2.4 GHz. Door middel van een representatieve benchmark zijn de operaties op deze machine gekarakteriseerd:

Operatie	CPI	Frequentie
Integer ALU	1	30%
FP ALU	4	10%
Branches	3	20%
Loads	2	30%
Stores	2	10%

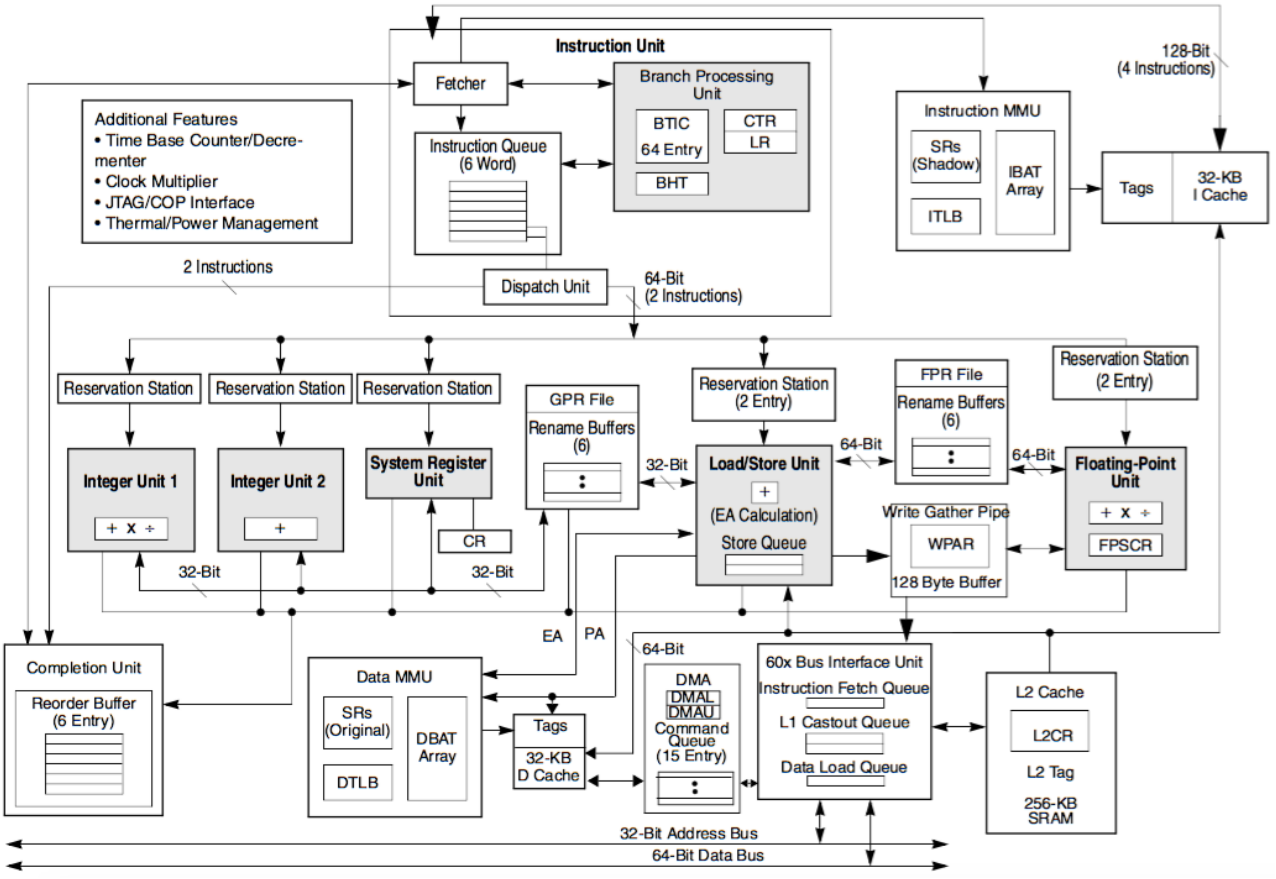
a. [5 punten] Wat is de overall CPI die wordt behaald op deze machine met deze instructiemix?

b. [4 punten] Hoeveel miljoen instructies per seconde zullen er (gemiddeld) worden uitgevoerd op deze machine met deze instructiemix?

c. [6 punten] Door gebruik te maken van pipelining reduceert het gemiddelde aantal klokperioden voor een FP ALU operatie tot 2 klokperioden. Echter load en store instructies gaan nu ieder gemiddeld 3 klokperioden kosten. Geef een kwantitatief advies om in dit geval wel of niet gebruik te maken van pipelining.

Opgave II - Processor organisatie [15 punten]

In deze opgave bekijken we de IBM PowerPC 750CL RISC microprocessor, onder andere gebruikt in de Nintendo Wii.



Bron: IBM PowerPC 750CL RISC Microprocessor User's Manual.

a. [4 punten] Hoeveel functional units bevat deze architectuur? Benoem alle functional units in uw antwoord.

b. [3 punten] Is dit een "superscalar" architectuur? Waarom wel/niet?

c. [3 punten] Hoeveel nieuwe instructies kunnen per klokperiode worden gefetched en hoeveel kunnen er per klokperiode worden opgestart ("issue")? Geef aan waaruit dit blijkt.

d. [5 punten] Dankzij de "completion unit", waarin instructies in-order worden afgehandeld, kan deze processor een "precise exception model" aanbieden. Wat houdt dit in en wat zouden de consequenties zijn wanneer het "exception model" niet "precise" zou zijn geweest?

Opgave III - Caching [25 punten]

a. [5 punten] Omschrijf de drie soorten cache misses die kunnen optreden.

De processor in de vorige opgave heeft een 8-way set associative L1 data cache van 32KB. Een cache line is 32 bytes groot.

b. [5 punten] Bereken het aantal sets in deze cache en het aantal benodigde bits om de cache te kunnen indexeren.

c. [5 punten] Ga ervan uit dat deze cache FIFO als replacement-strategie gebruikt en de cache in het begin leeg is. We voeren een programma uit dat de volgende geheugenreferenties genereert gegeven in blokadressen ("block addresses"): 0, 1, 3, 128, 256, 5, 512, 1024, 2048, 516, 4096, 8192, 16384, 3, 0, 5. Geef voor elke geheugenreferentie aan: in welke set deze referentie uitkomt, of dit een miss of hit betreft, en of er een cache line uit de set moet worden vervangen (zo ja, welke).

We willen graag uitzoeken of een cache line size anders dan 32 bytes zal leiden tot betere resultaten. Stel nu dat deze cache een hit time heeft van 2 klokperioden en in geval van een L1 miss wordt de volgende laag van de geheugenhiërarchie aangesproken. Deze laag kan 16 bytes opleveren in 103 klokperioden, 32 bytes in 106, en 64 in 112 klokperioden. Merk op dat deze tijden worden bepaald aan de hand van een vaste overhead (100 perioden) plus een variabele overhead afhankelijk van het aantal te lezen bytes. Middels een representatieve benchmark zijn de volgende miss rates vastgesteld: 4.02% bij een cache line size van 16 bytes, 2.56% bij 32 bytes en 2.49% bij 64 bytes.

d. [5 punten] Geef een advies welke cache line size (16, 32 of 64 bytes) het meest geschikt zou zijn voor deze workload op basis van een berekening van de "average memory access time" in klokperioden.

e. [5 punten] Volgens de documentatie van de chip van Opgave II maakt de implementatie van de L1 cache gebruik van de volgende optimalisatie: "A miss in the L1 cache causes a block reload from either the L2 if the block is in the L2 or from main memory. The critical doubleword is accessed first and forwarded to the load/store unit." Leg in je eigen woorden uit wat deze optimalisatie inhoudt en welke voordelen dit heeft.

Opgave IV - Instruction-Level Parallelism [25 punten]

In deze opgave bekijken we een 5-stage micro-architectuur (Instruction Fetch (IF), Decode and Register Fetch (ID), Execute (EX), Memory (MEM), Write back (WB)). Elke stage kost 1 klokperiode. Er is *wel* forwarding geïmplementeerd. Het bepalen of een branch wel of niet moet worden genomen wordt gedaan in de ID stage, de rest van de stages worden voor branch instructies niet uitgevoerd. Gegeven is de volgende code:

```
Loop:   ADD   R4,R2,R0        ; R4 <- R2 + R0
        LW    R2,0(R1)        ; R2 <- Mem[R1+0]
        ADD   R2,R2,#10       ; R2 <- R2 + 10
        SW    R4,0(R1)        ; Mem[R1+0] <- R4
        SUB   R1,R1,#4        ; R1 <- R1 - 4
        BNEZ  R1,Loop         ; If R1 != 0: jump to Loop
```

a. [6 punten] Laat voor één iteratie van de loop zien hoe de pipeline per kloktik wordt gevuld. Indien er een stall optreedt, benoem dan ook de reden waarom.

Wanneer we code in andere volgorden gaan uitvoeren kunnen we door WAR en WAW hazards in de problemen komen.

b. [4 punten] Benoem alle mogelijke WAW en WAR hazards in de gegeven code.

Tomasulo's algoritme gebruikt in feite register renaming om WAW en WAR hazards te elimineren. We hebben tot onze beschikking een oneindig aantal tijdelijke registers: `T0`, `T1`, `T2`, ..., enz.

c. [7 punten] Werk voor **4** iteraties van de bovenstaande loop uit hoe register renaming door Tomasulo's algoritme wordt toegepast.

d. [3 punten] Leg aan de hand van je antwoorden onder **b** en **c** uit hoe Tomasulo's algoritme WAR en WAW hazards elimineert.

e. [5 punten] Bespreek de overeenkomsten tussen loop unroll uitgevoerd door een compiler en dynamic scheduling zoals toegepast door Tomasulo's algoritme. Geef voor beide methoden een nadeel.

Opgave V - Parallelism [20 punten]

Kort voordat multi-core CPUs de standaard werden, was het de norm om een single-core CPU uit te rusten met meerdere hardware-matige threads middels Simultaneous Multithreading (SMT). Gezien processoren al werden geïmplementeerd als multiple-issue, dynamically scheduled processors, vergde dit maar een kleine uitbreiding

a. [5 punten] Leg uit hoe SMT wordt geïmplementeerd bovenop een multiple-issue, dynamically scheduled processor en waarom dit maar een kleine uitbreiding behoeft.

b. [5 punten] We hebben tot onze beschikking een single-core processor met 4 threads. Wanneer alle threads worden gebruikt, wordt de behaalde IPC grofweg verhoogd van 1.5 tot 4.6. Ga uit van een programma waarvan 55% van het werk kan worden verdeeld over 4 threads, de rest van het programma wordt afgehandeld in een enkele thread. Wat is de maximum speedup die voor dit programma kan worden behaald door gebruik te maken van threads?

We bekijken nu een 8-core CPU, zonder threads, die geklokt is op 2.8 GHz. Elke core is uitgerust met een SIMD unit van 512 bits breed. Een SIMD instructie kost gemiddeld 3 klokperioden. Een floating-point getal bestaat uit 8 bytes.

c. [5 punten] Bereken de theoretische pieksnelheid in GFLOP/s van deze multi-core CPU.

d. [5 punten] Een belangrijk probleem op grootschalige multi-core architecturen is het probleem van cache coherence. Leg de problematiek uit aan de hand van een voorbeeld en omschrijf één van de mogelijke oplossingen die worden geïmplementeerd.