EE457: Computer Systems Organization
Practice Final Questions

Solutions

1. **(15 pts. Short Answer)** [Fill in the blanks or select the correct answer]

   1.1. A 4-way set associative cache with 8 sets will require ____ (**1 / 4 / 8 / 32**) TAG RAMs.

   1.2. ____ (**Temporal / Spatial**) locality is why we choose Least Recently Used block replacement.

   1.3. Precise exceptions means exception handling should appear equivalent to ___ (**single-cycle / multi-cycle / in-order pipelined**) processors.

   1.4. ____ **True/False**: WAR and WAW hazards are NOT true dependencies.

   1.5. ___ (**Static / Dynamic**) scheduling refers to the compiler having responsibility to reorder instructions to obtain parallelism.

   1.6. Indicate which of the following items are supported by inclusion of the ROB: ____ (**precise exceptions / memory disambiguation**).

   1.7. Direct mapped caches are equivalent to ___ (**1 / n**) way set-associative caches.

   1.8. Which of the following is NOT a reason the ROB is required to support speculative execution where branches are predicted and dispatch of instructions continues.
      a. The need to buffer results but not write them back until confirmed that the instruction should execute
      b. The need to selectively flush speculatively executed instructions
      c. **The need to select the appropriate data or tag for source operands**

   1.9. ___ **True/False**: Simultaneous multithreading refers to the process of executing instructions from different threads on separate processor cores at the same time.

2. **(15 pts.) Tomasulo's Algorithm**. Given the code below, show their execution on an OoO processor using Tomasulo's algorithm. Assume the first 'lw' instruction causes a cache miss and will not produce $5 for several hundred clock cycles. The 'sw' will hit in cache.

   2.1. Show the state of the RST (Register Status Table) after each instruction issues. We have provided "tags" T1-T6 for each instruction; assume these are the tags assigned to each instruction, should you need them. Assume **no** instruction **finishes** *before they all issue*.

   (Place a '-' in any RST entry, if the latest value is actually in the Register File.)

   2.2. In addition show the instructions as they wait in execution queues. For each waiting instruction, show the tag, instruction name, and replace its source operands with either a TAG or REGISTER VALUE.

   2.3. Draw a line (like this ~~cross out~~) through instructions that will be able to execute and leave the queues before the 'lw' cache miss is completed.

   2.4. Also show the contents of the RegFile just before the 'lw' restarts execution

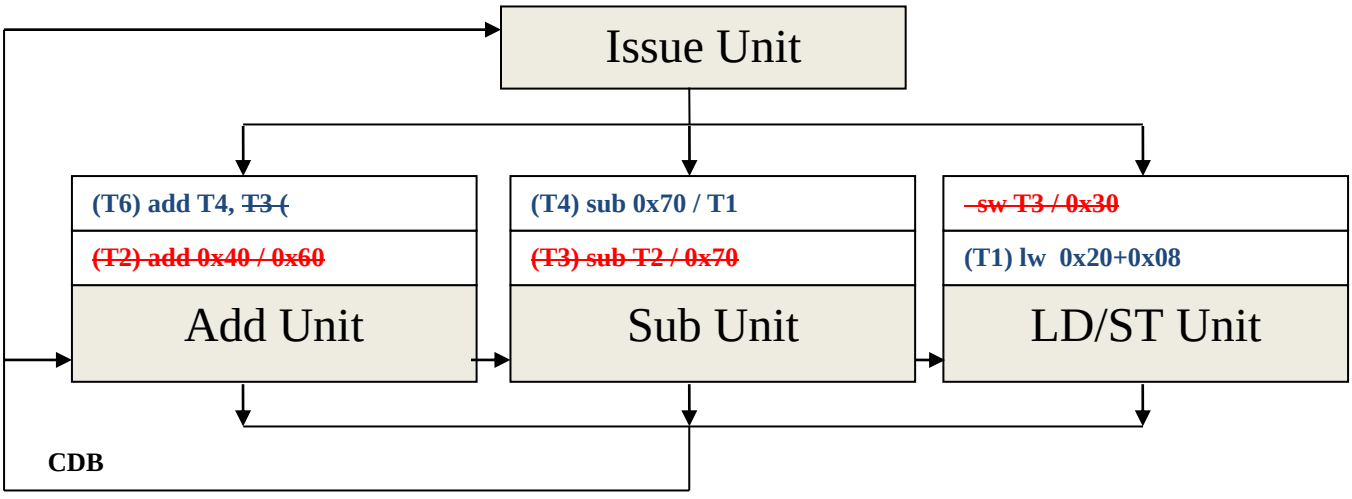
Code	Register Status Table (After Issue. Of Each Instruction)					
	$2	$3	$4	$5	$6	$7
(T1) lw $5,8($2)	-	-	-	T1	-	-
(T2) add $2,$4,$6	T2	-	-	T1	-	-
(T3) sub $6,$2,$7	T2	-	-	T1	T3	-
(T4) sub $7,$7,$5	T2	-	-	T1	T3	T4
(T5) sw $6,0($3)	T2	-	-	T1	T3	T4
(T6) add $3,$7,$6	T2	T6	-	T1	T3	T4

Initial RegFile

$2	0x20
$3	0x30
$4	0x40
$5	0x50
$6	0x60
$7	0x70

RegFile just before 'lw' restarts execution

$2	0xa0
$3	0x30
$4	0x40
$5	0x50
$6	0x30
$7	0x70



3. **(5 pts. Out-of-Order Execution):** In the following code, assume that the first **lw** instruction stalls due to a cache miss and that the miss latency is longer than the execution time of the trace. Assuming an out-of-order, dynamically-scheduled processor (with an ROB and automatic register renaming), which instructions would be allowed to execute (i.e., are independent) and which instructions would need to stall due to the cache miss?

```
      lw  $2, 0($16)  Cache Miss
(4.1) add $4,$4,$3       __ Stall    __ Execute
(4.2) sub $4,$4,$2       __ Stall    __ Execute
(4.3) add $5,$5,$4       __ Stall    __ Execute
(4.4) sw  $7,0($16)      __ Stall    __ Execute
(4.5) sw  $5,0($17)      __ Stall    __ Execute
```

4. **(15 pts. - MESI Cache Coherence Protocol :** Examine the table below showing sequence of memory accesses performed by three processors in a shared memory multiprocessor. Assume all caches are empty initially. Assume cache blocks of a single word (so we only show 1 data value). Show what bus requests and responses are initiated and the state of the block after each operation as well as what data is in memory.

(Bus Requests/ACtions: **BusRd**=Read, **BusRdX**=Read w/ Intent to Write, **BusUpgr** = Invalidate5others, **Flush** = Supply data on bus)

Memory Access	Bus Request / Response	P1 Cache State {M,E,S,I}	P2 Cache State {M,E,S,I}	P3 Cache State {M,E,S,I}	Mem. Data @ address X
P1 Read X	P1: BusRd	E	I	I	2
P3 Write X=3	P3: BusRdX	I	I	M	
P1 Read X	P1: BusRd P3: Flush	S	I	S	3
P1 Write X=4	P1: BusUpgr	M	I	I	
P1 EVICTS block X due to another read	P1: Flush	I	I	I	4
P2 Reads X	P2: BusRd	I	E	I	
P2 Writes X=5	-	I	M	I	

5. **(15 pts.) Caching.** Suppose you are given a system with 16-bit address and the following bits are used to access a **2-way Set Associative** cache.

Address Bits	15-8	7-6	5-2	1-0
Field	Tag	Set	Word offset	Byte enables

5.1. How many bytes total is the cache data memory? _**2-way*4 sets*64 bytes=512 bytes**___ bytes

5.2. How large is the tag RAM (rows x columns): __**4**_____ x __**9**___ (include V bit, but not Dirty bit)

5.3. Examine the following address trace/sequence and indicate which set the address maps to, whether the access will result in a hit or miss, and if a block will be evicted by an access. Assume Least Recently Used eviction policy.

Hex Address	**Equivalent Binary Address**	**Set #**	**Hit/Miss (Circle your answer)**	**Causes Eviction (Circle if Yes)**
0x1002	0001 0000 0000 0010	**0**	**Hit / Miss**	**Yes**
0x108c	0001 0000 1000 1100	**2**	**Hit / Miss**	**Yes**
0x311a	0011 0001 0001 1010	**0**	**Hit / Miss**	**Yes**
0x102c	0001 0000 0010 1100	**0**	**Hit / Miss**	**Yes**
0x10b0	0001 0000 1011 0000	**2**	**Hit / Miss**	**Yes**
0x2530	0010 0101 0011 0000	**0**	**Hit / Miss**	**Yes**
0x2518	0010 0101 0001 1000	**0**	**Hit / Miss**	**Yes**
0x311a	0011 0001 0001 1010	**0**	**Hit / Miss**	**Yes**

6. **(15 pts.) Memory Interleaving and Caching Implementation:** A legacy processor has a 20-bit (1 MB) address space and 16-bit data bus (i.e. a "word" unit = 2-bytes). The cache for this processor is 4 KB, uses a 2-way set associative mapping scheme, and uses blocks of size 4 bytes.

   6.1. How many total **cache** blocks are there? __**1K = 2^{10}**___

   6.2. Break the address bits into its tag, set, word, byte enable fields.

   6.3. Complete the diagram below filling in all the blanks below, including: the address bits and byte enables actually come out of the CPU core, the address bits that should be connected to the main memory banks, transceiver enable, cache data RAM's and cache tag RAM's

A[_19__:_11__]	**A[_10__:__2_]**	**A[_1__:_1_]**	**Byte Enables**
Tag	Set	Word	_____