CHALMERS

2018-01-05

Exam in DAT 105 (DIT 051) Computer Architecture

Time: January 13, 2017 14 – 18 Lecture Halls at Civil Engineering Bldng

Person in charge of the exam: Per Stenström, Phone: 0730-346 340

Supporting material/tools: Chalmers approved calculator.

Exam Review: On January 26, 2017 13-15 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 for approved exams used solely for higher grades.

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 which executes 1 million instructions. The operating frequencies of the three machines (A, B and R) are also shown.

Program P1	**A (Frequency in percent/CPI)**	**B**	**R**
Integer/branches	40/2	40/1	40/3
Loads	25/4	25/1	25/3
Stores	10/2	10/1	10/3
FP Multiply	5/100	5/200	5/800
Misc.	20/2	20/1	20/3

Program P2	**A (Frequency in percent/CPI)**	**B**	**R**
Integer/branches	20/2	20/1	20/3
Loads	35/4	35/1	35/3
Stores	20/2	20/1	20/3
FP Multiply	1/100	1/200	1/800
Misc.	24/2	24/1	24/3

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

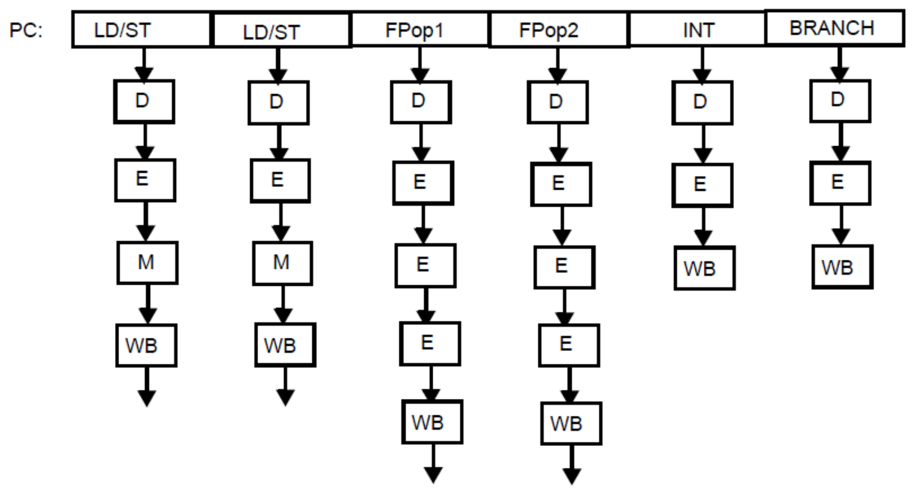
1A) Calculate the execution time for P1 and P2 on A, B and R (**4 points)**

1B) Determine which of the machines is the fastest using both **arithmetic means** and **geometric means** and explain which machine is the fastest based on each of the means **(4 points)**

1C) What would be the speedup if FP MULT could be implemented with a CPI of 1? Which machine is now the fastest?
(2 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) Consider the following code:

```
LOOP: LD F1,0(R10)
      LD F2,0(R11)
      ADD F3, F1,F2
      SD F3,0(R12)
      SUBI R1,R1,#1
      ADDI R10, R10, #8
      ADDI R11, R11, #8
      ADDI R12, R12, #8
      BNE R1, R2, LOOP
```

How many cycles does it take to execute one iteration of the code on the VLIW pipeline?
(2 points)

2B) Consider again the code in Assignment 2A. Assume that the program is run for an infinite number of iterations. Show a static schedule of the code in which instructions inside an iteration are reordered to minimize the number of cycles lost due to RAW hazards.
(3 points)

2C) Consider again the code in Assignment 2A. Now use software pipelining to minimize the number of cycles to execute a single iteration. Show the prologue, epilogue and the kernel for the software pipelined code. **(3 points)**

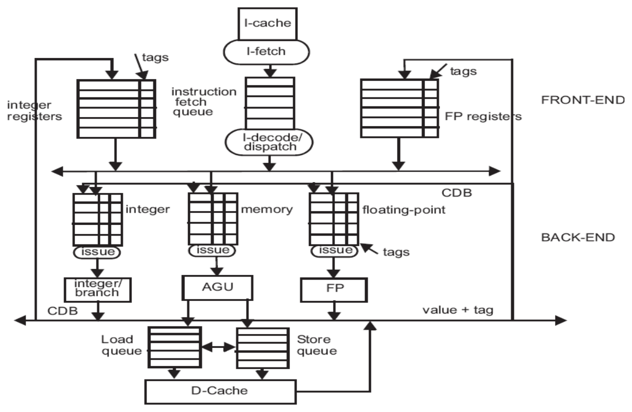
2D) The following code contains four basic blocks: A, B, C and D. Either the trace A, B, C is executed or A, D, C depending on whether R5=R4 (second trace) or not (first trace).

Assume that the first trace is much more likely than the second one. Rewrite the code to minimize the number of instructions in the first trace using software speculation. The program must produce correct results regardless of what trace is executed. (**4 points)**

```
LW R4,0(R1)
ADDI R6,R4,#1
/* block A */
BEQ R5,R4,LAB
LW R6,0(R2)
/* block B */
/* block D */
LAB: SW R6,0(R1)
/* block C */
```

ASSIGNMENT 3

The diagram below shows a pipeline with support for Tomasulo's algorithm. There are two functional units for adding floating-point numbers and a single functional unit for floating-point division. It takes 2 cycles to carry out an addition/subtraction and 5 cycles to carry out a division.



3A) Explain *in detail* what happens in each of the three pipeline stages: Issue, Execute, and Write result. In particular explain how data hazards are resolved and in which cycle each instruction in the sequence below enters the different stages by filling out a pipeline diagram similar to the one below for the following instruction sequence. (**6 points)**

```
ADDF   F1, F2, F3
DIVD    F4, F1, F2
SUBD   F2, F5, F6
```

	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6
ADD	Issue					
DIVD		Issue				
SUBD			Issue			

3B) We want to add support for speculative execution. Explain how a reorder buffer and another pipeline stage called COMMIT can allow the following instruction sequence to execute speculatively and what is done in each of the four stages.

```
BNEZ R1, LABEL
ADDF   F1, F2, F3
DIVD    F4, F1, F2
SUBD   F2, F5, F6
```

Assume that the branch instruction is predicted to NOT be taken and that the prediction is correct. **(3 points)**

3C) Explain what problem a branch target buffer solves and how it is organized
(3 points)

ASSIGNMENT 4

4A) A computer architect wants to establish the relative performance between a system with a blocking and a non-blocking cache. The cache-hit time is 1 and the miss penalty is 100 cycles for both caches. Each data element occupies 4 bytes and the block size is 16 bytes. Consider the following code:

```
for (i=0; i<N; i++)
   C+=A[i];
```

Assuming that there are 4 instructions between each load instruction, the CPI is 1 for all instructions except load instructions that miss in the cache. How many MSHRs (Miss Status Holding Registers) are needed to avoid having the cache to block on a load access?

How much faster does the code runs on the non-blocking cache with sufficient number of MSHRs in comparison with on a blocking cache? (**6 points)**

4B) In the table below, cache miss rates for a number of organizations are shown. Determine from the data the cold miss rate, the capacity miss rate and the conflict miss rate for a 2-way 16-KB cache. (**3 points)**

Cache organization	**Miss rate**
16-KB direct mapped cache	10%
16-KB 2-way assoc. cache	8%
16-KB fully associative cache	6%
Cache with infinite size	2%

4C) Consider the following code:

```
LOOP: LD F0, 0(R10)
      ADD F1,F0,F0
      SD F1, 0(R10)
      ADDI R10,R10,#8
      SUBI R1,R1,#1
      BNEZ R1, LOOP
```

A computer system supports software prefetching through an instruction PF D(Rx) that prefetches the data at address D + (Rx). Augment the code with software prefetching instructions. Assume that the memory latency corresponds to two iterations. **(3 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, mean 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:

R_1
W_1=0
R_2
W_1=1
R_2

What is 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? How can we modify the cache controller to make sure that the right value is returned?
(**6 points)**

5B)
Assume a write-back cache and the MSI-protocol. What bus transaction will cause a state transition from state S to state M and what transaction will cause a transition from state M to state S? **(2 points)**

5C) Explain the concept of block (coarse-grain) multithreading. Consider a five-stage pipeline. What additional mechanisms must be added to support block multithreading? How many cycles are lost on a thread switch? **(4 points)**

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

Solutions

ASSIGNMENT 1

1A)

T_{Exec} = IC x CPI x Tc

Program P1:

A: T_{Exec} = 2×10^6 x (0.40x2 + 0.25x4 + 0.10x2 + 0.05x100 + 0.20x2) x 1 x 10^{-9} s = 14.8 ms
B: T_{Exec}=2×10^6 x (0.40x1 + 0.25x1 + 0.10x1 + 0.05x200 + 0.20x1)x 1 x 10^{-9}/1.2 s = 9.13 ms
R: T_{Exec}=2×10^6 x (0.40x3 + 0.25x3 + 0.10x3 + 0.05x800 + 0.20x3)x 1 x 10^{-9} s = 42.9 ms

Program P2:

A: T_{Exec} = 1×10^6 x (0.20x2 + 0.35x4 + 0.20x2 + 0.01x100 + 0.24x2) x 1 x 10^{-9} s = 3.68 ms
B: T_{Exec}=1×10^6 x (0.20x1 + 0.35x1 + 0.20x1 + 0.01x200 + 0.24x1)x 1 x 10^{-9}/1.2 s = 2.99 ms
R: T_{Exec}=1×10^6 x (0.20x3 + 0.35x3 + 0.20x3 + 0.01x800 + 0.24x3)x 1 x 10^{-9} s = 10.97 ms

1B)

Arithmetic mean:

A: Arith. Mean = (14.8 + 3.68)/2 ms = 9.24 ms
B: Arith. Mean = (9.13 + 2.99)/2 ms = 6.06 ms

B is the fastest.

Geometric mean:

Use R as a reference machine

Geom mean = SQRT(S_1 x S_2)
A: Geom mean = SQRT(42.9 x 42.9/(14.8 x 3.68)) = 7.55
B: Geom mean = SQRT(42.9 x 42.9/9.13 x 2.99)) = 8.21

B is the fastest (higher is better). However, the difference is smaller because geometric means tend to smooth out the difference.

1C)

Program P1:

A: $T_{Exec} = 2x10^6 x (0.40x2 + 0.25x4 + 0.10x2 + \underline{0.05x1} + 0.20x2) x 1 x 10^{-9} s = 3.50$ ms
B: $T_{Exec}=2x10^6 x (0.40x1 + 0.25x1 + 0.10x1 + \underline{0.05x1} + 0.20x1)x 1 x 10^{-9}/1.2 s = 1.66$ ms
R: $T_{Exec}=2x10^6 x (0.40x3 + 0.25x3 + 0.10x3 + \underline{0.05x1} + 0.20x3)x 1 x 10^{-9} s = 5.8$ ms

Program P2:

A: $T_{Exec} = 1x10^6 x (0.20x2 + 0.35x4 + 0.20x2 + \underline{0.01x1} + 0.24x2) x 1 x 10^{-9} s = 2.69$ ms
B: $T_{Exec}=1x10^6 x (0.20x1 + 0.35x1 + 0.20x1 + \underline{0.01x1} + 0.24x1)x 1 x 10^{-9}/1.2 s = 0.83$ ms
R: $T_{Exec}=1x10^6 x (0.20x3 + 0.35x3 + 0.20x3 + \underline{0.01x1} + 0.24x3)x 1 x 10^{-9} s = 2.98$ ms

B is still the fastest

ASSIGNMENT 2

2A)

LD/ST1	LD/ST2	FPOP1	FPOP2	INT	BRA
LD F1,0(R10)	LD F2,0(R11)			SUBI R1…	
				ADDI R10..	
				ADDI R11	
		ADD F3…			
				ADDI R12…	BNE R1
	SD F3,0(R12)				

It takes eight cycles. Note how we schedule the ADDI instructions in such a way that they write back to the register file to avoid WAR hazards with respect to the memory instructions. Note also that the branch will be executed in the same cycle as the store instruction.

2B)

If the loop is unrolled a sufficient number of times (here three times), one can fill all the slots before the first ADD with Load instructions and respect the dependences:

LD/ST1	LD/ST2	FPOP1	FPOP2	INT	BRA
LD F1,0(R10)	LD F2,0(R11)			SUBI R1…	
LD F4,8(R10)	LD F5,8(R11)				
LD	LD			ADDI R10..	

F7,16(R10)	F8,16(R11)				
		ADD F3…		ADDI R11	
		ADD F6…			
		ADD F9…			
SD F3,0(R12)					BNE R1
SD F6,8(R12)				ADDI R12…	
SD F9,16(R12)					

2C) software pipelining of the loop in 2A)

Cycle	ITE1	ITE2	ITE3	ITE4	ITE5	ITE6	ITE7	ITE8	ITE9
1	LD/LD								
2		LD/LD							
3			LD/LD						
4				LD/LD					
5	ADD				LD/LD				
6		ADD				LD/LD			
7			ADD				LD/LD		
8				ADD					
9	SD				ADD			LD/LD	
10		SD				ADD			LD/LD
11			SD				ADD		
12				SD				ADD	
13					SD				ADD

Prologue: Rows 1 – 8
Kernel: Row 9
Epilogue: Rows 11-13 (and thereafter until all instructions have left the pipeline which happens five cycles later.

2D) See textbook and Fig 3.34 d) on page 152.

ASSIGNMENT 3

3A)

Issue: Instruction is sent to the corresponding issue queue of the functional unit to be used. If no free entry, it waits in the issue stage. It will wait for its operands there and when available it is issued for execution.
Execute: The instruction is executed and (2 cycles for add/sub and five cycles for division)
Write result (CDB): When the instruction is done the result will be broadcast to all issue queues and to the register file on the common data bus.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
ADD	Issue	Ex	Ex	CDB								
DIVD		Issue	Issue	Issue	Ex	Ex	Ex	Ex	Ex	CDB		
SUBD			Issue	Ex	Ex	CDB						

This table shows when instructions pass each stage. Note that due to the RAW hazard between the ADD and the DIVD, DIVD cannot execute until C5. SUBD has a WAR hazard with respect to DIVD that is resolved by the tag so it can start executing right after the ADD has left the execution unit in C4.

3B)

Branch prediction is done in the instruction fetch stage. Assuming that branch not taken is predicted the subsequent instructions will be speculatively executed until the branch instruction is executed four cycles later. Each speculatively executed instruction is inserted in the reorder buffer in the order it appears in the program. When the branch instruction has been executed and assuming that the prediction is correct, it will go to the COMMIT stage and at that point it will leave the reorder buffer. All subsequent speculatively executed instructions up until the next branch will also commit successfully and will be removed from the reorder buffer, one by one.

3C) Explain what problem a branch target buffer solves and how it is organized
For certain branch instructions, for example J (R12) which branches to the address contained in register 12, the branch target is needed already in the instruction fetch stage. Here, a branch target buffer uses the program counter to index into a table in which branch targets are stored. The branch target buffer is organized as a cache where the program counter is used as an address. Portions of it are used as an index and the rest is compared with a tag associated with each entry in the branch target buffer.

ASSIGNMENT 4

4A)

With a blocking cache, the load instruction in every fourth iteration will miss because a block contains four vector elements and all of them are accessed in consecutive iterations. Since CPI=1 and there are four instructions in each iteration, the first iteration takes 103 cycles and the next 3 take 4 x 3 cycles so four iterations take in total 115 cycles.

With a non-blocking cache an MSHR is occupied for as long as it takes to service the miss, i.e., 100 cycles. In that time, 25 iterations are executed, and each iteration needs an MSHR so 25 MSHRs are needed. With a non-blocking cache, four iterations are executed in 16 cycles. The speedup is 115/16 = 7 times.

4B)

Cold miss rate is the miss rate for an infinite cache: 2%
Capacity miss rate is the miss rate for a fully associative cache minus the cold miss rate: (6 – 2)% = 4%
Conflict miss rate is the miss rate for the cache minus the capacity and the cold miss rates: (8 – 2 – 4)% = 2%

4C)

```
LOOP: PF   16(R10)
      LD F0, 0(R10)
      ADD F1,F0,F0
      SD F1, 0(R10)
      ADDI R10,R10,#8
      SUBI R1,R1,#1
      BNEZ R1, LOOP
```

We have added PF 16(R10) which corresponds to the address the load instruction will use two iterations ahead.

ASSIGNMENT 5

5A)

The second read operation will return the value contained at the memory address initially which may not be the same as that of the second write by processor 1. The reason is that the modifications by processor 1 happens locally in processor 1's cache. By forcing all writes to generate a memory write and let all caches inspect the address of these memory writes a local copy containing the written location can be invalidates forcing a miss to happen which will load data from memory which is now up to date.

5B) If a block is in the S(hared) state it means that memory is up to date. What brings that block into the M(odified) state is a write to the block. Then an Upgrade message will be sent to memory and to all other caches forcing them to invalidate the block. On a subsequent read miss from some other cache, a Bus read request is posted on the bus. This will be intercepted by the cache having the block in the M(odified) state which will respond with the block in what is called a Flush operation. The new state is then S(hared).

5C) Explain the concept of block (coarse-grain) multithreading. Consider a five-stage pipeline. What additional mechanisms must be added to support block multithreading? How many cycles are lost on a thread switch? **(4 points)**

In block or coarse-grain multithreading, a switch to another thread happens when encountering a long-latency operation such as a cache miss. Since that is detected in the memory stage in a five stage pipeline, all instructions in the stages preceding that stage must be flushed. A thread switch stage is added between the instruction fetch stage