# Sample Final 1

## Problem 1. [10 Points] Tomasulo's Algorithm

This problem is about a processor that uses Tomasulo's algorithm (no ROB), with a total of 3 reservation stations. Any instruction can use any reservation station. The processor is 1-wide. Do not make any assumptions about the execution latencies of particular instructions. The current state of reservation stations is:

| RS | Busy? | Op | Vi | Vk | Qi | Qk |
|---|---|---|---|---|---|---|
| RS 0 | Yes | MUL | | | RS1 | RS2 |
| RS 1 | Yes | ADD | 3 | | | RS2 |
| RS 2 | Yes | SUB | 10 | 7 | | |

1. [2 points] What was the order in which these three instructions were issued?

You can tell that ADD has a RAW dependency on SUB, since the result from RS2 will be used as an operand for the instruction in RS1. Likewise, you can tell that MUL has a RAW dependency on ADD. Therefore, these must have been issued in this order: SUB, ADD, MUL.

2. [2 points] Let X be the number of instructions that are currently being executed. From the given current state of the reservation stations, what can we infer about X?

RS0 and RS1 are waiting for results from other instructions. RS2 MIGHT be executing, but it might not be. It's possible that RS2 had a dependency that was just now captured. Therefore, we can conclude that there is either 0 or 1 instruction executing.

3. [2 points] When the MUL instruction broadcasts its result on the CDB, what data value and tag (name) will it put on the CDB?

The result of SUB will be 3. That will supply the second operand to RS1, so the result of ADD will be 6. The operands for MUL will therefore be 3 and 6. Therefore, when MUL broadcasts, the tag will be RS0 and the result will be 18.

4. [2 points] When the ADD instruction broadcasts its result on the CDB, what data value and tag (name) will it put on the CDB?

The result of SUB will be 3. That will supply the second operand to RS1, so the result of ADD will be 6. Therefore, when ADD broadcasts, the tag will be RS1 and the result will be 6.

5. [1 points] True or False (circle the correct answer): We can say for sure that one of these three instructions was the last instruction issued by this processor thus far.

True. When an instruction is issued, it is put into an RS. (I believe the intent is to include the load/store queue in the unified reservation stations. To see an example of this, look at the "Long Example Introduction" video in the "Instruction Scheduling" lectures (https://www.youtube.com/watch?v=2M5NQFAaILk). Also see the textbook, Section 3.4 "Overcoming Data Hazards with Dynamic Scheduling". The idea behind unified reservation stations is that they reduce a potential bottleneck that occurs when the processor runs out of one type of reservation station. Since load/store queues can be represented in almost the same way, using unified reservation stations for the load/store queue also reduces a potential bottleneck.)

6. [1 points] True or False (circle the correct answer): It is possible that these are not the last three instructions issued by this processor thus far.

True.

## Problem 2. [10 points] Caches

The processor has a 32Kbyte, 32-way set associative, write-back, write-allocate, virtually indexed, physically-tagged L1 cache with 64-byte blocks. The system uses 32-bit virtual and physical addresses, with 4K byte pages.

1. [2 points] Can we overlap virtual-to-physical address translation with the L1 cache access? Explain you answer.

Yes. The number of block offset bits plus the number of index bits is less than the number of page offset bits. Therefore, the index is part of the page offset. Therefore, the index can be obtained from the virtual address, so overlap of address translation and cache access can occur.

2. [2 points] Each time the processor accesses the cache, what is the number of tag comparisons that are needed to determine if the access is a cache hit?

The cache is 32-way, so there are 32 tags which must be compared. Way prediction is not mentioned in the problem, so the assumption is that there is no way prediction.

3. [2 points] If the processor reads memory at virtual address 0x00001234 (physical address 0xFFFF1234), what is the index that is used to access the cache?

0x00001234 = 0000 0000 0000 0000 0001 0010 0011 0100 = 00000000000000000100/1000/110100, so the index is 1000 = 0x8.

4. [2 points] If the processor reads memory at virtual address 0x00001234 (physical address 0xFFFF1234), what is the tag that is used by the cache to determine if the access is a cache hit?

0xFFFF1234 = 1111 1111 1111 1111 0001 0010 0011 0100 = 11111111111111110100/1000/110100, so the tag is 11111111111111110100 = 11 1111 1111 1111 1100 0100 = 0x3FFFC4.

5. [2 points] What would be the main argument in favor of changing this cache to make it virtually-tagged?

The primary argument is that the elapsed time to hit in a VIVT cache might be less than the amount of time to hit in a VIPT cache, since no address translation is necessary. Whether this is actually true depends on the relative times that it takes for the two paths in a VIPT - selecting a set and getting tags from the cache to the comparators vs. doing address translation and getting the tag from the address to the comparators. If the latter takes more time, than we could get data from a VIVT cache faster.

But does that matter? It's possible that getting the data from the cache in some fraction of a cycle faster would not make any difference. But it's also possible that it would be just enough faster to allow cache accesses to be done in the same cycle that the data is needed by the processor, instead of taking 2 cycles to do both.

Another argument that could be made is that a VIVT allows the processor to not access the TLB at all when there is a cache hit, which can save energy. However, this isn't really true for most processors because the TLB also contains information that determines whether the code is allowed to access that memory page.

## Problem 3. [10 points] Cache Coherence

A shared-memory bus-based multiprocessor has two processors. Each processor has a direct- mapped L1 cache, with only sixteen 256-byte blocks in each cache. All addresses in this problem are physical addresses, and all caches are physically indexed and tagged. The MESI coherence protocol is used in this multiprocessor. The current state of the two caches is: The next access is on P0, which executes SB (store byte) to address 0xFFFFFCFF.

| | P0 State | P0 Tag | P1 State | P1 Tag |
|---|---|---|---|---|
| 0 | I | 0xFFFF0 | M | 0xFFFF0 |
| 1 | E | 0xFFFF0 | E | 0xFFFFF |
| 2 | S | 0xFFFF0 | S | 0xFFFFF |
| 3 | S | 0xFFFFF | S | 0xFFFFF |
| 4 | E | 0xFFFFF | I | 0xFFFFF |
| 5 | M | 0xFFFF0 | M | 0xFFFF1 |
| 6 | I | 0xFFFFF | M | 0xFFFFF |
| 7 | S | 0xFFFFF | S | 0xFFFFF |
| 8 | E | 0xFFFF1 | S | 0xFFFFF |
| 9 | I | 0xFFFF1 | S | 0xFFFFF |
| A | I | 0xFFFF1 | S | 0xFFFF1 |
| B | I | 0xFFFF1 | E | 0xFFFF1 |
| C | S | 0xFFFFF | I | 0xFFFFF |
| D | E | 0xFFFFF | I | 0xFFFF1 |
| E | M | 0xFFFFF | I | 0xFFFFF |
| F | I | 0xFFFFF | I | 0xFFFF1 |

1. [1 point] Does this access result in a bus broadcast? (Yes or No)

```
Yes.
0xFFFFFCFF = 0xFFFFF/0xC/0xFF, so index is 0xC. P0 has same tag in line 0xC. State is S,
which is not M or E, so broadcast is necessary to invalidate any line holding that block in P1.
```

2. [1 point] What is the new content of row C in P0's and P1's caches?

```
P0 state = M. P0 tag = 0xFFFFF. P1 state = I. P1 tag = 0xFFFFF.
There is no change to P1 because it does not have that memory block. Having a line with the relevant
tag, but in the Invalid state, doesn't really count as "having" the block.
```

3. [1 point] The next access is on P1, which executes SB (store byte) to address 0xFFFFF5FF. Does this access result in a bus broadcast? (Yes or No)

```
Yes.
0xFFFFF5FF = 0xFFFFF/0x5/0xFF, so index is 0x5. A bus broadcast is required to invalidate
any line in P0 that contains the memory block. Also, the block that is currently in line
5 of P1 must be ejected, and since the state is M, that requires a write to memory (assuming
that the cache is write-back).
```

4. [1 point] What is the new content of row 5 in P0's and P1's caches?

```
P0 state = M. P0 tag = 0xFFFF0. P1 state = M. P1 tag = 0xFFFFF.
There is no change to P0 because it does not have that memory block. In P1, line 5 will
have tag 0xFFFFF and state M.
```

5. [1 point] The next access is on P1, which executes LB (load byte) to address 0xFFFFF3FF. Does this access result in a bus broadcast? (Yes or No)

```
No.
0xFFFFF3FF = 0xFFFFF/0x3/0xFF, so index is 0x3. P1 already has the memory block in line 3,
so no broadcast is necessary.
```

6. [1 point] What is the new content of row 3 in P0's and P1's caches?

```
P0 state = S. P0 tag = 0xFFFFF. P1 state = S. P1 tag = 0xFFFFF.
There is no change to P0 since no broadcast occurs. There is no change to P1 since it
already has the memory block with state S.
```

7. [1 point] The next access is on P0, which executes SB (store byte) to address 0xFFFFFD00. Does this access result in a bus broadcast? (Yes or No)

```
No.
0xFFFFFD00 = 0xFFFFF/0xD/0x00, so index is 0xD. P0 already has the memory block with state E,
so it can change the state to M and write to the block without any bus activity.
```

8. [1 point] What is the new content of row D in P0's and P1's caches?

```
P0 state = M. P0 tag = 0xFFFFF. P1 state = I. P1 tag = 0xFFFF1.
Since there is no bus activity, nothing changes on P1.
```

9. [1 point] The next access is on P0, which executes SB (store byte) to address 0xFFFFF700. Does this access result in a bus broadcast? (Yes or No)

```
Yes.
0xFFFFF700 = 0xFFFFF/0x7/0x00, so index is 0x7. State is S, which is not M or E, so broadcast
is necessary to invalidate any line holding that block in P1.
```

10. [1 point] What is the new content of row 7 in P0's and P1's caches?

```
P0 state = M. P0 tag = 0xFFFFF. P1 state = I. P1 tag = 0xFFFFF.
P1 invalidates its line since it held the memory block.
```

## Problem 4. [10 points] Parallel Processing

A shared-memory system with two processors is executing the following two code fragments in parallel (one on each processor). All variables are shared and are initially zeros. The system uses sequential consistency.

```
//Processor P0
A=10;
Atmp=A;
B=Atmp-2;
while(C==0);
C=6;
printf("%d %d %d\n",A,B,C);

// Processor P1
Btmp=B;
C=Btmp;
if(C==0) {
    C=3;
}
A=3;
```

1. [2 points] Is it possible for the printout in P0 to be â€œ10 8 6â€•. If yes, show the execution interleaving that produces this printout. If no, explain why not.

```
P0: A=10;               // 10
P0: Atmp=A;             // 10
P0: B=Atmp-2;           // 8
P1: Btmp=B;             // 8
P1: C=Btmp;             // 8
P0: Find that C != 0.   // =8
P0: C=6;                // 6
P0: printf(A,B,C);      // 10, 8, 6
P1: Find that C!=0.
P1: A=3;
```

2. [2 points] Is it possible for P0 to get stuck and never print out anything? If yes, show the execution interleaving that produces this situation. If no, explain why not.

```
Neither P0 nor P1 will ever assign the value 0 to C (although C is initialized to 0). Thus,
if a non-zero value is ever assigned to C, P0 cannot stay stuck because nothing would change
it back to 0 after that. If the first assignment to C in P1 assigns a non-zero value, then
P0 cannot stay stuck. If the first assignment to C in P1 assigns 0 to C, then the "if" statement
will assign 6 to C, so P1 cannot stay stuck.
```

3. [2 points] Is it possible for P0 to print 3 as the value of C? If yes, show the execution interleaving that produces this printout. If no, explain why not.

```
Is it possible for P1 to assign 3 to C in the second statement? For that to occur, Btmp must be 3,
and therefore B must have the value 3 in the first statement in P1. This can only occur if Atmp is
5 in the third statement in P0, and that can only occur if A is 5 in the second statement in P0. A
can only ever have a value in the set {0, 10, 3}. Therefore, it's not possible for P1 to assign
3 to C in the second statement.

Is it possible for P1 to assign 3 to C in the fourth statement, and for C to still have that value
when the printf is reached? For that to occur, the second statement in P1 must assign 0 to C so
that the "if" statement will find the conditional to be true, but if that occurs, P0 cannot get
past the "while" loop until P1 reaches its fourth statement. But P0 will assign 6 to C after
it gets past the "while" loop and before it gets to the printf. So, there's no sequence that works.
```

4. [2 points] Is it possible for P0 to print a value of C other than 3 or 6? If yes, show the execution interleaving that produces this printout. If no, explain why not.

```
For this to happen, these things have to happen in this order:
1) Something assigns a non-zero value to C.
2) P0: Find that C != 0.
3) P0: C=6;
4) Something assigns a value to C which is not 3 or 6.
5) printf(A,B,C);

The assignment of Btmp to C in P1 is the only statement that could do step 4. But if it does Step 4,
there are no statements that could do Step 1, since the only other assignments to C are after the "whil
in P0, and after the assignment of Btmp to C in P1.
```

5. [2 points] Is it possible for P0 to print a value of B other than 8? If yes, show the execution interleaving that produces this printout. If no, explain why not.

```
P0: A=10;               // 10
P1: Btmp=B;             // 0
P1: C=Btmp;             // 0
P1: Find that C=0.      // 0
P1: C=3;                // 3
P1: A=3;                // 3
P0: Atmp=A;             // 3
P0: B=Atmp-2;           // 1
P0: Find that C != 0.   // 3
P0: C=6;                // 6
P0: printf(A,B,C);      // 3, 1, 6
```

## Problem 5. [10 points] Disks and RAID

A 6,000RPM disk drive has 5 platters, with 100,000 tracks per surface, 1,000 sectors per tracks, and 512 data bytes per sector. Each sector also has a 128-bit error detection code that can detect all errors in its sector but cannot correct any errors. The head can takes one microsecond (one millionth of a second) to move from

one cylinder to an adjacent cylinder, and multi-cylinder movements are done at the same speed (one cylinder per microsecond). The disk controller is very fast (assume zero latency for everything it does) and the I/O bus has very large (assume infinite) bandwidth.

time_per_rotation = 1 / ((6000 rev/minute) * (1 min / 60 sec)) = 1 / (100 rev/sec) = 0.01 seconds

1. [1 point] How many heads does the disk drive have?

```
Each platter has 2 surfaces, and there is a head for each surface.
2 * 5 = 10
```

2. [2 points] Assuming that the disk controller and the drive itself are not servicing any other requests, what is the worst-case time needed to read a sector from the disk?

```
0.110009 seconds
The worst case occurs when the sector to be read is at one extreme, and the head is at the other
extreme, and when the head gets to the correct track the start of the sector has just passed. It
takes 99,999 us = 0.099999 sec for the head to seek the right track. The disk has to rotate 1001/1000
revolutions, which takes 0.010010 sec. The sum is 0.110009 sec. (Here is an explanation of why the disk
has to rotate through 1001 sectors to read the right sector, in the worst case. There are two steps
necessary to read a sector after the head is on the right track. The first step is to wait until the dr
has rotated so that the head is at the beginning of the desired track. The next step is to read data fr
the desired track as the head rotates 1/1000 revolutions. The first step can require 1000/1000=1 revolu
of the disk, or at least arbitrarily close to that amount. In other words, when the head arrives at the
correct track, we might have just barely missed the start of the desired track, so we have to wait very
very close to one revolution to get back to the start. Note that it is not possible, as far as I know, i
a drive to read the second half of a sector as it's going by, and then when the disk has rotated almost
one revolution, read the first half of the sector.)
```

3. [2 points] Assuming that the disk controller and the drive itself are not servicing any other requests, what is the best-case time needed to read the entire disk?

```
1000.099999 seconds
The best case occurs if the head is at one extreme, allowing for no back-tracking of the head when movi
between cylinders. The best case also assumes that every time the head moves to a new track, it just ha
to be just in time to start reading the sector that the drive wants to read from that track. It also as
that all heads can read simultaneously, which I'm confident is true for all modern hard drives. It will
99,999 us = 0.099999 sec to seek across all cylinders. Since there are 100,000 tracks per surface, it w
take 100,000 revolutions to read all of the data, which requires 100000 * 0.01 sec = 1000 sec.
The total is 1000.099999 sec.
```

4. [1 points] If we use two of these disk drives in a RAID0 configuration, what is the total data capacity of the resulting RAID array?

```
1024 * 10^9 bytes
Because there is no redundancy of any kind (including parity) implemented by a RAID0 array,
the capacity of the array is the sum of the capacity of each of the disks.
```

5. [1 points] If we use two of these disk drives in a RAID0 configuration, can we recover all data if one sector is damaged on one of the disk drives? Explain your answer.

```
Because there is no redundancy of any kind, we can't recover data.
```

6. [1 points] If we use two of these disk drives in a RAID1 configuration, can we recover all data if one disk drive is accidentally dropped into an active volcano? Explain your answer.

```
Because each individual drive contains all of the data, we can recover all of the data from a single dr
```

7. [1 points] If we use five of these disk drives in a RAID5 configuration, what is the total data capacity of the resulting RAID array?

2048 * 10^9 bytes
In RAID5, the amount of parity data is the capacity of a single drive. So, the capacity of the array is the sum of the capacities of all but one of the drives.

8. [1 points] If we use five of these disk drives in a RAID5 configuration, is it possible to damage only two sectors in a way that the RAID array cannot recover from? Explain your answer.

Suppose that only a single bit is lost, which means that the bit is flipped to the opposite value. The error detection code in the sector would detect that there was an error in the sector, but that doesn't necessarily imply that the bit position would be known. Each bit position spanning all 5 drives is a parity group. Examining the parity group for each bit in the sector would allow us to identify which bit was damaged.

What if two bits in the same sector on a single drive are lost? We could recover each of them in the same way that we recovered a single bit.

What if the same bit (same bit position on the same sector position) was lost on two drives? That would be unrecoverable. The error detection code could tell us which drives have errors, but the parity scheme would not allow us to determine which bit position is damaged, since parity cannot detect two errors.