

15740-18740 Computer Architecture, Fall 2010
Midterm Exam I

Instructor: Onur Mutlu

Teaching Assistants: Evangelos Vlachos, Lavanya Subramanian, Vivek Seshadri

Date: October 11, 2010

Name:

Problem 1 (50 points)	:	<input type="text"/>
Problem 2 (15 points)	:	<input type="text"/>
Problem 3 (12 points)	:	<input type="text"/>
Problem 4 (25 points)	:	<input type="text"/>
Problem 5 (15 points)	:	<input type="text"/>
Problem 6 (20 points)	:	<input type="text"/>
Problem 7 (25 points)	:	<input type="text"/>
Bonus Problem 8 (10 points)	:	<input type="text"/>
Legibility and Name (3 points)	:	<input type="text"/>
Total (165 + 10 points)	:	<input type="text"/>

Instructions:

1. This is a closed book exam. You are allowed to have one letter-sized cheat sheet.
2. No electronic devices may be used.
3. This exam lasts 1 hour 50 minutes.
4. Clearly indicate your final answer.
5. Please show your work when needed.
6. Please write your name on every sheet.
7. Please make sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Problem 1 (50 points)

1) Pipelining (5 points)

Give two reasons why IPC could decrease when we increase the pipeline depth of a processor, if this is actually possible. Explain very briefly.

Give two reasons why IPC could increase when we increase the pipeline depth of a processor, if this is actually possible. Explain very briefly.

2) Interrupts & Exceptions (6 points)

Assume that a floating point instruction generates a divide by zero exception during runahead mode in a runahead execution processor. When is this exception handled? Explain briefly.

Assume that a load instruction addresses a memory location that is not present in physical memory in runahead mode. When is this exception handled? Explain briefly.

Assume that the keyboard generates an interrupt request to the processor during runahead mode. When is this interrupt handled? Explain briefly.

3) Instruction & Data Caches (4 points)

An engineer designs a deeply pipelined, 30-stage machine. The goal is to achieve very high clock frequency, while having reasonable IPC. She is considering two design options for the first-level caches:

1. A unified instruction+data cache that has a total of 64KB data storage.
2. A separate 32KB instruction cache and 32KB data cache.

Describe which option you would recommend and why, in less than 30 words.

4) TLB Management (6 points)

Some ISAs (e.g., MIPS) specify a TLB miss to be handled by the operating system instead of hardware. This is called a software managed TLB, as opposed to a hardware-managed one, in which the hardware performs the TLB miss handling.

- a) Describe one advantage of a software managed TLB over a hardware managed one (less than 20 words please).

- b) Describe one advantage of a hardware managed TLB over a software managed one (less than 20 words please).

- c) Suppose a sleep-deprived engineer proposes handling TLB hits in software, in the wake of a eureka moment that comes after drinking many cups of coffee. Is this a good idea or a bad idea? Explain in no more than 20 words.

5) Branch Handling (4 points)

a) In what sense is the handling of a branch misprediction similar to the handling of an exception?

b) In what sense is the handling of a branch misprediction different from the handling of an exception?

6) Runahead, In-order, Out-of-order Execution (4 points)

The idea of runahead execution, as discussed in class, can be employed in conjunction with in-order instruction scheduling or out-of-order instruction scheduling. We have discussed results from multiple different systems showing that the relative performance benefit of runahead execution is higher when it is implemented on top of a machine that employs in-order scheduling than when it is implemented on top of a machine that employs out-of-order instruction scheduling. Explain why this is so. (20 words or less)

7) Asymmetric CMPs (5 points)

A multi-core processor consists of 16 simple cores on a single chip. A program consists of code that is 80% perfectly parallelizable and 20% serial.

a) What is the maximum speed-up obtained by executing this program on the 16-core chip versus running it on a chip that only has a single simple core?

b) Suppose instead we execute this program on a chip where 4 of the 16 simple cores have been replaced by one heavyweight core that processes instructions 2 times faster than a simple core. What is the speed-up of running on this chip versus running on a chip that has 16 simple cores? Assume that when the chip is processing the parallelizable code, the heavyweight core is idle.

8) ISA Tradeoffs (8 points)

Embedded systems usually have relatively small physical memories due to size and power constraints. Assume that you are designing such an embedded system from scratch with a very small physical memory. What are some of the design choices you will make with respect to the following parameters? For each parameter, specify which of the choices you would pick with a brief explanation. If there are trade-offs, clearly mention them in your answer.

a) Complex instructions or simple instructions

b) Many architectural registers or few architectural registers

c) 64-bit registers or 32-bit registers

d) Virtual memory or no virtual memory

e) Small pages or large pages (assuming you have virtual memory)

9) L1 Caches and Virtual Memory (8 points)

We wish to design a 64 KB L1 cache for an ISA having a 4 KB page size. We want to keep the TLB out of the critical path of cache access, but also ensure that a given physical address can reside in only one unique location in the cache without any software support. What is the associativity of our design that satisfies the above constraints? Show your work.

Suppose we relax the requirement of not having software support above, but we would still like to satisfy the other above constraints. In addition, we would like to build the cache as direct mapped. What does the system software need to ensure such that a given physical address resides in only one unique location in the cache? Be specific. Show your work.

Problem 2 (Interlocking) (15 points)

a) In lecture, we discussed the motivation behind MIPS, which stands for Microprocessor without Interlocked Pipe Stages. The motivation for MIPS was to keep the hardware design as simple as possible by eliminating the logic that checks for dependencies between instructions. What was the key idea to eliminate hardware interlocks used in early MIPS designs? Explain.

b) Fine-grained multithreading (FGMT), also discussed in lecture, and employed in Denelcor HEP and Sun Niagara is another alternative method for eliminating the need for hardware-based interlocking. Explain how this works briefly. Hint: Describe the invariant fine-grained multithreading enforces to eliminate hardware interlocks.

c) Describe two advantages of the MIPS approach over FGMT.

Name:

8

d) Describe two advantages of FGMT over the MIPS approach.

e) An alternative way to eliminate the need for interlocking is to predict the unavailable source values of an instruction. What additional logic does this approach require that FGMT and the MIPS approach do not require?

Problem 3 (Mispredicted Path and Memory) (12 points)

An engineer designs a piece of logic that, upon discovering that a branch has been mispredicted, cancels all memory requests that were initiated by load instructions along the mispredicted path. We evaluate this new feature on application A, and notice a performance improvement. On application B, we notice a performance degradation. There is nothing wrong with the measurement methodology or hardware implementation. Explain how both results are possible. Show code examples, if you think this would aid the explanation.

Problem 4 (Runahead) (25 points)

We have covered the notion of runahead execution in class, as an alternative to scaling up the instruction window size of an out-of-order execution processor. Remember that a runahead processor speculatively executes the program under an L2 cache miss. Also remember that the instruction window determines how many instructions an out of order execution engine contains that are decoded but not yet committed to architectural state.

a) What benefit(s) of a large instruction window can runahead execution provide? Be specific.

b) What benefit(s) of a large instruction window can runahead execution not provide? Be specific.

c) Remember that runahead mode ends when the L2 cache miss that caused entry into runahead mode is fully serviced by main memory. How does the processor exit runahead mode? Describe the actions taken by the processor.

d) Where does the processor start execution after runahead mode?

e) Two computer architects debate whether it is a good idea to exit runahead mode when the L2 cache miss that caused entry into runahead mode (i.e., the runahead-causing miss) is fully serviced. Each comes up with a different suggestion. Vivek suggests that the processor stay 20 more cycles after the runahead-causing miss is fully serviced. Explain why this could be a good idea (i.e., when could it provide benefits).

f) Explain why Vivek's suggestion could be a bad idea (i.e., when could it do harm or not provide benefits).

g) Lavanya, on the other hand, develops a cheap mechanism that enables the processor to detect that the runahead-causing cache miss will be fully serviced in 20 cycles. Using this mechanism, Lavanya suggests that the processor exit runahead mode 20 cycles before the runahead-causing miss is serviced. Explain why this could be a good idea.

h) Explain why Lavanya's suggestion could be a bad idea.

i) A third architect, Evangelos, listens to both and develops a mechanism that aims to achieve the best of both solutions. Briefly describe what such a mechanism could be. Focus on key ideas (not implementation), but be specific (i.e., the mechanism needs to be implementable; it should not assume the existence of information whose collection is not possible).

j) Explain why the mechanism you propose could be a good idea.

k) Explain why the mechanism you propose could be a bad idea.

Problem 5 (Caches) (15 points)

A byte-addressable system with 16-bit addresses ships with a two-way set associative, write back cache with perfect LRU replacement. The tag store requires a total of 4352 bits of storage. What is the block size of the cache? Show all your work.

(Hint: $4352 = 2^{12} + 2^8$)

Problem 6 (Out-of-order Core Design) (20 points)

In lectures, we described the design of out-of-order execution machines in which register data values are consolidated in a single structure, called the physical register file (PRF). This approach is used in several modern out-of-order execution processors, including the Pentium 4, Alpha 21264, and MIPS R10000. Let's call the approach the "PRF design." We have contrasted this approach to the approach of storing register values in reservation stations, reorder buffer, future register file, and architectural register file. The latter approach is used in the Pentium Pro design. Let's call this approach the "PPro Design."

a) At what stage during its execution does an instruction write its result to the physical register file in the PRF design?

At the same stage, where does the instruction write its result in the PPro design?

b) At what stage during its execution does an instruction write its result to the architectural register file in the PPro design?

At the same stage, where does the instruction write its result in the PRF design?

c) How does the architectural register state get updated in each design when an instruction commits? Explain briefly.

PRF:

PPro:

d) When does a physical register get deallocated (i.e., freed) in the PRF design? Explain.

e) What are the advantages of the PRF design over the PPro design?

f) What are the disadvantages of the PRF design over the PPro design?

Problem 7 (Out of Order Execution) (25 points)

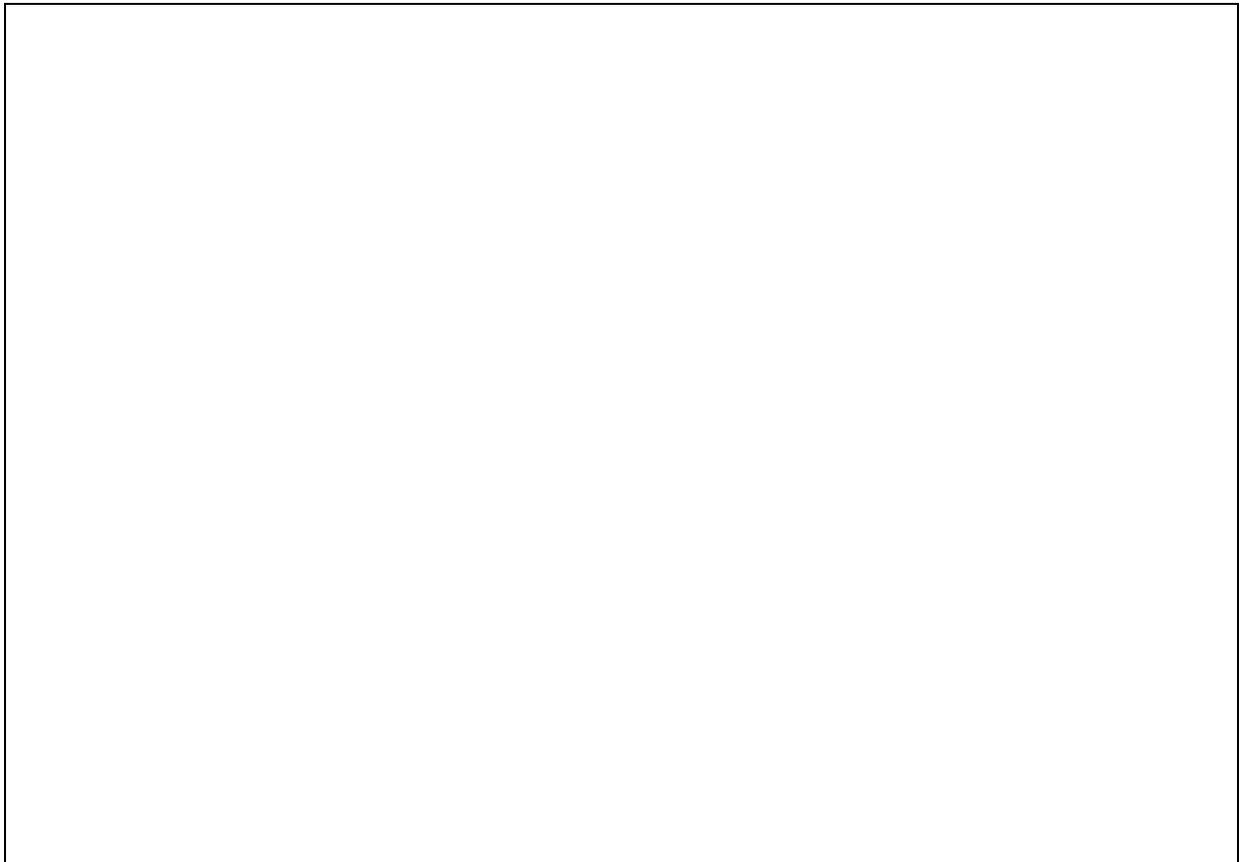
The diagram below shows a snapshot at a particular point in time of various parts of the microarchitecture for an implementation supporting out-of-order execution in the spirit of Tomasulo's algorithm.

Note that both the adder and multiplier are pipelined, the adder has three stages and the multiplier has four stages. At the particular point in time when the snapshot was taken, the adder had only one instruction in the pipeline (i.e., in the final stage). The multiplier had two instructions in the pipeline, one in the final stage, one in stage 3. Note that the multiplier and adder have separate output data buses, which allow both an add result and a multiply result to update in the same cycle.

The adder has two reservation stations, the multiplier has four reservation stations. An instruction continues to occupy a reservation station slot until it completes execution and its result is written to its destination register.

The processor has been supplied with a six instruction program segment. Instructions are fetched, decoded and executed as discussed in class. Assume full data forwarding, whenever possible.

Your job: First, identify the instructions and draw the data flow graph for the six instructions. Use + for ADD and \times for MULTIPLY.



Second, complete the table identifying the six instructions, their opcodes (ADD, MUL), and their register operands (R_{dest} , R_{src1} , R_{src2}). The first instruction is already filled for you: ADD R1, R5, R6. (R1 is the destination register and R5, R6 are the source registers.) Also, show which stage of processing (F, D, E, R, W) each instruction is in during each cycle. Assume a 5-stage pipeline: Fetch(F), Decode(D), Execute(E), Reorder Buffer(R), Writeback(W), as we covered in lecture. If an instruction is stalled, indicate it with a ‘-’.

Finally, identify the cycle after which the snapshot of the microarchitecture was taken. Shade the corresponding cycle in the last row of the table.

Reg	V	Tag	Value
R0	1	-	0
R1	1	-	110
R2	0	χ	20
R3	1	-	30
R4	0	ρ	40
R5	0	π	50
R6	0	β	60
R7	1	-	70

ID	V	Tag	Value	V	Tag	Value
α						
β	0	π	-	0	ρ	-
δ	1	-	30	1	-	40
χ	1	-	110	1	-	40

ID	V	Tag	Value	V	Tag	Value
π	0	ρ	-	0	χ	-
ρ	1	-	50	1	-	60

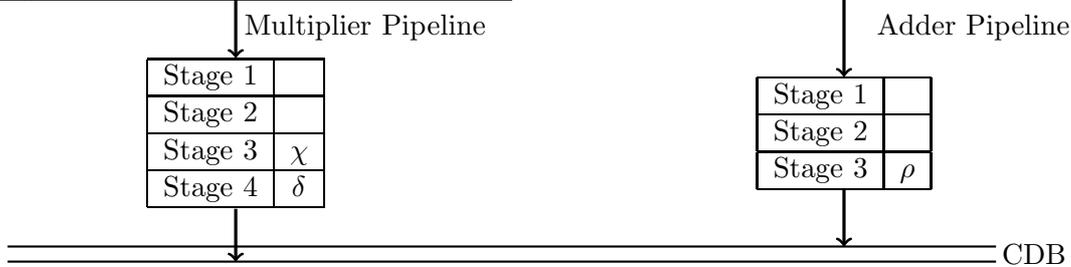


Figure 1: Snapshot of the Register file, Reservation stations and the Pipelines

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
ADD R1, R5, R6																						
Snapshot Cycle																						

Table 1: **THIS IS THE TABLE FOR YOU TO FILL OUT**

Bonus Problem 8 (Good Enough Computing) (10 points)

Ben Zorn spoke about the concept of “good enough” computing and mentioned the idea of approximate loop computing. Consider the following program:

```
int a[1000000];
int avg = 0;

for (int k = 0; k < 1000000; k ++) {
    avg += a[k];
}

avg /= 1000000;
```

Assume that the array 'a' is not in cache and needs to be accessed from main memory. If we are interested in just computing the average (avg) *quickly*, how could you modify the above code in the context of “Good enough” computing as suggested by Ben Zorn? How will you provide an estimate of how good the output is?

If we were to annotate the data in the program to distinguish critical data (remember Ben’s lecture) from non-critical data, which of the variables will you mark as critical?

When does “good enough” computing make sense? When does it not? Explain briefly.

Name:

Name: