

## CS152 – Exam 2 Review

2003-11-18

Jack and Kurt

[www-inst.eecs.berkeley.edu/~cs152/](http://www-inst.eecs.berkeley.edu/~cs152/)



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1a:

**Problem 1a:** Assume that we have a 32-bit processor (with 32-bit words) and that this processor is byte-addressed (i.e. addresses specify bytes). Suppose that it has a 512-byte cache that is two-way set-associative, has 4-word cache lines, and uses LRU replacement. Split the 32-bit address into "tag", "index", and "cache-line offset" pieces. Which address bits comprise each piece?

- Tag:
- Index:
- Block Offset:



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1b:

**Problem 1a:** Assume that we have a 32-bit processor (with 32-bit words) and that this processor is byte-addressed (i.e. addresses specify bytes). Suppose that it has a 512-byte cache that is two-way set-associative, has 4-word cache lines, and uses LRU replacement. Split the 32-bit address into "tag", "index", and "cache-line offset" pieces. Which address bits comprise each piece?

- Tag: 24 bits total: 31-8
- Index: 4 bits total: 7-4
- Block Offset: 4 bits total: 3-0



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1b:

**Problem 1a:** Assume that we have a 32-bit processor (with 32-bit words) and that this processor is byte-addressed (i.e. addresses specify bytes). Suppose that it has a 512-byte cache that is two-way set-associative, has 4-word cache lines, and uses LRU replacement. Split the 32-bit address into "tag", "index", and "cache-line offset" pieces. Which address bits comprise each piece?

- Tag: 24 bits total: 31-8
- Index: 4 bits total: 7-4
- Block Offset: 4 bits total: 3-0

**Problem 1b:** How many sets does this cache have? Explain.



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1b:

**Problem 1a:** Assume that we have a 32-bit processor (with 32-bit words) and that this processor is byte-addressed (i.e. addresses specify bytes). Suppose that it has a 512-byte cache that is two-way set-associative, has 4-word cache lines, and uses LRU replacement. Split the 32-bit address into "tag", "index", and "cache-line offset" pieces. Which address bits comprise each piece?

- Tag: 24 bits total: 31-8
- Index: 4 bits total: 7-4
- Block Offset: 4 bits total: 3-0

**Problem 1b:** How many sets does this cache have? Explain.

4 bits in the index field →  
2<sup>4</sup> possible values →  
16 sets



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1c:

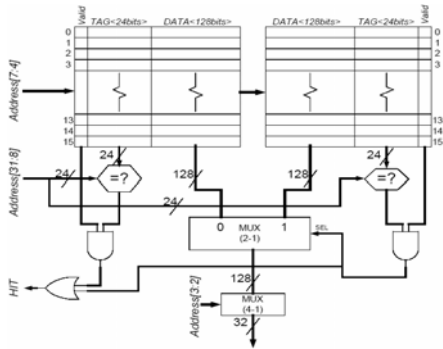
**Problem 1c:** Draw a block diagram for this cache. Show a 32-bit address coming into the diagram and a 32-bit data result and "Hit" signal coming out. Include, all of the comparators in the system and any muxes as well. Include the data storage memories (indexed by the "Index"), the tag matching logic, and any muxes. You can indicate RAM with a simple block, but make sure to label address widths and data widths. Make sure to label the function of various blocks and the width of any buses.



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1c:



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1d:

**Problem 1d:** Below is a series of memory read references set to the cache from part (a). Assume that the cache is initially empty and classify each memory references as a hit or a miss. Identify each miss as either **compulsory**, **conflict**, or **capacity**. One example is shown. *Hint: start by splitting the address into components. Show your work.*



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1d:

**Problem 1d:** Below is a series of memory read references set to the cache from part (a). Assume that the cache is initially empty and classify each memory references as a hit or a miss. Identify each miss as either **compulsory**, **conflict**, or **capacity**. One example is shown. *Hint: start by splitting the address into components. Show your work.*

Address	Hit/Miss?	Miss Type?	Address	Hit/Miss?	Miss Type?
0x300	Miss	Compulsory	0x3B2	Miss	Compulsory
0x1BC	Miss	Compulsory	0x10C	Hit	—
0x206	Miss	Compulsory	0x205	Miss	Conflict
0x109	Miss	Compulsory	0x301	Miss	Conflict
0x308	Miss	Conflict	0x3AE	Miss	Compulsory
0x1A1	Miss	Compulsory	0x1A8	Miss	Conflict
0x1B1	Hit	—	0x3A1	Hit	—
0x2AB	Miss	Compulsory	0x1BA	Hit	—



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1e:

**Problem 1d:** Below is a series of memory read references set to the cache from part (a). Assume that the cache is initially empty and classify each memory references as a hit or a miss. Identify each miss as either **compulsory**, **conflict**, or **capacity**. One example is shown. *Hint: start by splitting the address into components. Show your work.*

Address	Hit/Miss?	Miss Type?	Address	Hit/Miss?	Miss Type?
0x300	Miss	Compulsory	0x3B2	Miss	Compulsory
0x1BC	Miss	Compulsory	0x10C	Hit	—
0x206	Miss	Compulsory	0x205	Miss	Conflict
0x109	Miss	Compulsory	0x301	Miss	Conflict
0x308	Miss	Conflict	0x3AE	Miss	Compulsory
0x1A1	Miss	Compulsory	0x1A8	Miss	Conflict
0x1B1	Hit	—	0x3A1	Hit	—
0x2AB	Miss	Compulsory	0x1BA	Hit	—

**Problem 1e:** Calculate the miss rate and hit rate.



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1e:

**Problem 1d:** Below is a series of memory read references set to the cache from part (a). Assume that the cache is initially empty and classify each memory references as a hit or a miss. Identify each miss as either **compulsory**, **conflict**, or **capacity**. One example is shown. *Hint: start by splitting the address into components. Show your work.*

Address	Hit/Miss?	Miss Type?	Address	Hit/Miss?	Miss Type?
0x300	Miss	Compulsory	0x3B2	Miss	Compulsory
0x1BC	Miss	Compulsory	0x10C	Hit	—
0x206	Miss	Compulsory	0x205	Miss	Conflict
0x109	Miss	Compulsory	0x301	Miss	Conflict
0x308	Miss	Conflict	0x3AE	Miss	Compulsory
0x1A1	Miss	Compulsory	0x1A8	Miss	Conflict
0x1B1	Hit	—	0x3A1	Hit	—
0x2AB	Miss	Compulsory	0x1BA	Hit	—

**Problem 1e:** Calculate the miss rate and hit rate.

$$\text{Hit Rate} = \frac{4}{16} = 0.25$$

$$\text{Miss Rate} = 1 - \text{Hit Rate} = \frac{12}{16} = 0.75$$



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1f:

**Problem 1f:** You have a 500 MHz processor with 2-levels of cache, 1 level of DRAM, and a DISK for virtual memory. Assume that it has a Harvard architecture (separate instruction and data cache at level 1). Assume that the memory system has the following parameters:

Component	Hit Time	Miss Rate	Block Size
First-Level Cache	1 cycle	4% Data 1% Instructions	64 bytes
Second-Level Cache	20 cycles + 1 cycle/64bytes	2%	128 bytes
DRAM	100ns + 25ns/8 bytes	1%	16K bytes
DISK	50ms + 20ms/byte	0%	16K bytes

Finally, assume that there is a TLB that misses 0.1% of the time on data (doesn't miss on instructions) and which has a fill penalty of 40 cycles. What is the average memory access time (AMAT) for Instructions? For Data (assume all reads)?

AMAT<sub>disk</sub> = ?



CS 152 Review

Jack Kang and Kurt Meitz

### Question 1f:

**Problem 1f:** You have a 500 MHz processor with 2-levels of cache, 1 level of DRAM, and a DISK for virtual memory. Assume that it has a Harvard architecture (separate instruction and data cache at level 1). Assume that the memory system has the following parameters:

Component	Hit Time	Miss Rate	Block Size
First-Level Cache	1 cycle	4% Data 1% Instructions	64 bytes
Second-Level Cache	20 cycles + 1 cycle-64bits	2%	128 bytes
DRAM	100ns+ 25ns/8 bytes	1%	16K bytes
DISK	50ms + 20ms/byte	0%	16K bytes

Finally, assume that there is a TLB that misses 0.1% of the time on data (doesn't miss on instructions) and which has a fill penalty of 40 cycles. What is the average memory access time (AMAT) for Instructions? For Data (assume all reads)?

$$\begin{aligned}
 \text{AMAT}_{\text{Disk}} &= \text{AccessTime} + \text{AMATMissPenalty} + \text{TransferRate} * \text{TransferSize} \\
 &= 50\text{E}6\text{ns} + 0 + (20\text{ns}/\text{byte} * 16\text{Kbytes}) \\
 &= \sim 5\text{E}7\text{ns} \\
 &= 5\text{E}7\text{ns} / (2\text{ns}/\text{clock}) \rightarrow 2.5\text{E}7 \text{ clocks}
 \end{aligned}$$

Cal

CS 152 Review

Jack Kang and Kurt Meitz

### Question 1f:

**Problem 1f:** You have a 500 MHz processor with 2-levels of cache, 1 level of DRAM, and a DISK for virtual memory. Assume that it has a Harvard architecture (separate instruction and data cache at level 1). Assume that the memory system has the following parameters:

Component	Hit Time	Miss Rate	Block Size
First-Level Cache	1 cycle	4% Data 1% Instructions	64 bytes
Second-Level Cache	20 cycles + 1 cycle-64bits	2%	128 bytes
DRAM	100ns+ 25ns/8 bytes	1%	16K bytes
DISK	50ms + 20ms/byte	0%	16K bytes

Finally, assume that there is a TLB that misses 0.1% of the time on data (doesn't miss on instructions) and which has a fill penalty of 40 cycles. What is the average memory access time (AMAT) for Instructions? For Data (assume all reads)?

$$\begin{aligned}
 \text{AMAT}_{\text{DRAM}} &= \text{AccessTime} + \text{AMATMiss} + \text{TransferRate} * \text{TransferSize} \\
 &= 100\text{ns} + 5\text{E}7\text{ns} * 0.01 + (25\text{ns}/8\text{bytes} * 128\text{bytes}) = \\
 &= \sim 5\text{E}5\text{ns} \\
 &= 5\text{E}5\text{ns} / (2\text{ns}/\text{clock}) \rightarrow 2.5\text{E}5 \text{ clocks}
 \end{aligned}$$

Cal

CS 152 Review

Jack Kang and Kurt Meitz

### Question 1f:

**Problem 1f:** You have a 500 MHz processor with 2-levels of cache, 1 level of DRAM, and a DISK for virtual memory. Assume that it has a Harvard architecture (separate instruction and data cache at level 1). Assume that the memory system has the following parameters:

Component	Hit Time	Miss Rate	Block Size
First-Level Cache	1 cycle	4% Data 1% Instructions	64 bytes
Second-Level Cache	20 cycles + 1 cycle-64bits	2%	128 bytes
DRAM	100ns+ 25ns/8 bytes	1%	16K bytes
DISK	50ms + 20ms/byte	0%	16K bytes

Finally, assume that there is a TLB that misses 0.1% of the time on data (doesn't miss on instructions) and which has a fill penalty of 40 cycles. What is the average memory access time (AMAT) for Instructions? For Data (assume all reads)?

$$\begin{aligned}
 \text{AMAT}_{\text{L2}} &= \text{AccessTime} + \text{AMATMiss} + \text{TransferRate} * \text{TransferSize} \\
 &= (20\text{c} * 2\text{ns}/\text{c}) + 5\text{E}5\text{ns} * 0.02 + (2\text{ns}/8\text{bytes} * 64\text{bytes}) = \\
 &= \sim 1\text{E}4 \text{ ns} \\
 &= 1\text{E}4\text{ns} / (2\text{ns}/\text{clock}) \rightarrow 5\text{E}3 \text{ clocks}
 \end{aligned}$$

Cal

CS 152 Review

Jack Kang and Kurt Meitz

### Question 1f:

**Problem 1f:** You have a 500 MHz processor with 2-levels of cache, 1 level of DRAM, and a DISK for virtual memory. Assume that it has a Harvard architecture (separate instruction and data cache at level 1). Assume that the memory system has the following parameters:

Component	Hit Time	Miss Rate	Block Size
First-Level Cache	1 cycle	4% Data 1% Instructions	64 bytes
Second-Level Cache	20 cycles + 1 cycle-64bits	2%	128 bytes
DRAM	100ns+ 25ns/8 bytes	1%	16K bytes
DISK	50ms + 20ms/byte	0%	16K bytes

Finally, assume that there is a TLB that misses 0.1% of the time on data (doesn't miss on instructions) and which has a fill penalty of 40 cycles. What is the average memory access time (AMAT) for Instructions? For Data (assume all reads)?

$$\begin{aligned}
 \text{AMAT}_{\text{L1Inst}} &= \text{AccessTime} + \text{AMATMiss} + \text{TransferRate} * \text{TransferSize} \\
 &= (1\text{c} * 2\text{ns}/\text{c}) + (5\text{E}3\text{ns} * 0.01) + 0 \\
 &= \sim 1\text{E}2 \text{ ns} \\
 &= 1\text{E}2\text{ns} / (2\text{ns}/\text{clock}) \rightarrow 50 \text{ clocks (!)}
 \end{aligned}$$

Cal

CS 152 Review

Jack Kang and Kurt Meitz

### Question 1f:

**Problem 1f:** You have a 500 MHz processor with 2-levels of cache, 1 level of DRAM, and a DISK for virtual memory. Assume that it has a Harvard architecture (separate instruction and data cache at level 1). Assume that the memory system has the following parameters:

Component	Hit Time	Miss Rate	Block Size
First-Level Cache	1 cycle	4% Data 1% Instructions	64 bytes
Second-Level Cache	20 cycles + 1 cycle-64bits	2%	128 bytes
DRAM	100ns+ 25ns/8 bytes	1%	16K bytes
DISK	50ms + 20ms/byte	0%	16K bytes

Finally, assume that there is a TLB that misses 0.1% of the time on data (doesn't miss on instructions) and which has a fill penalty of 40 cycles. What is the average memory access time (AMAT) for Instructions? For Data (assume all reads)?

$$\begin{aligned}
 \text{AMAT}_{\text{L1Data}} &= \text{AccessTime} + \text{AMATMiss} + \text{TransferRate} * \text{TransferSize} \\
 &= (1\text{c} * 2\text{ns}/\text{c}) + (5\text{E}3\text{ns} * 0.04) + 0 \\
 &= \sim 4\text{E}2 \text{ ns} \\
 &= 4\text{E}2\text{ns} / (2\text{ns}/\text{clock}) \rightarrow 200 \text{ clocks (!)}
 \end{aligned}$$

Cal

CS 152 Review

Jack Kang and Kurt Meitz

### Question 1g:

**Problem 1g:** Suppose that we measure the following instruction mix for benchmark "X":

**Loads: 20%, Stores: 15%, Integer: 30%, Floating-Point: 15%, Branches: 20%**

Assume that we have a single-issue processor with a minimum CPI of 1.0. Assume that we have a branch predictor that is correct 95% of the time, and that an incorrect prediction costs 3 cycles. Finally, assume that data hazards cause an average penalty of 0.7 cycles for floating point operations. Integer operations run at maximum throughput. What is the average CPI of Benchmark X, including memory misses (from part g)?

Cal

CS 152 Review

Jack Kang and Kurt Meitz

### Question 1g:

**Problem 1g:** Suppose that we measure the following instruction mix for benchmark "X":  
Loads: 20%, Stores: 15%, Integer: 30%, Floating-Point: 15% Branches: 20%  
Assume that we have a single-issue processor with a minimum CPI of 1.0. Assume that we have a branch predictor that is correct 95% of the time, and that an incorrect prediction costs 3 cycles. Finally, assume that data hazards cause an average penalty of 0.7 cycles for floating point operations. Integer operations run at maximum throughput. What is the average CPI of Benchmark X, including memory misses (from part g)?

$$\begin{aligned} \text{CPI} &= \text{MinCPI} + \Sigma [\text{CPI of exceptional events}] \\ &= \text{MinCPI} + \text{CPIHazardStalls} + \text{CPIMemoryStalls} \end{aligned}$$



CS 152 Review

Jack Kang and Kurt Meigs

### Question 1g:

**Problem 1g:** Suppose that we measure the following instruction mix for benchmark "X":  
Loads: 20%, Stores: 15%, Integer: 30%, Floating-Point: 15% Branches: 20%  
Assume that we have a single-issue processor with a minimum CPI of 1.0. Assume that we have a branch predictor that is correct 95% of the time, and that an incorrect prediction costs 3 cycles. Finally, assume that data hazards cause an average penalty of 0.7 cycles for floating point operations. Integer operations run at maximum throughput. What is the average CPI of Benchmark X, including memory misses (from part g)?

$$\begin{aligned} \text{CPI} &= \text{MinCPI} + \Sigma [\text{CPI of exceptional events}] \\ &= \text{MinCPI} + \text{CPIHazardStalls} + \text{CPIMemoryStalls} \\ &= 1 + \Sigma (\text{InstTypeFreq} * \text{CPI}) + \Sigma (\text{MemAccessFreq} * \text{AccessAMAT}) \end{aligned}$$



CS 152 Review

Jack Kang and Kurt Meigs

### Question 1g:

**Problem 1g:** Suppose that we measure the following instruction mix for benchmark "X":  
Loads: 20%, Stores: 15%, Integer: 30%, Floating-Point: 15% Branches: 20%  
Assume that we have a single-issue processor with a minimum CPI of 1.0. Assume that we have a branch predictor that is correct 95% of the time, and that an incorrect prediction costs 3 cycles. Finally, assume that data hazards cause an average penalty of 0.7 cycles for floating point operations. Integer operations run at maximum throughput. What is the average CPI of Benchmark X, including memory misses (from part g)?

$$\begin{aligned} \text{CPI} &= \text{MinCPI} + \Sigma [\text{CPI of exceptional events}] \\ &= \text{MinCPI} + \text{CPIHazardStalls} + \text{CPIMemoryStalls} \\ &= 1 + \Sigma (\text{InstTypeFreq} * \text{CPI}) + \Sigma (\text{MemAccessFreq} * \text{AccessAMAT}) \\ &= 1 + [ (\text{FPFreq} * \text{FPCPI}) \\ &\quad + (\text{BBranchFreq} * \text{BBCPI}) ] \\ &\quad + [ (\text{MemInstFreq} * \text{AMATL1Inst} / (2\text{ns/clock})) \\ &\quad + (\text{DataInstFreq} * \text{AMATL1Data} / (2\text{ns/clock})) ] \\ &= 1 + 0.15 * 0.7 + 0.2 * 0.05 * 0.3 + 1 * 100 / 2 + 0.35 * 400 / 2 \rightarrow 124 \text{CPI (!)} \end{aligned}$$



CS 152 Review

Jack Kang and Kurt Meigs

### Question 2a:

2a) Explain why we would be unable to pick a single optimum number of branch delay slots for the above processor.



CS 152 Review

Jack Kang and Kurt Meigs

### Question 2a

2a) Explain why we would be unable to pick a single optimum number of branch delay slots for the above processor.

*Branch delay slots affect correctness they represent functional behavior – things always executed when a branch is executed, we have to pick a single number. The result wouldn't be optimal under all circumstances, since we issue 0, 1, or 2 instructions per cycle after the branch.*



CS 152 Review

Jack Kang and Kurt Meigs

### Question 2b

2a) Explain why we would be unable to pick a single optimum number of branch delay slots for the above processor.



CS 152 Review

Jack Kang and Kurt Meigs

## Question 2b

2a) Explain why we would be unable to pick a single optimum number of branch delay slots for the above processor.

*This depends on whether or not the two memory stages are separable. A WAR hazard would occur if it were possible for a later store to change the value of an early read. If stores go to memory early but loads take two cycles, this might be a problem. The way to fix this (if it happens) is to make sure that stores take two cycles just like loads. Note that the answer to this question is likely "NO" unless you do something weird with your memory system.*



CS 152 Review

Jack Kang and Kurt Meitz

## Question 2c

2c) Below is a *start* at a simple diagram for the pipelines of this processor.

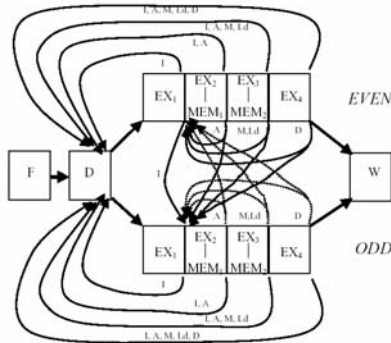
- 1) Finish the diagram. Stages are boxes with letters inside. Use "F" for a fetch stage, "D" for a decode stage, EX<sub>1</sub> through EX<sub>4</sub> for the execution stages of each of the pipelines (including memory accesses), and "W" for a writeback stage. Clearly label which is the even pipeline. Include arrows for forward information flow of this is not obvious.
- 2) Next, describe what is being computed in each EX stage (including partial results).
- 3) Show all forwarding paths (as arrows). Your pipeline should never stall unless a value is not ready. Label each bypass arrow with the types of instructions that will forward their results along that path (i.e. use "M" for `mem`, "D" for `dat`, "A" for `add`, "I" for integer operations, and "Ld" for load results). [Hint: think carefully about inputs to store instructions!]



CS 152 Review

Jack Kang and Kurt Meitz

## Question 2c



CS 152 Review

Jack Kang and Kurt Meitz

## Question 2c

EX Stages: EX<sub>1</sub>: Integer ops, Branches, Memory address computation, First stage of A, M, D  
 EX<sub>2</sub>: First stage of load/store, Finish A, Second stage of M, D  
 EX<sub>3</sub>: Final stage of load/store, Finish M, Third stage of D  
 EX<sub>4</sub>: Final stage of D

Notes: The primary forwarding is from the end of EX stages back to the end of decode stage. Store forwarding is shown between the two pipes and only involves special cases in which an operation finishes and needs to be forwarded into the beginning of EX<sub>3</sub>. Note in particular the very special case of integer forwarding from an integer op in even pipeline to store in odd. With this arc, you can actually issue a integer op and a store of the result in the same cycle.



CS 152 Review

Jack Kang and Kurt Meitz

## Question 2d

2d) Note that we assume that a load is not completed until the end of EX<sub>3</sub> and that a store must have its value by the beginning of EX<sub>2</sub>. Consider the following common sequence for a memory copy:

```
loop: ld r1, 0(r2)
      st r1, 0(r3)
      add r2, r2, #4
      subi r4, r4, #1
      add r3, r3, #4
      bne r4, r0, loop
      nop
```

Why can't the load and store be dispatched in the same cycle? What is the minimum number of instructions that must be placed between them to avoid stalling?



CS 152 Review

Jack Kang and Kurt Meitz

## Question 2d

2d) Note that we assume that a load is not completed until the end of EX<sub>3</sub> and that a store must have its value by the beginning of EX<sub>2</sub>. Consider the following common sequence for a memory copy:

```
loop: ld r1, 0(r2)
      st r1, 0(r3)
      add r2, r2, #4
      subi r4, r4, #1
      add r3, r3, #4
      bne r4, r0, loop
      nop
```

Why can't the load and store be dispatched in the same cycle? What is the minimum number of instructions that must be placed between them to avoid stalling?

*They cannot be dispatched in the same cycle because of the dependency through r1. In this pipeline, the store must execute 2 cycles later than the load (because loads take 2 cycles). In the best case (load in the odd pipeline, store in the even pipeline), there must be 1 bubble cycle or 2 instructions. So, answer: 2 instructions*

*The easiest way to understand this is to imagine that the load is in the EX<sub>3</sub> stage of the odd pipeline while the store is in the EX<sub>2</sub> stage of the even pipeline. Look at the answer for the previous problem. There is a special store arc to handle this circumstance. The load is 2 cycles ahead of the store. We need to fill instructions in the two different EX<sub>3</sub> stages.*



CS 152 Review

Jack Kang and Kurt Meitz

### Question 2e

2e) What can you change about the pipeline to reduce your answer to (2d)? Assume that you are not allowed to change the latencies of any instructions.



CS 152 Review

Jack Kang and Kurt Meitz

### Question 2e

2e) What can you change about the pipeline to reduce your answer to (2d)? Assume that you are not allowed to change the latencies of any instructions.

*By shifting the memory stages in the even pipeline forward 1 cycle, we can get 6 instructions. What this means is that the two mem stages for the even pipeline are in EX<sub>1</sub> and EX<sub>2</sub>. Then, if the load is in the odd pipeline and the store is in the even pipeline (next cycle), we have no stalls.*



CS 152 Review

Jack Kang and Kurt Meitz

### Question 2g

2g) [Extra Credit: 5pts] Briefly describe the logic that would be required in the decode stage of this pipeline. In five (5) sentences or less (and possibly a small figure), describe a mechanism that would permit the decode stage to decide which of two instructions presented to it could be dispatched.



CS 152 Review

Jack Kang and Kurt Meitz

### Question 2g

2g) [Extra Credit: 5pts] Briefly describe the logic that would be required in the decode stage of this pipeline. In five (5) sentences or less (and possibly a small figure), describe a mechanism that would permit the decode stage to decide which of two instructions presented to it could be dispatched.

-We have to check to see if the 2<sup>nd</sup> instruction depends on the first one.

-We have to check the operands of the two instructions against any instructions still in the pipeline, and see if it can issue. This step is slightly complex because different instructions in the pipeline finish at different times.



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

#### Extra Credit (Problem 3X):

Assume that you have a Tomasulo architecture with functional units of the same execution latency (number of cycles) as our deeply pipelined processor (be careful to adjust use latencies to get number of execution cycles!). Assume that it issues one instruction per cycle and has an unpipelined divider with a small number of reservation stations. Suppose the other functional units are duplicated with many reservation stations and that there are many CDBs. What is the minimum number of divide reservation stations to achieve one instruction per cycle with the optimized code of (3b)? Show your work. [Hint: assume that the maximum issue rate is sustained and look at the scheduling of a single iteration]

Load: 3 cycles, Add: 2 cycles, Multiply: 4 cycles, Divide: 9 cycles (careful here!)

```
loop: ldf SF20, 0($r10)
      ldf SF10, 8($r10)
      multf SF6, SF20, SF1
      addf SF12, SF6, SF2
      addi $r10, $r10, #16
      divf SF13, SF12, SF10
      addi $r20, $r20, #8
      subi $r1, $r1, #1
      bne $r1, $zero, loop
      stf -8($r20), SF13
```



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Load: 3 cycles, Add: 2 cycles, Multiply: 4 cycles, Divide: 9 cycles (careful here!)

```
loop: ldf SF20, 0($r10)
      ldf SF10, 8($r10)
      multf SF6, SF20, SF1
      addf SF12, SF6, SF2
      addi $r10, $r10, #16
      divf SF13, SF12, SF10
      addi $r20, $r20, #8
      subi $r1, $r1, #1
      bne $r1, $zero, loop
      stf -8($r20), SF13
```

#### Keys to Problem:

1) # of station entries needed = # of div instructions in flight at same time

2)



CS 152 Review

Jack Kang and Kurt Meitz





### Question 3:

Tomasulo Trace: **CC 11: First->Second Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5								
ldf	f10	r10		2	3	5	6								
multf	f6	f20	f1	3	6	9	10								
addf	f12	f6	f2	4	11	12*									
addi	r10	r10		5	6	6	7								
divf	f13	f12	f10	6											
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11											



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 12: First->Second Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5								
ldf	f10	r10		2	3	5	6								
multf	f6	f20	f1	3	6	9	10								
addf	f12	f6	f2	4	11	12									
addi	r10	r10		5	6	6	7								
divf	f13	f12	f10	6											
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14*									
ldf	f10	r10		12											



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 13: First->Second Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5								
ldf	f10	r10		2	3	5	6								
multf	f6	f20	f1	3	6	9	10								
addf	f12	f6	f2	4	11	12	13								
addi	r10	r10		5	6	6	7								
divf	f13	f12	f10	6											
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14*									
ldf	f10	r10		12	13	15*									
multf	f6	f20	f1	13											



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 14: First->Second Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14			
ldf	f10	r10		2	3	5	6								
multf	f6	f20	f1	3	6	9	10								
addf	f12	f6	f2	4	11	12	13								
addi	r10	r10		5	6	6	7								
divf	f13	f12	f10	6	14	22*									
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14									
ldf	f10	r10		12	13	15*									
multf	f6	f20	f1	13											



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 15: First->Second Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14			
ldf	f10	r10		2	3	5	6	addi	r10	r10		15			
multf	f6	f20	f1	3	6	9	10								
addf	f12	f6	f2	4	11	12	13								
addi	r10	r10		5	6	6	7								
divf	f13	f12	f10	6	14	22*									
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15									
multf	f6	f20	f1	13											



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 16: First->Second Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14			
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16			
addf	f12	f6	f2	4	11	12	13								
addi	r10	r10		5	6	6	7								
divf	f13	f12	f10	6	14	22*									
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19*									



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 17: First->Second->Third Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14			
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	17
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16			
addf	f12	f6	f2	4	11	12	13	addi	r20	r20		17			
addi	r10	r10		5	6	6	7								
divf	f13	f12	f10	6	14	22									
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19*									



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 18: First->Second->Third Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14			
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	17
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16			
addf	f12	f6	f2	4	11	12	13	addi	r20	r20		17	18	18	
addi	r10	r10		5	6	6	7	subi	r1	r1		18			
divf	f13	f12	f10	6	14	22									
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19*									



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 19: First->Second->Third Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14			
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	17
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16			
addf	f12	f6	f2	4	11	12	13	addi	r20	r20		17	18	18	19
addi	r10	r10		5	6	6	7	subi	r1	r1		18	19	19	
divf	f13	f12	f10	6	14	22		bne	--	r1		19	--	--	--
addi	r20	r20		7	8	8	9								
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19									



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 20: First->Second->Third Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14			
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	17
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16			
addf	f12	f6	f2	4	11	12	13	addi	r20	r20		17	18	18	19
addi	r10	r10		5	6	6	7	subi	r1	r1		18	19	19	20
divf	f13	f12	f10	6	14	22		bne	--	r1		19	--	--	--
addi	r20	r20		7	8	8	9	stf	--	f13	r20	20			
subi	r1	r1		8	9	9	10								
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19	20								



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 21: First->Second->Third Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14	21	22	
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	17
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16			
addf	f12	f6	f2	4	11	12	13	addi	r20	r20		17	18	18	19
addi	r10	r10		5	6	6	7	subi	r1	r1		18	19	19	20
divf	f13	f12	f10	6	14	22		bne	--	r1		19	--	--	--
addi	r20	r20		7	8	8	9	stf	--	f13	r20	20			
subi	r1	r1		8	9	9	10	ldf	f20	r10		21			
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19	20								



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 22: First->Second->Third Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14	21	22	
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	17
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16			
addf	f12	f6	f2	4	11	12	13	addi	r20	r20		17	18	18	19
addi	r10	r10		5	6	6	7	subi	r1	r1		18	19	19	20
divf	f13	f12	f10	6	14	22		bne	--	r1		19	--	--	--
addi	r20	r20		7	8	8	9	stf	--	f13	r20	20			
subi	r1	r1		8	9	9	10	ldf	f20	r10		21	22	24*	
bne	--	r1		9	--	--	--								
stf	--	f13	r20	10											
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19	20								



CS 152 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 23: First->Second->Third Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14	21	22	23
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	17
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16			
addf	f12	f6	f2	4	11	12	13	addi	r20	r20		17	18	18	19
addi	r10	r10		5	6	6	7	subi	r1	r1		18	19	19	20
divf	f13	f12	f10	6	14	22	23	bne	--	r1		19	--	--	--
addi	r20	r20		7	8	8	9	stf	--	f13	r20	20			
subi	r1	r1		8	9	9	10	ldf	f20	r10		21	22	24*	
bne	--	r1		9	--	--	--	ldf	f10	r10		22	23	25*	
stf	--	f13	r20	10				multf	f6	f20	f1	23			
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19	20								



CS 162 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 24: First->Second->Third Divf**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	f20	r10		1	2	4	5	addf	f12	f6	f2	14	21	22	23
ldf	f10	r10		2	3	5	6	addi	r10	r10		15	16	16	17
multf	f6	f20	f1	3	6	9	10	divf	f13	f12	f10	16	24	32*	
addf	f12	f6	f2	4	11	12	13	addi	r20	r20		17	18	18	19
addi	r10	r10		5	6	6	7	subi	r1	r1		18	19	19	20
divf	f13	f12	f10	6	14	22	23	bne	--	r1		19	--	--	--
addi	r20	r20		7	8	8	9	stf	--	f13	r20	20			
subi	r1	r1		8	9	9	10	ldf	f20	r10		21	22	24	
bne	--	r1		9	--	--	--	ldf	f10	r10		22			
stf	--	f13	r20	10				multf	f6	f20	f1	23			
ldf	f20	r10		11	12	14	15								
ldf	f10	r10		12	13	15	16								
multf	f6	f20	f1	13	16	19	20								



CS 162 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 23: First->Second->Third Divf**

Divf1: Issued 6 Finished 23  
 Divf2: Issued 16 Finished 33  
 Divf3: Issued ?? Finished ??

We're Done!



CS 162 Review

Jack Kang and Kurt Meitz

### Question 3:

Tomasulo Trace: **CC 23: First->Second->Third Divf**

Divf1: Issued 6 Finished 23  
 Divf2: Issued 16 Finished 33  
 Divf3: Issued 26 Finished 43

We're Done!

The second divf issues before the first finished, so we will need at least 2 entries.

The first finishes before the third issues, so we will need at most 2 entries.

Therefore, we need 2 entries.



CS 162 Review

Jack Kang and Kurt Meitz

### Question 4

- TLB Page Cache table
- A. miss miss miss
  - B. miss miss hit
  - C. miss hit miss
  - D. miss hit hit
  - E. hit miss miss
  - F. hit miss hit
  - G. hit hit miss
  - H. hit hit hit



CS 162 Review

Jack Kang and Kurt Meitz

### Question 4

- TLB Page Cache Possible? If so, under what circumstance table
1. miss miss miss TLB misses and is followed by a page fault; after retry, data must miss in cache.
  2. miss miss hit Impossible: data cannot be allowed in cache if the page is not in memory.
  3. miss hit miss TLB misses, but entry found in page table; after retry, data misses in cache.
  4. miss hit hit TLB misses, but entry found in page table; after retry, data is found in cache.
  5. hit miss miss Impossible: cannot have a translation in TLB if page is not present in memory.
  6. hit miss hit Impossible: cannot have a translation in TLB if page is not present in memory.
  7. hit hit miss Possible, although the page table is never really checked if TLB hits.
  8. hit hit hit Possible, although the page table is never really checked if TLB hits.



CS 162 Review

Jack Kang and Kurt Meitz