CHALMERS

2020-08-14

Exam in DAT 105 (DIT 051) Computer Architecture

Time: August 18, 14-18 (Canvas)

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

Supporting material/tools: Chalmers approved calculator, textbook.

Exam Review: More information on this will be available via Canvas

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

NOTE 3: Read the document Instructions for Canvas exam, available at Canvas, carefully

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

You are supposed to characterize the performance of two computers, A and B, with data regarding three programs P1, P2 and P3.

The following data has been collected.

Assumptions for computer model A	
Execution time for programs P1, P2 and P3:	1s, 10s and 2s, respectively
Number of instruction cache accesses:	10^8, 10^9 and 2 x 10^8 for P1, P2 and P3, resp.
Clocks per instruction assuming an instruction/data caches have zero misses	1.5 for P1-P3
Fraction of memory instructions	20%, 30%, 20% for P1, P2 and P3, resp.
Miss penalty	100 ns
Speedup over reference machine R	10, 10, 10 for P1, P2 and P3, resp.
A's operating frequency	1 GHz
Assumptions for computer model B	
Number of instruction cache accesses:	10^8, 10^8 and 10^8 for P1, P2 and P3, resp.
Clocks per instruction assuming an instruction/data caches have zero misses	1 for P1-P3
Fraction of memory instructions	30%, 20%, 30% for P1, P2 and P3, resp.
Miss penalty	100 ns
Speedup over reference machine R	5, 30, 5 for P1, P2 and P3, resp.
B's operating frequency	1 GHz

1A) What is the execution time of P1, P2 and P3 on B? **(3 points)**

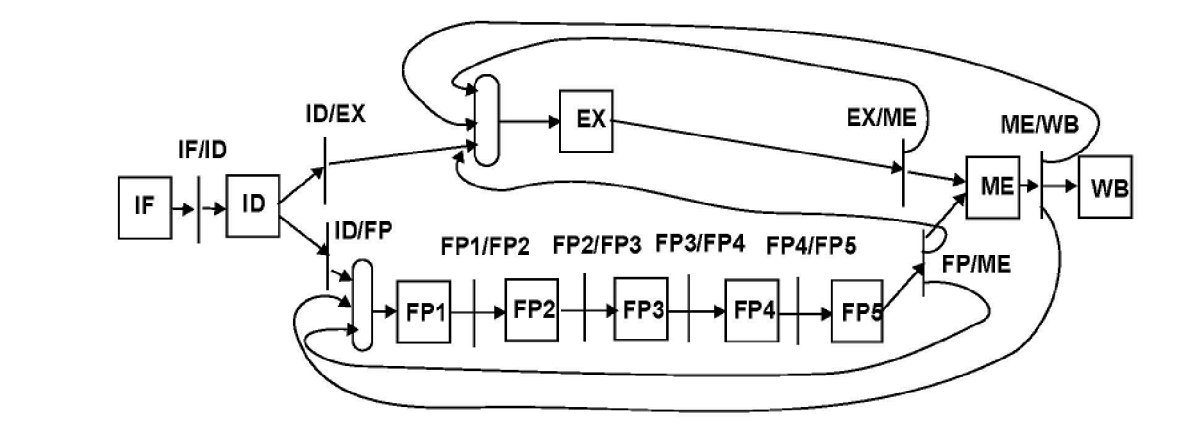
1B) What is the geometric mean execution time on A and B? Which one is the fastest and by how much? **(3 points)**

1C) Calculate the number of Misses Per Kilo Instructions (MPKI) for P1 on A and B. **(3 points)**

1D) Assume that 50% of the misses are caused by instruction fetches. What is average number of misses per memory instruction for P1 on A and B? **(3 points)**

ASSIGNMENT 2

We consider in this assignment a pipeline with a 5-stage pipelined floating-point unit and a single-stage execution unit that executes integer, load/store and branch instructions. There are forwarding units from the output of each execution unit and from the memory stage.



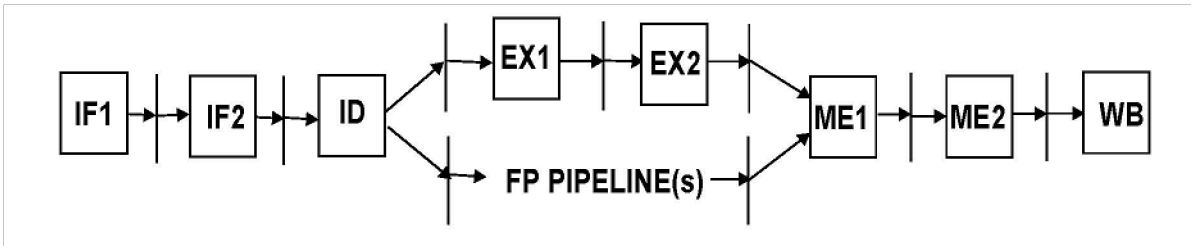
2A) Consider the following instruction sequence:

I1: ADD F0,F1,F2
I2: ADD R1,R2,R3
I3: ADD R4,R1,R3
I4: ADD F3,F4,F5
I5: ADD F6,F7,F5

Determine the number of cycles it takes from I1 is issued from the ID-stage until I5 enters the ME-stage in the case that i) the floating-point unit is fully pipelined and ii) when it is not. (**4 points)**

2B) The model in 2A) is now changed to cope with higher clock frequencies as follows. - The IF stage is replaced by two IF stages: IF1 and IF2

- The integer execution unit is replaced by two **fully pipelined** stages: EX1 and EX2
- The FP pipeline now consists of five stages where only a single FP operation can be initiated every five cycles
- The ME stage is replaced by two **fully pipelined** stages



Consider the same program as in 2A) and determine the number of cycles it takes from I1 is issued from the ID-stage until I5 enters the ME1-stage. Assuming that we can double the frequency for this pipeline, which of the pipelines (the two in 2A and the one here) is the fastest? **(4 points)**

2C) Consider the following loop and apply loop unrolling twice. **(4 points)**

LOOP: LD F0, 0(R1)
LD F1, 0(R2)

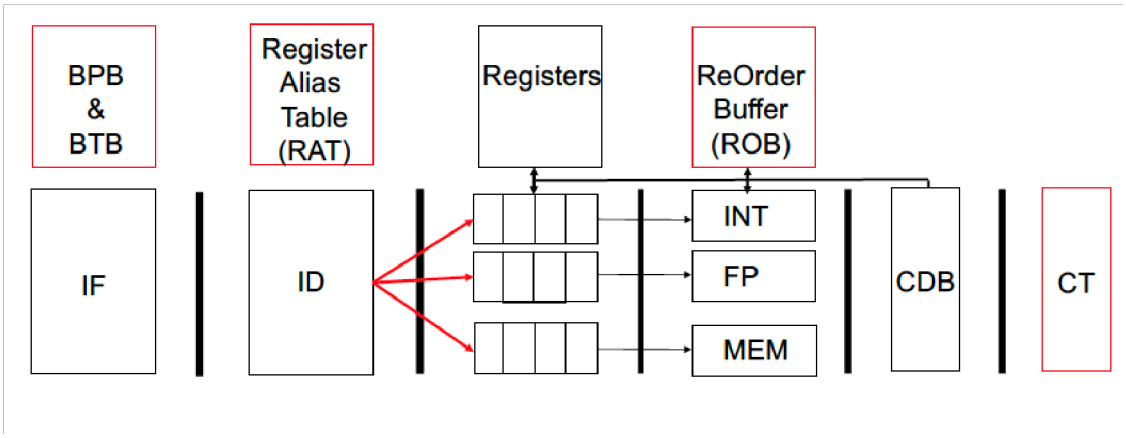
```
ADD F4, F0, F1
SD F4, 0(R1)
ADDI R1, R1,#8
ADDI R2, R2,#8
SUBI R3, R3,#1
BNEZ R3, LOOP
```

ASSIGNMENT 3

The diagram below shows a pipeline with support for speculative execution. There are three functional units: one for integer/branch instructions (INT); one for floating-point instructions (FP) and one for memory instructions (MEM). There is a branch prediction mechanism (BPB) using 1 bit for prediction, meaning that the prediction is changed every time there is a misprediction. In addition, a branch target buffer (BTB) is in the IF stage.

For the functional units, it takes a single cycle to execute an INT instruction, three cycles for an FP instruction, a single cycle for a load and two cycles for a store in the MEM unit.

The pipeline supports register renaming using a register alias table (RAT) and data hazards are handled using the Tomasulo algorithm. Speculation is enabled by a reorder buffer and speculatively executed instructions are committed in the commit stage (CT).



Consider the following program:

```
I1: LOOP: L.S    F0, 0(R1)
I2:       L.S    F1, 0(R2)
I3:       ADD.S  F2, F1, F0
I4:       ADD.S  F1, F3, F4
I5:       SD     F4, 0(R1)
I6:       ADDI   R1, R1,#8
I7:       ADDI   R1, R2,#8
I8:       SUBI   R3, R3,#1
I9:       BNEZ   R3, LOOP
```

3A)

i) Explain in detail how the RAT is instrumental in detecting the WAR hazard between I3 and I4. **(2 points)**

ii) Explain in detail how the ROB and the RAT are instrumental in detecting the RAW hazard between I1/I2 on one hand and I3 on the other. **(2 points)**

3B) Show with a pipeline timing diagram, cycle-by-cycle, when each of the instructions I1..I9 enters a specific pipeline stage assuming that I1 has entered ID at cycle 0 and when I9 reaches the CDB stage. **(3 points)**

3C) Explain in detail what happens in the Commit (CT) stage when a branch that is mispredicted is committed (**2 points)**

3D) Assume that the branch predictor is initialized to Not-Taken and that the loop in 3A) is executed 100 times. What is the fraction of correct branch predictions in percent? (**3 points)**

ASSIGNMENT 4

4A) Explain which of the statements, below, are **not true** and why they are **not true.**
In a two-level **inclusive** memory-hierarchy the following holds:

i) A block in the upper level always exists in the lower level
ii) A block in the upper level do not exist in the lower level
iii) A block in the lower level may also exist in the upper level
iv) A block in the lower level does not exist in the upper level

Note: A wrong answer cancels a correct answer. **(2 points)**

4B) A computer architect wants to establish the relative performance between a system with a blocking and a non-blocking cache using software prefetching. In software prefetching, prefetch instructions are appropriately inserted in the code shown below. Assume that five instructions are executed per iteration where the CPI for each instruction is one assuming that it doesn't cause a cache miss. On the other hand, if the instruction causes a cache miss, CPI is 20. In addition, prefetch instructions result in CPI=2. Both caches (blocking and non-blocking) have a block size of four words. In the non-blocking cache, there are 8 miss-status-holding registers.

```
for (i=0; i<1000; i++)
     C+=A[i];
```

First show how the code is annotated with prefetch instructions so that the cache miss penalty can be eliminated. How much faster does the program run on the system with a non-blocking cache using software prefetching? A convincing explanation that is easy to follow is needed for full points. (**6 points)**

4C) Consider a two-way set-associative cache with 8 blocks and the following sequence of block accesses:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 17, 18, 1, 2

Determine the number of cold, capacity and conflict misses in the cache. Assume LRU for determination of capacity and conflict misses (**4 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-through write policy. $X_i=R_i$ and $X_i=W_i$, mean a read and a write request to the *same* address X from processor *i,* respectively, where $W_i=C$ means that the value C is written by processor *i*. Now consider the following access sequence assuming that X is not present in any cache from the beginning and that X originally contains the value 0:

$W_{1=}1$
R_1
R_2
$W_{1=}2$
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) How can a simple change of the write-through cache make sure that the correct value is returned to processor 2? **(3 points)**

5C) Multithreading refers to a general technique to switch to another thread when a high latency operation is encountered. Explain for the three example techniques a) block multithreading b) interleaved (or fine-grained) multithreading and c) simultaneous multithreading how a five-stage pipeline must be extended to support each of the techniques. (**6 points).**

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

Solutions to the exam in DAT105/DIT 051 2020-08-18

ASSIGNMENT 1

1A) Half of the number of misses stems from memory instructions
Denote SP_A = Execution time R/ Execution time A and SP_B = Execution time R/ Execution time B. Then

Execution time B = Execution time R/SP_B = Execution time A x SP_A/ SP_B
Execution time B (P1) = 1 x 10/5 s = 2 s

Execution time B (P2) = 10 x 10/30 s = 3.3 s
Execution time B (P3) = 2 x 10/5 s = 4 s

1B) GM = (T(P1) x T(P2) x T(P3)) $^{1/3}$

A: $GM_A=(1x10x2)^{1/3}$ = 2.7
B: $GM_B=(2x3.3x4)^{1/3}$ = 2.9

Hence, A is 7% faster than B.

1C) The number of instructions executed is the same as the number of instruction cache accesses. Since we know the number of clocks per instructions, assuming 100% hit rate, and the clock frequency, we can calculate how long time it would take to execute the program *without* any cache misses:

A: Ideal execution time for P1 is 10^8 x 1.5 x 1 ns = 0.15 s. Hence 0.85 s (out of 1s) is used to service cache misses. Since 1 cache miss takes 100 ns, there must be $0.85/10^{-7}$ misses = 0.85 x 10^7. That is, the total number of misses per instruction is 0.85 x $10^7/10^8$ = 0.085 misses and 85 MPKI.

B: Ideal execution time for P1 is 10^8 x 1 x 1 ns = 0.1 s. Hence 2 – 0.1s =1.9 s is used to service cache misses. Since 1 cache miss takes 100 ns, there must be $1.9/10^{-7}$ misses = 1.9 x 10^7. That is, the total number of misses per instruction is 1.9 x $10^7/10^8$ = 0.19 misses and 190 MPKI.

1D)
A: From 1C) we know that each instruction generates 0.085/2 instruction misses including memory instructions. On the other hand, there are 20% memory instructions for P1, that is, 2 x 10^7. The average number of misses for a memory instruction is $0.85x10^7/(2x2x10^7)$ = 0.21 misses. Hence, the total number of misses per memory instruction is 0.0425 + 0.21 = 0.25.

B: From 1C) we know that each instruction generates 0.19/2 = 0.095 instruction misses including memory instructions.
On the other hand, there are 30% memory instructions for P1, that is, 3 x 10^7. The average number of misses for a memory instruction is $0.095x10^7/(3x10^7)$ = 0.031 misses. Hence, the total number of misses per memory instruction is 0.095+ 0.031 = 0.12.

ASSIGNMENT 2

2A)

i) Fully pipelined functional unit: 10 cycles

	ID	FP1	FP2	FP3	FP4	FP5	EX	ME
C1	I1							
C2	I2	I1						
C3	I3		I1				I2	
C4	I4			I1			I3	I2
C5	I5	I4			I1			I3
C6		I5	I4			I1		
C7			I5	I4				
C8				I5	I4			I1

C9					I5	I4		
C10						I5		I4
C11								I5

ii) Not pipelined functional unit: 16 cycles

	ID	**FP1**	**FP2**	**FP3**	**FP4**	**FP5**	**EX**	**ME**
C1	I1							
C2	I2	I1						
C3	I3		I1				I2	
C4	I4			I1			I3	I2
C5	I4				I1			I3
C6	I4					I1		
C7	I5	I4						
C8	I5		I4					I1
C9	I5			I4				
C10	I5				I4			I4
C11	I5					I4		
C12		I5						
C13			I5					
C14				I5				
C15					I5			
C16						I5		
C17								I5

2B)

	ID	**FP1**	**FP2**	**FP3**	**FP4**	**FP5**	**EX1**	**EX2**	**ME1**
C1	I1								
C2	I2	I1							
C3	I3		I1				I2		
C4	I3			I1				I2	
C5	I4				I1		I3		I2

C6	I5	I4				I1		I3	
C7		I5	I4					I3	I1
C8			I5	I4					I3
C9				I5	I4				
C10					I5	I4			
C11						I5			I4
C12									I5
C13									
C14									
C15									
C16									
C17									

The program will need 10, 16 and 11 cycles for the two pipelines in 2A i/ and ii/ and the one here, respectively. Since this pipeline is clocked at twice the frequency of the ones in 2A, this pipeline is the fastest: 10/5.5 = 1.81 and 16/5.5 = 2.9, that is by 81% and 190%, resp.

2C)

```
LOOP:  LD F0, 0(R1)
       LD F1, 0(R2)
       LD F2, 8(R1)
       LD F3, 8(R2)
       ADD F4, F0, F1
       ADD F5, F2, F3
      ADDI R1, R1,#16
      ADDI R2, R2,#16
      SUBI R3, R3,#2
      SD F4, -16(R1)
      SD F5, -8(R1)
      BNEZ R3, LOOP
```

ASSIGNMENT 3

3A)

i) When I3 accesses the RAT, F1 will be associated with the tag of F1 from I2 so it cannot erroneously access the result from I4. When I4 is accessing the RAT, register F1 will be

associated with the tag of I4 so there is a complete disassociation between F1 accessed in I3 and F1 written to in I4 eliminating the WAR hazard between I3 and I4.

ii) When I1 and I2 are issued, entries will be allocated in the ROB at say entry i1 and i2. These entries will be used as tags for the destination operands F0 and F1, respectively and are recorded in the RAT as the destination operands that will eventually be written back to the register file when I1 and I2 are committed. When I3 is issued, the RAT will provide the tags telling I3 to wait for the values produced by I1 and I2 hence eliminating the RAW hazard.

3B)

	ID	**IS**	**INT**	**FP1**	**FP2**	**FP3**	**MEM1**	**MEM2**	**CDB**
C1	I1								
C2	I2	I1							
C3	I3	I2					I1		
C3	I4	I3					I2		I1
C5	I5	I4		I3					I2
C6	I6	I5			I3				I3
C7	I7	I6				I3	I5		
C8	I8	I7	I6	I4				I5	I3
C9	I9	I8	I7		I4				I5
C10		I9	I8			I4			I6
C11			I9						I7
C12									I4
C13									I8
C14									I9

3C)

The ROB records all instructions in the order they appear in the program, the program order. When a branch is mispredicted and reaches the Commit stage (CT), all instructions that have been executed speculatively or are under execution appear in the ROB, after the branch. These instructions are flushed from the ROB and the branch is re-executed.

3D) Since the predictor is initialized to Not-Taken, the first branch (that is taken) is mispredicted and the last branch (that is not taken) is also mispredicted. The rest of the branches (98) are correctly predicted so the fraction is 2/100 = 2%.

ASSIGNMENT 4

4A)

i) **True** because the upper level is included in the lower level

ii) **False** as it contradicts i)

iii) **True** because a block can exist in the lower level that does not exist in the upper level.

iv) **False** because all blocks in the upper level exist in the lower level.

4B)

Each iteration takes 5 cycles. There is a cache miss every fourth iteration as there are four words in each block. Since four iterations take 20 cycles to execute, launching a prefetch every fourth iteration for the address to be used four iterations ahead will eliminate the miss penalty:

```
if (i mod 4) == 0 then
prefetch(i+4);
```

Execution time on a blocking cache:
Time for four iterations: 4 x 5 + 20 = 40 cycles

Execution time on a non-blocking cache:
Time for four iterations 4 x 5 + 2 = 22 cycles (2 for the prefetch instruction disregarding the conditional statement)

Speedup = 40/22 = 1.81. That is, 81% faster.

4C)

Cold misses: 12 (number of unique blocks) Capacity misses:

```
1 2 3 4 5 6 7 8 9 10 1 2 3 4 17 18 1 2
M M M M M M M M M  M M H H M  M  H H
```

Total: 14
Capacity=Total – Cold = 2

Conflict

```
1 2 3 4 5 6 7 8 9 10 1 2 3 4 17 18 1 2
M M M M M M M M M  M M H H M  M  M M
```

Total: 16
Conflict= Total – Capacity = 2

ASSIGNMENT 5

5A) The second read from processor 2 should return 2 reflecting the second write by processor 2 but will return the value 0 as the processor will read from its own cache.

5B) By simply invalidating the block in another cache when it sees a write to that block. This will force the second read to experience a cache miss that will return the correct value.

5C)

Regardless of the particular technique employed any multithreaded architecture must support as many register files as the number of threads including multiple program counters.

Block multithreading:

- A thread switch stage is added to switch to another thread on a long-latency operation.
- On a long latency operation, pipeline is flushed before switching to the next thread. - Thread ID accompanies the instructions for hazard detection etc.

Interleaved multithreading:

- A thread switch stage is added to switch to another thread on each cycle. - Thread ID accompanies the instructions for hazard detection etc.

Simultaneous multithreading:

- A thread scheduler selects from which thread(s) to fetch
- Thread ID accompanies the instructions for hazard detection etc.