EECS 470 F21 Final Exam Answer Key

Answers in red. Notes/Explanations in blue.

1. Pick and hope [12 points, -2 per wrong/blank, minimum 0]

Circle the best answer.

1) In the MESI protocol taught in class, which of the following transitions might happen due to a transaction by another processor?

M → E **S → E** **I → S** **M → S** **S → M**

The M → S transition occurs in response to another processor requesting data in the shared state (BusRd).

2) You would expect a 1024-byte direct-mapped cache with a block size of 64 bytes to get about what hit rate on a memory access with a stack distance of 2?

0% **80%** **88%** **92%** **96%** **100%**

There are (1024 bytes) / (64 bytes per cache block) = 16 cache blocks.

Stack distance refers to the number of unique memory references made between two successive accesses to the same memory location.

The access pattern (A, B, C, A) has a stack distance of two. For arbitrary A to be evicted and cause a miss on the subsequent access of A, either B or C must alias to the same cache block as A. The probability that the subsequent A is a miss: P(B evicts A) + P(C evicts A) = 1/16 + 1/16 = 1/8. Therefore, the hit rate is 1 – 1/8 = 88%.

3) In the R10K scheme, if you have a ROB size of 16, RS size of 8, and ARF size of 32, what is the maximum number of PRF entries you would expect to have?

40 **48** **50** **56** **70** **80**

Due to our ROB size, we can have a maximum of 16 instructions in-flight, each requiring a destination physical register. Due to the ARF size, we need 32 physical registers to maintain the current architected state. So, we need a total of 16 + 32 = 48 physical registers.

4) With what type of cache type can you be certain you won't need dirty bits?

write-back **write-allocate** **2-way assoc** **write-through** **skew**

5) Say you have a 4KB, 2-way associative write-back cache with 32-byte blocks and a 40-bitaddress space. How many bits would you need to index this cache?

14 **12** **10** **8** **6** **4**

Set size = 2 blocks per set * 32 bytes per block = 64 bytes
Number of sets = cache size / set size = 4 KB / 64 bytes = 64 sets
Set index bits = log2(# of sets) = log2(64) = 6 bits

6) Say you have a pipelined multiplier which is on the critical path of your processor. If you increase the number of pipeline stages you would expect which of the following?

- **The clock period and the CPI would go up.**
- **The clock period would go up and the CPI would go down.**
- **The clock period would go down and the CPI would go up.**
- **The clock period and the CPI would go down.**

7) In the P6 scheme taught in class, the RAT points to a _____ entry.

PRF **ARF** **RS** **ROB** **CDB**

8) Say you had an ISA where every instruction was predicated where any GPR can be used as a predicate. How many *more* instruction encodings would be needed for an add instruction (Rd=Ra+Rb) than in a non-predicated ISA? Assume there are 32 GPRs in both ISAs.

$2^{20}-2^{15}$ **2^{15}** **2^{20}** **$2^{20}+2^{15}$** **2^{5}** **2^{35}**

The total number of encodings for the non-predicated add instruction is 32 (choice of Rd) * 32 (choice of Ra) * 32 (choice of Rb) = 2^{15} encodings.

The total number of encodings for the predicated add instruction is 32 (choice of Rd) * 32 (choice of Ra) * 32 (choice of Rb) * 32 (choice of predicate GPR) = 2^{20} encodings.

Therefore, predication adds $2^{20}-2^{15}$ more encodings.

2. It's looking MESI out there [15 points, -0.5 per wrong box, minimum 0]

Consider a case of having 2 processors using a snoopy MESI protocol where the memories can snarf data. Both have a 2-line direct-mapped cache with ***each line consisting of 16 bytes***. The caches begin with all lines marked as invalid. Fill in the following tables indicating:

- If the processor gets a hit or a miss in its cache.
- What bus transaction(s) (if any) the processor performs (BRL, BWL, BRIL, BIL)
- If a HIT or HITM (or nothing) occurs on the bus during snoop.
- For misses only, indicate if the miss is compulsory, capacity, conflict, or coherence. A coherence miss is one where there would have been a hit, had some other processor not interfered.

Processor	Address	Read/Write	Cache Hit/Miss	Bus Transaction(s)	HIT/HITM	"4C" miss type (if any)
1	0x00	Write	Miss	BRIL	-	Compulsory
1	0x14	Read	Miss	BRL	-	Compulsory
2	0x10	Read	Miss	BRL	HIT	Compulsory
1	0x06	Write	Hit	-	-	-
2	0x19	Write	Miss	BIL	HIT	Coherence
1	0x10	Write	Miss	BRIL	HITM	Coherence
1	0x50	Write	Miss	BRIL/BWL	-	Compulsory
1	0x30	Read	Miss	BRL/BWL	-	Compulsory
2	0x00	Read	Miss	BRL	HITM	Compulsory
1	0x50	Read	Miss	BRL	-	Conflict

Final state:

Proc 1:

	Address	State
Set 0	0x00	S
Set 1	0x50	E

Proc 2:

	Address	State
Set 0	0x00	S
Set 1		I

3. Caching in [8 points, -2 per wrong/blank box, minimum 0]

Consider the following C-code segment:

```
char A[4096];
for (j = 0; j < 10000; j++)
    for (i = 0; i < Y; i = i + X)
        A[i] = A[i] + 1;
```

Assume that only accesses to the array A go to the data cache (the other values are in registers). For this code, what would be the expected *hit-rate* for the various values of X and Y if the data cache were 1 KB with 32-byte lines that was direct-mapped?

	X = 2	X = 4	X = 64
Y = 2048	31/32	15/16	1/2
Y = 1025	512/513	X	16/17

Students often struggle with the Y = 1025 row. For X = 2 and Y = 1025, the access pattern is: i = 0, 2, 4, 8, ..., 1022, 1024. Since 1024 is included, the inner for-loop executes 513 times. During these 513 iterations, there are 513 loads and 513 stores to array A.

In the cache access for address 1024, the block containing address 0 is evicted since both address 0 and 1024 alias to set 0. Then, when address 0 is loaded again in the next execution of the inner for-loop, it will be a miss due to the prior eviction, and it will evict the block containing address 1024. Consequently, the subsequent access to address 1024 will also be a miss.

Due to the back-and-forth evictions between addresses 0 and 1024, two loads will always miss (i.e. the load for 0 and load for 1024). With 1026 total cache accesses, the hit rate is 1024/1026 = 512/513.

If you did not separate loads and stores and treated each iteration of the inner for-loop as a single access to the data cache, you should have got:

	X = 2	X = 4	X = 64
Y = 2048	15/16	7/8	0
Y = 1025	511/513	X	15/17

4. ISA vs Microarchitecture [6 points, -1.5 per wrong/blank answer, minimum 0]

Indicate whether each of the following design choices in a processor is typically a feature of the ISA or of the microarchitecture. For each write either “ISA” or “MA”.

MA A 64-bit wide data bus to memory

MA Gshare branch prediction

ISA Predicated instruction execution

ISA A floating-point unit that uses wide floating-point values for additional accuracy

ISA An additional set of user-visible registers

ISA 32-bit instructions

MA 64KB L1 cache

MA 5-stage pipeline

5. Copypasta Processors [14 points]

HobTech, a small startup in Ann Arbor, has asked you to design a multicore system with different processor sizes (asymmetric multicore). Your design is to consist of one large processor with the rest of the die are being taken up by small cores.

You have three cores available for use. These numbers include caches and all uncore components needed for the cores to function correctly.

Core	Area	Performance	Dynamic Power	Static Power
A	20 mm^2	7 GIPS	50 Watts	10 Watts
B	10 mm^2	4 GIPS	25 Watts	5 Watts
C	2 mm^2	1 GIPS	1 Watt	1 Watt

<u>You have a total of 30 mm^2 area on your chip.</u>

a) Say HobTech wants to optimize performance for a program that is 90% perfectly parallel and 10% perfectly serial. How should you allocate your selection of cores? That is, how many of each would result in the best performance? To be clear, when running the serial part, nothing else can run at the same time and when running the parallel part, all processors can be utilized. Justify your answer **[6]**

Consider a program that has 9 GI in parallel and 1 GI in series. Under the design constraints, there are 3 combinations of the cores. Finding the execution time of the program on each combination:

- A & 5C: (1 GI / 7 GIPS) + (9 GI / 12 GIPS) = 0.89 s
- B & 10C: (1 GI / 4 GIPS) + (9 GI / 14 GIPS) = 0.89 s
- A & B: (1 GI / 7 GIPS) + (9 GI / 11 GIPS) = 0.96 s

There is a tie between A & 5C and B & 10C

b) Say HobTech has limited you two configurations:
   - **Configuration X:** 1 A core and the rest C cores.
   - **Configuration Y:** 1 B core and the rest C cores.

   If the workload we run has a fraction S of the work be perfectly serial and (1-S) be perfectly parallel, for what values of S will Configuration X use less energy to accomplish the same work? Show your work. You should assume that when a core is unused we can't shut it down, but we can insure no transistors are switching. Clearly state and justify any assumptions you are making. **[8]**

   Say we have 1 GI instruction total.

   We want to minimize J / GI, or equivalently watts / GIPS. During serial execution, total power will be the static power of all the C cores plus the static and dynamic

power of core A or B (depending on the configuration). During parallel execution, we can choose to shut off as many cores as desired.

Configuration X

Serial: 65 W / 7 GIPS

Parallel: 70 W / 12 GIPS (all cores on) or 20 W / 5 GIPS (A off) → 20 W / 5 GIPS is smaller

Total energy: S * (65 W / 7 GIPS) + (1 – S) * (20 W / 5 GIPS) = 5.29 * S + 4

Configuration Y

Serial: 40 W / 4 GIPS

Parallel: 50 W / 14 GIPS (all cores on) or 25 W / 10 GIPS (B off) → 25 W / 10 GIPS is smaller

Total energy: S * (40 W / 4 GIPS) + (1 – S) * (25 W / 10 GIPS) = 7.5 * S + 2.5

Solving for S

5.29 * S + 4 = 7.5 * S + 2.5

S = 0.677

Configuration X uses less energy than Y for S > 0.677.

Your answer does not need to be this verbose.

6. Short answer – numbers and letters [12 points]

a) Given a 40-bit virtual memory system with a 32-bit physical memory and a physically-addressed L1 cache that is 64KB, where each line has a 25-bit tag, what is the minimum degree of associativity of the cache? **[4]**

512-way associative

The problem statement does not specify block size, but that is not needed to solve it. Given that memory addresses are 32 bits, and the tag is 25 bits, we know that the set index bits + block offset bits = 32 – 25 = 7 bits. This tells us that each way of associativity occupies 2^7 bytes. Dividing total cache size by the size of each way, we can solve for associativity: 64 KB / (2^7 bytes per way) = 2^9 ways.

b) Say your out-of-order processor has 32 RoB entries, 8 RS entries, and the ISA supports 16 architected registers. How many bits would you need for the RRAT (just the pointers in the table, not valid bits, the free list or anything else)?

i. If you are using the P6 scheme taught in class? **[2]**
log2(32) = 5 bits 5 bits * 16 registers = 80 bits

In the P6 scheme, the RRAT points to ROB entries

ii. If you are using the R10K scheme taught in class? **[2]**

log2(48) = 6 bits 6 bits * 16 registers = 96 bits

In the R10K scheme, the RRAT points to PRF entries. See problem 1.3 for why there are 48 entries in the PRF.

c) Say you have an unified LSQ with non-speculative load-to-store forwarding. Each instructions reads/writes 4 bytes. Which of the following loads are able to issue (either from a store forward or to the memory)? **[4]**

Slot	**LSQ Address**
A (head)	Store 0x20
B	**Load 0x40**
C	**Load 0x20**
D	**Load ???**
E	Store ???
F	Store 0x34
G	**Load ???**
H	**Load 0x50**
I (tail)	**Load 0x34**

B, **C**, and **I** can issue

For a non-speculative LSQ, we cannot issue a load if there is an ambiguous store between it and the head of the LSQ, or between it and a store in the LSQ to the same address.

B and C have no ambiguous stores between them and the head of the LSQ.

I has no ambiguous store between it and the resolved store in slot F to the same address.

7. Short answer – numbers and letters [12 points]

You must answer each question in 20 words or less.

a) Loads and Stores. Answer each question in 20 words or less.
   i) Why does a compiler have to be careful about hoisting a load above a store? **[3]**
   If there is a store->load address conflict then the load could have the incorrect value if hoisted.

   ii) How does a dynamic out-of-order processor deal with the same issue? **[3]**
   LSQ only sends loads to memory when there are no stores in front of them that they're dependent on.

b) A RAS helps predict where returns will branch to. Why do function return statements require a special structure but function calls do not? **[3]**
   Function calls are handled by the branch predictor instead, and since returns are dependent on the initial jump location, they cannot be handled by just a branch predictor.

c) What is the main advantage of a virtually addressed cache over a physically addressed one? **[3]**
   Since you don't need to translate the virtual address to a physical one before the cache lookup in a virtually addressed cache, the latency of a cache lookup will be lower.

8. I have no memory of that [8 points]

You are back to working for HobTech, and they want you to design a processor using a memory technology that supports *very* low memory bandwidth. Which of the following options would you likely choose to deal with this limitation? Very briefly justify each answer.

a) RISC or CISC ISA? **[2]**
CISC because it is usually more code dense.

b) Large cache blocks or small cache blocks? **[2]**
Small cache blocks, so you reduce evictions and only retrieve the memory that you need.

c) Store-to-load forwarding or no store-to-load forwarding? **[2]**
Store-to-load forwarding as the LSQ will reduce the number of loads sent to memory.

d) Prefetching or no prefetching? **[2]**
No, while it helps performance, it will potentially fetch useless lines such as in the case of a branch mispredict which will result in more fetches being sent to memory.

Also correct: Yes, because if the prefetcher is given the lowest priority, it can utilize the bandwidth when it would have otherwise gone unused.

9. It's All About the Pentiums [15 points]

Consider the following tables that represent the state of a processor that implements what we have called the P6 scheme:

RAT	
Arch Reg #	**ROB# (-- if in ARF)**
0	--
1	8
2	--
3	--
4	--
5	7

ROB					
Buffer Number	**PC**	**Done with EX?**	**Dest Arch Reg #**	**Value**	**Head/Tail**
0	~~20~~	~~Y~~	~~1~~	~~12~~	T
1	~~24~~	~~Y~~	~~1~~	~~-12~~	
2	~~28~~	~~Y~~	~~3~~	~~0~~	
3	~~32~~	~~Y~~	~~-~~	~~--~~	
4	~~36~~	~~Y~~	~~1~~	~~14~~	
5	~~100~~	~~Y~~	~~3~~	~~1~~	
6	104	N	1	--	H
7	108	N	5	--	
8	112	N	1	--	

RS						
RS #	**Op Type**	**OP1 Ready?**	**OP1 ROB/Value**	**OP2 Ready?**	**OP2 ROB/Value**	**Dest ROB**
0	~~+~~	~~Y~~	~~12~~	~~Y~~	~~1~~	~~1~~
1	~~+~~	~~N~~	~~1~~	~~Y~~	~~12~~	~~2~~
2	~~+~~	~~Y~~	~~-1~~	~~Y~~	~~-1~~	~~5~~
3	+	Y	-1	N	6	7
4	*	Y	3	N	6	8

ARF	Reg #	0	1	2	3	4	5
	Value	1	-12	3	1	6	-1

The instruction at PC 32 is a branch that has been predicted not-taken, but it is actually taken. The destination of the branch is PC 100, where the following code resides:

```
R3=R3+R0        // A
R1=R1+R3        // B
R5=R5+R1        // C
R1=R2*R1        // D
```

Show the state of the above tables if instruction A has retired, instruction B has been issued but has not finished execution, while C and D have progressed as far along as possible. Be sure to label the head and tail of the ROB. Please place instruction A in slot 5 of the ROB. When other arbitrary decisions need to be made, you are to just make them. **[15 points]**