CS152 – Exam 2 Review

2003-11-18

Jack and Kurt
www-inst.eecs.berkeley.edu/~cs152/

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:

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

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.

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

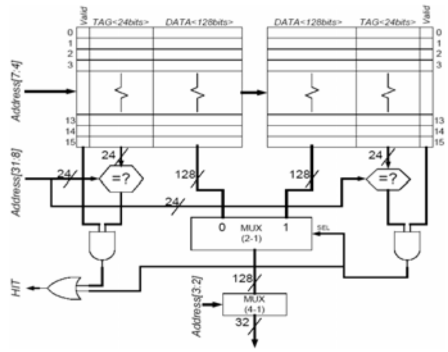
Problem 1b: How many sets does this cache have? Explain.

4 bits in the index field ➔
2^4 possible values ➔
16 sets

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.

Question 1c:



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.*

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*	—
0x2AE	*Miss*	*Compulsory*	0x1BA	*Hit*	—

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*	—
0x2AE	*Miss*	*Compulsory*	0x1BA	*Hit*	—

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

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*	—
0x2AE	*Miss*	*Compulsory*	0x1BA	*Hit*	—

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

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

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

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 + 20ns/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)?

AMATDisk = ?

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 + 20ns/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)?

AMATDisk = AccessTime + AMATMissPenalty + TransferRate*TransferSize
= 50E6ns + 0 + (20ns/byte * 16Kbytes)
= ~~ 5E7ns
= 5E7ns/ (2ns/clock) → 2.5E7 clocks

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 + 20ns/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)?

AMATDRAM = AccessTime + AMATMiss + TransferRate*TransferSize
= 100ns + 5E7ns*0.01 + (25ns/8bytes * 128bytes) =
= ~~ 5E5ns
= 5E5ns/ (2ns/clock) → 2.5E5 clocks

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 + 20ns/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)?

AMATL2 = AccessTime + AMATMiss + TransferRate*TransferSize
= (20c*2ns/c) + 5E5ns*0.02 + (2ns/8bytes * 64bytes) =
= ~~ 1E4 ns
= 1E4ns/ (2ns/clock) → 5E3 clocks

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 + 20ns/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)?

AMATL1Inst = AccessTime + AMATMiss + TransferRate*TransferSize
= (1c*2ns/c) + (5E3ns * 0.01) + 0
= ~~ 1E2 ns
= 1E2ns/ (2ns/clock) → 50 clocks (!!)

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 + 20ns/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)?

AMATL1Data = AccessTime + AMATMiss + TransferRate*TransferSize
= (1c*2ns/c) + (5E3ns * 0.04) + 0
= ~~ 4E2 ns
= 4E2ns/ (2ns/clock) → 200 clocks (!!)

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)?

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)?

CPI = MinCPI + Σ [CPI of exceptional events]

= MinCPI + CPIHazardStalls + CPIMemoryStalls

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)?

CPI = MinCPI + Σ [CPI of exceptional events]

= MinCPI + CPIHazardStalls + CPIMemoryStalls

= 1 + Σ(InstTypeFreq*CPI) + Σ(MemAccessFreq*AccessAMAT)

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)?

CPI = MinCPI + Σ [CPI of exceptional events]

= MinCPI + CPIHazardStalls + CPIMemoryStalls

= 1 + Σ(InstTypeFreq*CPI) + Σ(MemAccessFreq*AccessAMAT)

= 1 + [(FPFreq*FPCPI)
+ (BBranchFreq*BBCPI)]
+ [(MemInstFreq * AMATL1Inst/(2ns/clock))
+ (DataInstFreq *AMATL1Data/(2ns/clock))]

= 1 + 0.15*0.7 + 0.2*0.05*0.3 + 1*100/2 + 0.35*400/2 ➔ 124CPI (!!)

Question 2a:

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

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.

Question 2b

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

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.

Question 2c

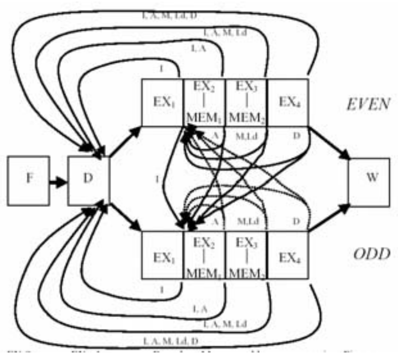
2c) Below is a ***start*** at a simple diagram for the pipelines of this processor.

1) Finish the diagram. Stages are boxes with letters inside: Use "F" for a fetch stage, "D" for a decode stage, EX_1 through EX_4 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 if 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 **multf**, "D" for **divf**, "A" for **addf**, "I" for integer operations, and "Ld" for **load results**). *[Hint: think carefully about inputs to store instructions!]*

Question 2c



Question 2c

EX Stages:
- EX_1: Integer ops, Branches, Memory address computation, First stage of A, M, D
- EX_2: First stage of load/store, Finish A, Second stage of M, D
- EX_3: Final stage of load/store, Finish M. Third stage of D
- EX_4: 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_2. 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.

Question 2d

2d) Note that we assume that a load is not completed until the end of EX_3 and that a store must have its value by the beginning of EX_2. 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 to be dispatched in the same cycle? What is the minimum number of *instructions* that must be placed between them to avoid stalling?

Question 2d

2d) Note that we assume that a load is not completed until the end of EX_3 and that a store must have its value by the beginning of EX_2. 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 to 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_4 stage of the odd pipeline while the store is in the EX_2 stage of the even pipeline. Look at the answer for the previous problem. There is a special store arc to handle this circumstances. The load is 2 cycles ahead of the store. We need to fill instructions in the two different EX_3 stages.

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.

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 0 instructions. What this means is that the two mem stages for the even pipeline are in EX_3 and EX_4. Then, if the load is in the odd pipeline and the store is in the even pipeline (next cycle), we have no stalls.

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.

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 2nd 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.

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    $F20, 0($r10)
       ldf    $F10, 8($r10)   <--
       multf  $F6, $F20, $F1
       addf   $F12, $F6, $F2  <--
       addi   $r10, $r10, #16
       divf   $F13, $F12, $F10
       addi   $r20, $r20, #8
       subi   $r1, $r1, #1
       bne    $r1, $zero, loop  <--
         stf  -8($r20), $F13
```


Question 3:

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

```
loop:  ldf    $F20, 0($r10)
       ldf    $F10, 8($r10)   <--
       multf  $F6, $F20, $F1
       addf   $F12, $F6, $F2  <--
       addi   $r10, $r10, #16
       divf   $F13, $F12, $F10
       addi   $r20, $r20, #8
       subi   $r1, $r1, #1
       bne    $r1, $zero, loop  <--
         stf  -8($r20), $F13
```

Keys to Problem:

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

2)

Question 3:

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

```
loop:  ldf    $F20, 0($r10)
       ldf    $F10, 8($r10)     <--
       multf  $F6, $F20, $F1
       addf   $F12, $F6, $F2    <--
       addi   $r10, $r10, #16
       divf   $F13, $F12, $F10
       addi   $r20, $r20, #8
       subi   $r1, $r1, #1
       bne    $r1, $zero, loop  <--
         stf  -8($r20), $F13
```

Keys to Problem:

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

2) We can trace through a few iterations of the loop to see how many divs are active at any given time

Question 3:

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

```
loop:  ldf    $F20, 0($r10)
       ldf    $F10, 8($r10)     <--
       multf  $F6, $F20, $F1
       addf   $F12, $F6, $F2    <--
       addi   $r10, $r10, #16
       divf   $F13, $F12, $F10
       addi   $r20, $r20, #8
       subi   $r1, $r1, #1
       bne    $r1, $zero, loop  <--
         stf  -8($r20), $F13
```

Keys to Problem:

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

2) We can trace through a few iterations of the loop to see how many divs are active at any given time

3) We need a table to trace the tomasulo!

Question 3:

Tomasulo Trace: **CC 1: First Few instructions**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	**F20**	**R10**		**1**											

Question 3:

Tomasulo Trace: **CC 2: First Few instructions**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	F20	R10		1	**2**	**4***									
Ldf	**F10**	**R10**		**2**											

Question 3:

Tomasulo Trace: **CC 3: First Few instructions**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	F20	R10		1	2	4*									
Ldf	F10	R10		2	**3**	**5***									
multf	**F6**	**F20**	**F1**	**3**											

Question 3:

Tomasulo Trace: **CC 4: First Few instructions**

N	rd	rs	rt	I	E1	EF	WB	N	rd	rs	rt	I	E1	EF	WB
ldf	F20	R10		1	2	**4**									
Ldf	F10	R10		2	3	5*									
multf	**F6**	F20	F1	3											
addf	**F12**	**F6**	**F2**	**4**											

Question 3:

Tomasulo Trace: CC 5: First Few instructions

N	rd	rs	rt	I	E1	EF	WB
ldf	F20	R10		1	2	4	5
Ldf	F10	R10		2	3	5	
multf	F6	F20	F1	3			
addf	F12	F6	F2	4			
addi	R10	R10		5			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 6: First Few instructions

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*	
addf	F12	F6	F2	4			
addi	R10	R10		5	6	6	
divf	F13	F12	F10	6			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 7: First->Second Divf

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*	
addf	F12	F6	F2	4			
addi	R10	R10		5	6	6	7
divf	F13	F12	F10	6			
addi	R20	R20		7			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 8: First->Second Divf

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*	
addf	F12	F6	F2	4			
addi	R10	R10		5	6	6	7
divf	F13	F12	F10	6			
addi	R20	R20		7	8	8	
Subi	R1	R1		8			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 9: First->Second Divf

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	
addf	F12	F6	F2	4			
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	
bne	--	R1		9			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 10: First->Second Divf

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			
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			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 11: First->Second Divf

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**			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 12: First->Second Divf

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**			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 13: First->Second Divf

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**			

N	rd	rs	rt	I	E1	EF	WB

Question 3:

Tomasulo Trace: CC 14: First->Second Divf

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	**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			

N	rd	rs	rt	I	E1	EF	WB
addf	**f12**	**f6**	**f2**	**14**			

Question 3:

Tomasulo Trace: CC 15: First->Second Divf

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	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			

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14			
addi	**r10**	**r10**		**15**			

Question 3:

Tomasulo Trace: CC 16: First->Second Divf

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	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*	

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14			
addi	r10	r10		15	**16**	**16**	
divf	**f13**	**f12**	**f10**	**16**			

Question 3:

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

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	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*	

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14			
addi	r10	r10		15	16	16	**17**
divf	f13	f12	f10	16			
addi	**r20**	**r20**		**17**			

Question 3:

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

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	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*	

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14			
addi	r10	r10		15	16	16	17
divf	f13	f12	f10	16			
addi	r20	r20		17	**18**	**18**	
subi	**r1**	**r1**		**18**			

Question 3:

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

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	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	

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14			
addi	r10	r10		15	16	16	17
divf	f13	f12	f10	16			
addi	r20	r20		17	18	18	**19**
subi	r1	r1		18	**19**	**19**	
bne	**--**	r1		**19**	**--**	**--**	**--**

Question 3:

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

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	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	**20**

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14			
addi	r10	r10		15	16	16	17
divf	f13	f12	f10	16			
addi	r20	r20		17	18	18	19
subi	r1	r1		18	19	19	**20**
bne	--	r1		19	--	--	--
stf	**--**	f13	**r20**	**20**			

Question 3:

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

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	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	20

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14	**21**	**22**	
addi	r10	r10		15	16	16	17
divf	f13	f12	f10	16			
addi	r20	r20		17	18	18	19
subi	r1	r1		18	19	19	20
bne	--	r1		19	--	--	--
stf	--	f13	r20	20			
ldf	**f20**	**r10**		**21**			

Question 3:

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

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	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	20

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14	21	22	
addi	r10	r10		15	16	16	17
divf	f13	f12	f10	16			
addi	r20	r20		17	18	18	19
subi	r1	r1		18	19	19	20
bne	--	r1		19	--	--	--
stf	--	f13	r20	20			
ldf	f20	r10		21	**22**	**24***	
ldf	**f10**	**r10**		**22**			

Question 3:

Tomasulo Trace:

CC 23: First->Second- >Third Divf

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	14	22	**23**
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	20

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14	21	22	**23**
addi	r10	r10		15	16	16	17
divf	f13	f12	f10	16			
addi	r20	r20		17	18	18	19
subi	r1	r1		18	19	19	20
bne	--	r1		19	--	--	--
stf	--	f13	r20	20			
ldf	f20	r10		21	22	24*	
ldf	f10	r10		22	23	25*	
multf	**f6**	**f20**	**f1**	**23**			

Question 3:

Tomasulo Trace:

CC 24: First->Second- >Third Divf

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	14	22	23
addi	r20	r20		7	8	8	9
subi	r1	r1		8	9	9	10
bne	--	r1		9	--	--	--
stf	--	f13	r20	10	**24**	**26***	
ldf	f20	r10		11	12	14	15
ldf	f10	r10		12	13	15	16
multf	f6	f20	f1	13	16	19	20

N	rd	rs	rt	I	E1	EF	WB
addf	f12	f6	f2	14	21	22	23
addi	r10	r10		15	16	16	17
divf	f13	f12	f10	16	24	32*	
addi	r20	r20		17	18	18	19
subi	r1	r1		18	19	19	20
bne	--	r1		19	--	--	--
stf	--	f13	r20	20			
ldf	f20	r10		21	22	24	
ldf	f10	r10		22			
multf	f6	f20	f1	23			
…							

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!

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.

Question 4

TLB	Page table	Cache
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

Question 4

TLB	Page table	Cache	Possible? If so, under what circumstance
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.