CHALMERS

2013-12-09

Exam in DAT 105 (DIT 051) Computer Architecture

Time: December 16, 2013 14 – 18 in V (Chalmers students) and H (Gothenburg University students)

Person in charge of the exam: Angelos Arelakis, Phone: 0763-103557

Supporting material/tools: Chalmers approved calculator.

Exam Review: On January 8, 2014 10-12 in Room 4128

Grading intervals:

- **Fail**: Result < 24
- **Grade 3**: 24 <= Result < 36
- **Grade 4:** 36 <= Result < 48
- **Grade 5:** 48 <= Result

NOTE 1: Bonus points from Real-stuff studies and Quizzes will be added to the exam results

NOTE 2: Answers must be given in English

GOOD LUCK!
Per Stenström

CHALMERS UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
412 96 Göteborg
Visiting address: Rännvägen 5
Phone: 031-772 1761 Fax: 031-772 3663
Org. Nr: 556479-5598
E-mail: pers@chalmers.se

[General disclaimer: If you feel that sufficient facts are not provided to solve a problem, either 1) ask the teacher when he visits the exam, or 2) make your own additional assumptions. Additional assumptions will be accepted if they are reasonable and required to solve the problem. Always make sure to motivate your answers.]

ASSIGNMENT 1

The tables below show the relative instruction frequency and CPI on two machines (A and B) and a reference machine (R) with the **same** Instruction Set Architecture (ISA) for two single-threaded programs, P1 and P2, respectively, where P1 executes twice as many instructions as P2. The operating frequencies of the three machines (A, B and R) are also shown.

Program P1	**A (Frequency in percent/CPI)**	**B**	**R**
Arithmetic/Logic	40/1	40/2	40/2
Loads	25/2	25/3	25/2
Stores	10/1	10/1	10/2
Branches (untaken)	8/1	8/1	8/2
Branches (taken)	12/3	12/3	12/2
Misc.	5/1	5/1	5/2

Program P2	**A (Frequency in percent/CPI)**	**B**	**R**
Arithmetic/Logic	20/1	20/2	20/2
Loads	35/2	35/3	35/2
Stores	20/1	20/1	20/2
Branches (untaken)	8/1	8/1	8/2
Branches (taken)	12/3	12/3	12/2
Misc.	5/1	5/1	5/2

Clock freq. (GHz)	
Machine A	**1**
Machine B	**1.2**
Machine R	**1**

1A) Using execution time as the performance measure, derive the geometric mean of the performance of machine A and B, using R as a reference. (**3 points)**

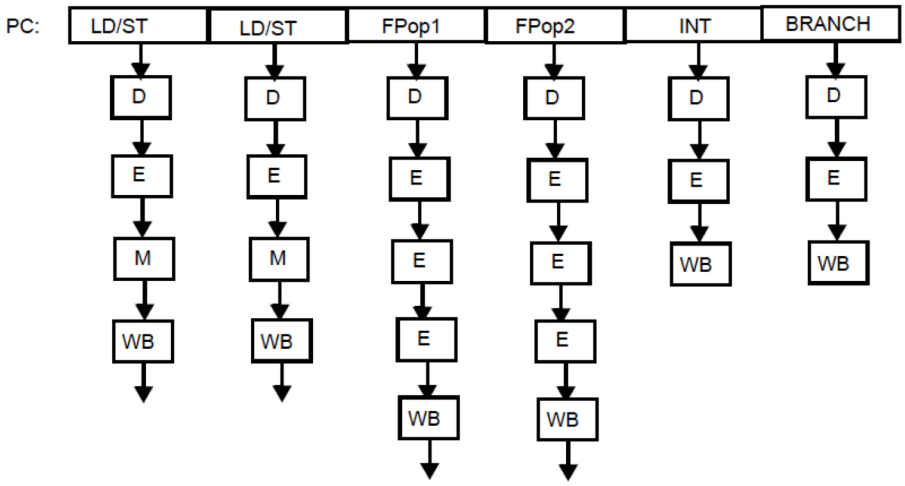
1B) For a third program, let's call it P3, the average CPI is 1, 2, and 1, on machine A, B, and R, respectively and P3 executes twice as many instructions as P1. What is the geometric mean of the performance of A and B, using R as a reference for programs P1, P2 and P3? (**3 points)**

1C) What is the maximum improvement in performance that can be achieved for Machine A and B if the execution of Arithmetic/Logic instructions could be accelerated indefinitely on program P1?
(**3 points)**

1D) Now assume a homogeneous multicore system on a 1 cm^2 die built from multiple instances of either machine A or B. Let us assume that machine A and B occupy 0.1 cm^2 and 0.05 cm^2 each, respectively. Now, given that throughput is our performance metric, what is the throughput of the multicore built from machine A versus the one built from machine B, when running multiple instances of Program P1? (**3 points)**

ASSIGNMENT 2

We consider in this assignment a VLIW architecture that can issue two memory, two floating-point, one integer, and one branch instruction each cycle according to the pipeline organization below. There are no forwarding units.



2A) How many cycles does it take to resolve a RAW dependency between instructions of different types, i.e., between a floating-point operation and a subsequent floating point operation, between a floating-point operation and a subsequent store operation, between a load operation and a subsequent floating point operation, and between a store operation and a subsequent load operation? (**3 points)**

2B)
Consider the following simple computation:

```
LOOP: L.D   F0, 0(R1)
      ADD.D F4,F0 F2
      S.D   F4,0(R1)
      SUBI  R1,R1, #8
      BNE   R1,R2, LOOP
```

We want to use software pipelining to execute the loop on the VLIW architecture. Derive the kernel of a *software-pipelined loop* by filling out the schedule table below for as many cycles needed to fill the "software pipeline" (i.e., not the VLIW pipeline). (**3 points)**

	ITE 1	ITE 2	…
INST 1			
INST 2			
…			

2C)
Under what circumstances do WAR hazards **not** occur? In case they occur, show how *Rotating Registers* can be used to avoid them by first explaining how Rotating Registers work and then by showing how the kernel is modified to make use of them. (**3 points)**

2D)
Consider the following code:

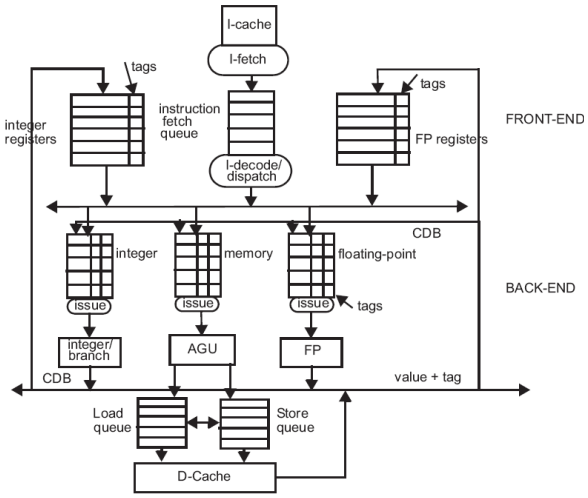
```
LW  R4, 0(R1)
ADDI R6,R4,#1
BEQ  R5,R4,LAB
LW   R6,0(R2)
LAB: SW R6, 0(R1)
```

Use *Predicated Instructions* to eliminate the branch in the code above and clearly define the semantics of the used predicated instructions. (**3 points)**

ASSIGNMENT 3

The diagram below shows a pipeline using the Tomasulo algorithm for dynamic instruction scheduling. The pipeline consists of 5 stages (apart from Instruction Fetch): Dispatch, Issue, Execution, Cache access and CDB write. Assume that all integer and branch instructions execute in a single cycle whereas floating-point instructions execute in six cycles, thus the duration of the execution stage varies. A cache access (instruction and data) takes a single

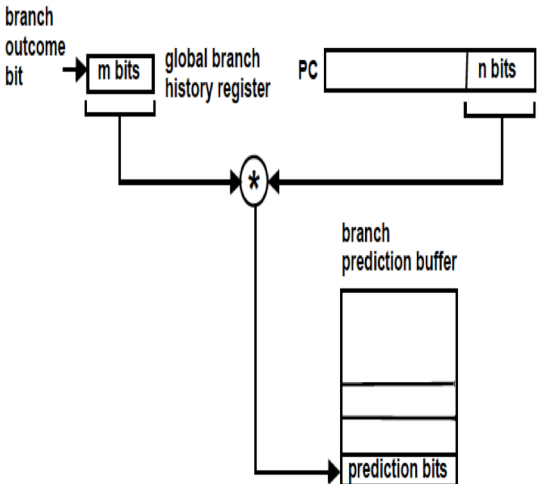
cycle whereas stores are decomposed into two instructions (one for address generation and one for cache access).



3A)
Describe in detail what happens in each stage when the floating-point ADD instruction – ADD.S F4, F2, F4 – is executed and the previous load instruction – L.S F2, 0(R1) is being executed and has not updated the register file (due to a cache miss) when the ADD instruction is dispatched. (**3 points)**

3B)
Explain in detail how a Reorder Buffer can enable speculative execution by first explaining what information is kept in each of the entries and what operations are carried out when an instruction finishes its execution (speculatively) and when it is eventually committed.
(3 points)

3C)
The figure below shows a two-level branch predictor. Let us use it to predict the outcome of one particular branch instruction using a single bit predictor and using m=4 bits of branch outcome. Initially, all prediction bits are cleared (not taken). If the 16 most recent executions of the branch are taken, what is the branch prediction accuracy? (**3 points)**



3D)
Explain how explicit register renaming is used using a physical register file and then keeping track of the latest and the retired value of a particular register using a front-end and a back-end *Register Alias Table* (RAT). **(3 points)**

ASSIGNMENT 4

4A)
A computer architect wants to establish how many misses in each of the three categories there are for a 4-block, two-way associative cache using LRU. A program does the following block references: 0 1 2 3 4 0 5 1 8 4 9 5.

Establish the number of cold, capacity (using the OPT replacement policy) and conflict misses. (**6 points)**

4B)
Consider the following program:

```
LOOP: LD.S F0, 0(R1)
      ADDI R1,R1,#8
      J LOOP
```

It is executed on a system with a lock-up free cache having 16 miss-status holding registers (MSHRs). The CPI for ADDI and J is 1 and 3, respectively, whereas the CPI of LD in case the cache is not blocked is 1. The miss penalty is 100 cycles and the cache block size is 64 bytes.

i) After how many cycles will the cache make the processor stall?
ii) What is the average CPI when executing the program assuming an infinite number of iterations?

(4 points)

4C)
Explain what next-block prefetching is and how it is used to improve the performance of the instruction cache. **(2 points)**

ASSIGNMENT 5

5A) Consider a multicore system comprising a number of processors (cores) on a chip that are connected to a single-level private cache. The private caches use the *write-back* write policy. X_i=R_i and X_i=W_i, means a read and a write request to the *same* address from processor *i,* respectively, where W_i=*C* means that the value C is written by processor *i*. Now consider the following access sequence:

$W_{1=}0$
R_1
R_2
$W_{1=}1$
R_2

What should be returned by the second read operation from processor 2 and what is the reason that the correct value is not returned given the cache write policy assumed?
(**3 points)**

5B)
Explain how we can change the cache controller of a write-back cache using appropriate states to implement the MSI cache coherence protocol. Give at least one example of a transition from one of the M-S-I states to any other. **(6 points)**

5C)
A computer architect is contemplating whether to use block (coarse-grained) multithreading or interleaved (fine-grained) multithreading. She knows that a miss is taken every tenth instruction and that it takes ten cycles to handle it. Moreover, all instructions are executed with CPI=1. How many threads are needed in each case to keep the pipeline busy all the time? **(3 points)**

****** GOOD LUCK! ******

Solutions for Exam 2013-12-16

ASSIGNMENT 1

1A) According to the assumptions, P1 executes twice as many instructions as P2. That is, if P2 executes N instructions, P1 executes 2N instructions. Moreover, since all machines have the same ISA, the number of instructions executed for one program is the same across the machines.

For P1 on A: T = 2N x CPI x Tc,
where CPI = 0.40x1 + 0.25x2 + 0.10x1 + 0.08x1 + 0.12x3 + 0.05x1 =
T = 2N x 1.49 x 1 = 2.98 N ns

For P2 on A: T = N x CPI x Tc,
where CPI = 0.20x1 + 0.35x2 + 0.20x1 + 0.08x1 + 0.12x3 + 0.05x1 =
T = N x 1.59 x 1 = 1.59 N ns

For P1 on B: T = 2N x CPI x Tc
where CPI = 0.40x2 + 0.25x3 + 0.10x1 + 0.08x1 + 0.12x1 + 0.05x1 =
T = 2N x 1.90 x 0.83 = 2.42 N ns

For P2 on B: T = N x CPI x Tc,
where CPI = 0.20x2 + 0.35x3 + 0.20x1 + 0.08x1 + 0.12x3 + 0.05x1 =
T = N x 1.55 x 0.83 = 1.29 N ns

For P1 on R: T = 2N x CPI x Tc
where CPI = 0.40x2 + 0.25x2 + 0.10x2 + 0.08x2 + 0.12x2 + 0.05x2 = 1
T = 2N x 2 x 1 ns = 4N ns

For P2 on R: T = N x CPI x Tc,
where CPI = 0.20x2 + 0.35x2 + 0.20x2 + 0.08x2 + 0.12x2 + 0.05x2 =
T = N x 1 x 1 = 2N ns

We can now determine the geometric means starting with Machine A.

P1: SP = TR/TA= 4/2.98 = 1.34
P2: SP = TR/TA = 2/1.59 = 1.26
Geom mean for Machine A= $(1.34x1.26)^{1/2}$=1.30

For Machine B
P1: SP = TR/TB = 4/2.42 = 1.65

P2: SP = TR/TB = 2/1.29 = 1.55
Geom mean for Machine B= $(1.65 \times 1.55)^{1/2}$=1.46

1B)
For Machine A CPI is 1 so execution time for P3 is 4N x 1 x 1 = 4N
For Machine B CPI is 2 so execution time for P3 is 4N x 2 x 0.83 = 6.6N
For Machine R CPI is 1 so execution time for P3 is 4N x 1 x 1 = 4N

We can now calculate geometric mean for Machine A as follows:
P1 and P2: Geometric mean is 1.30
P3: Speedup is 1, so geom mean = $(1 \times 1.30^2)^{1/3}$=1.19

Same for Machine B:
P1 and P2: Geom mean is 1.46
P3: Speedup is 6.6/4=1.65, so geom mean = $(1.65 \times 1.46^2)^{1/3}$= 1.52

1C) The maximum performance improvement is obtained if ALU instructions are done in zero time on program P1.

For P1 on A: T = 2N x CPI x Tc,
where CPI = 0.25x2 + 0.10x1 + 0.08x1 + 0.12x3 + 0.05x1 =
T = 2N x 1.09 x 1 = 2.18 N ns (as compared to 2.98N)

Etc.

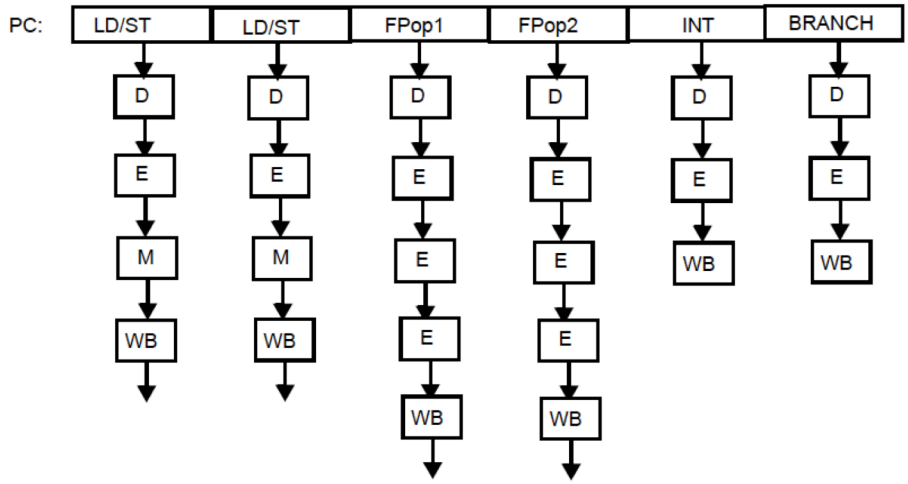
1D) We can host 10 As or 20 Bs on the same die given the area assumptions.

First, the speedup of the execution time of B over A on P1 is TA/TB = 2.98/2.42 = 1.23

The speedup of the throughput of B over A on P1 is 20 x 1.23/10 = 2.46

ASSIGNMENT 2

We consider in this assignment a VLIW architecture that can issue two memory, two floating-point, one integer, and one branch instruction each cycle according to the pipeline organization below. There are no forwarding units.



2A) How many cycles does it take to resolve a RAW dependency between instructions of different types, i.e., between a floating-point operation and a subsequent floating point operation, between a floating-point operation and a subsequent store operation, between a load operation and a subsequent floating point operation, and between a store operation and a subsequent load operation? (**3 points)**

RAW	**LD**	**FP ADD**	**SD**
LD	**--**	**2**	**--**
FP ADD	**--**	**4**	**4**
SD	**1**	**--**	**--**

2B)
Let's mark the first three instructions with O1 – O3:

```
LOOP: L.D    F0, 0(R1)   - O1
      ADD.D  F4,F0 F2    - O2
      S.D    F4,0(R1)    - O3
      SUBI   R1,R1, #8
      BNE    R1,R2, LOOP
```

The scheduling table will look like this. We note that the software pipeline is filled at INST 7 in which the Store from iteration 1 is executed together with the Add and the Load from iterations 6 and 8, respectively.

	ITE 1	ITE 2	ITE3	ITE4	ITE5	ITE6	ITE7	ITE8
INST 1	O1							
INST 2		O1						
INST 3	O2		O1					
INST 4		O2		O1				
INST 5			O2		O1			
INST 6				O2		O1		
INST 7	O3				O2		O1	
INST 8		O3				O2		O1
INST 9			O3				O2	
INST10				O3				O2
INST 11					O3			
INST 12						O3		
INST 13							O3	
INST 14								O3

2C) See textbook.

2D)
The predicated code is:

```
LW  R4, 0(R1)
ADDI R6,R4,#1
SUBI R7,R5,R4
LWNZ   R6,0(R2),R7
SW R6, 0(R1)
```

Where LWNZ is executed if and only if R7 is zero.

ASSIGNMENT 3

3A) See textbook.

3B) See textbook.

3C)

On the first four predictions, the four most significant bits in the index for the branch prediction buffer are the following:
0001, 0011, 0111, 1111. These will all cause mispredictions. All subsequent predinctions will be correct so the prediction accuracy is 12/16.

3D) See textbook.

ASSIGNMENT 4

4A)

Cold misses: This is essentially the number of unique blocks being accessed: 8

Capacity misses:

Access to block 4 will evict block 2 so the content is 0, 1, 3, 4
Access to block 0 will hit
Access to block 5 will evict block 3 as it is not accessed in the future so the content is 0, 1, 4, 5
Access to block 1 hits
Access to block 8 will evict 0 so content is 1, 4, 5, 8.
Access to block 4 hits
Access to block 9 evicts 1
Access to block 5 hits
No further misses so number of capacity misses is 0.

Conflict misses:
Access to block 4 replaces LRU which is o so conent is 2,4,1,3
Access to block 0 misses (conflict miss) and evicts 2 (same set) so content is 0,4,1,3
Access to block 5 misses (cold miss) and evicts 1 so content is 0,4,3,5
Access to block 1 misses (conflict miss) and evicts 3 so content is 0,4,1,5
Access to block 8 misses (cold miss) and evicts 4 so content is 0,8,1,5
Access to block 4 misses (conflict miss) and evicts 0 so content is 4,8,1,5
Access to block 9 misses (cold miss) and evicts 5 so content is 4,8,1,9
Access to block 5 misses (conflict miss) and evicts 1 so content is 4,8,1,5

Number of conflict misses: 4

4B)

The cache can have 16 outstanding cache misses (primary or secondary) so it will block on its 17^{th} iterations. One iteration takes 5 cycles so 16 iterations take 80 cycles. Since the miss penalty is 100 cycles the processor will have to stall for 20 cycles on the 17^{th} execution of the load instruction.

The average CPI is essentially the number of cycles it takes to execute 16 iterations (100 cycles) divided by the number of instructions in these 16 iterations (3x16=48). Thus, CPI = 100/48

4C) See textbook

ASSIGNMENT 5

5A) The second read operation from processor 2 will return the content in its cache which is zero whereas one should be returned which is the last written value.

5B) See textbook.

5C)

Under fine-grained multithreading, when a processor encounters a cache miss we will have to hide the latency of it by switching from thread to thread so we need ten.

Under coarse-grained multithreading we switch to another thread only when a miss is encountered and continue with that one until that encounters a cache miss. Ideally, we could do with two threads because when one encounters a cache miss, the next one can execute another ten instructions before a cache miss is encountered.