CMU 18-447 Introduction to Computer Architecture, Spring 2014

Final Exam

Date: Tue., 5/6

Instructor: Onur Mutlu
TAs: Rachata Ausavarungnirun, Varun Kohli, Xiao Bo Zhao, Paraj Tyle

Name:

Legibility & Name (5 Points):	
Problem 1 (40 Points):	
Problem 2 (45 Points):	
Problem 3 (45 Points):	
Problem 4 (40 Points):	
Problem 5 (40 Points):	
Problem 6 (35 Points):	
Problem 7 (40 Points):	
Problem 8 (85 Points):	
Bonus (30 Points):	
Total (375 + 30 Points):	

Instructions:

1. This is a closed book exam. You are allowed to have three letter-sized cheat sheets.
2. No electronic devices may be used.
3. This exam lasts 3 hours.
4. Clearly indicate your final answer for each problem.
5. Please show your work when needed.
6. Please write your initials at the top of every page.
7. Please make sure that your answers to all questions (and all supporting work that is required) are contained in the space required.

Tips:

- **Be cognizant of time.** Do not spend to much time on one question.
- **Be concise.** You will be penalized for verbosity.
- **Show work when needed.** You will receive partial credit at the instructors' discretion.
- **Write legibly.** Show your final answer.

1. The GPU Strikes Back, Just Like Last Year... [40 points]

We define the *SIMD utilization* of a program run on a GPU as the fraction of SIMD lanes that are kept busy with *active threads* during the run of a program.

The following code segment is run on a GPU. Each thread executes **a single iteration** of the shown loop. Assume that the data values of the arrays A, B, C, and D are already in vector registers so there are no loads and stores in this program. (Hint: Notice that there are 5 instructions in each thread and each instruction takes the same number of cycles.) A warp in the GPU consists of 64 threads. There are 64 SIMD lanes in the GPU.

```
for (i = 0; i < 8*1024*1024; i++) {
    B[i] = A[i] - C[i];        // Instruction 1
    if (A[i] > 0)              // Instruction 2
       A[i] = A[i] * C[i];     // Instruction 3
       B[i] = A[i] + B[i];     // Instruction 4
       C[i] = B[i] + 1;        // Instruction 5
    }
}
```

(a) How many warps does it take to execute this program?

2^{17} warps
Warps = (Number of threads) / (Number of threads per warp)
Number of threads = 2^{23} (i.e., one thread per loop iteration).
Number of threads per warp = $64 = 2^6$ (given).
Warps = $2^{23}/2^6 = 2^{17}$

(b) When we measure the SIMD utilization for this program with one input set, we find that it is 55%. What can you say about arrays A, B, and C? Be precise.

A: 16 of every 64 consecutive elements of A are positive.
If A[i] ¡= 0, Only 2/5th of the instructions are executed. If 1/4th of the threads execute at full utilization, and 3/4 of the threads execute at 40% utilization, the total utilization becomes 55%. For this to happen, 16 of every 64 consecutive elements of A must be positive.

B: Nothing.

C: Nothing.

(c) Is it possible for this program to yield a SIMD utilization of 100% (circle one)?

YES NO

If YES, what should be true about arrays A, B, C for the SIMD utilization to be 100%? Be precise.

A: For every 64 consecutive values of A, A's are positive or all A's are non-positive.

B: Nothing.

C: Nothing.

If NO, explain why not.

(d) Is it possible for this program to yield a SIMD utilization of 40% (circle one)?

YES [NO]

If YES, what should be true about arrays A, B, and C for the SIMD utilization to be 40%? Be precise.

A:

B:

C:

If NO, explain why not.

The minimum SIMD utilization of every warp is slightly larger than 40%. This is achieved by 63 threads having 40% utilization and 1 thread having full utilization. Therefore, it is not possible to achieve utilization of 40%.

2. Prefetching [45 points]

A processor is observed to have the following access pattern to cache blocks. Note that the addresses are **cache block addresses**, not byte addresses.

A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,
A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,
A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48

Assume we have the following hardware prefetchers. None of them employ confidence bits, but they all start out with empty tables at the beginning of the access stream shown above. Unless otherwise stated, assume that 1) each access is separated long enough in time such that all prefetches issued can complete before the next access happens, 2) the prefetchers have large enough resources to detect and store access patterns, 3) the prefetchers prefetch into a fully-associative cache whose size is 8 cache blocks.

(a) A stream prefetcher with prefetch degree of 1 (i.e., a next-block prefetcher).

Circle which of the cache block addresses are prefetched by the prefetcher:

A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,
A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,
A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48

None.

(b) A next-block prefetcher with prefetch degree of 4 (i.e., a next-4-blocks prefetcher).

Circle which of the cache block addresses are prefetched by the prefetcher:

A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,
A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,
A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48

None.

(c) A next-block prefetcher with prefetch degree of 8 (i.e., a next-8-blocks prefetcher).

Circle which of the cache block addresses are prefetched by the prefetcher:

A, [A + 8], [A + 16], [A + 24], [A + 32], [A + 40], [A + 48],
A, [A + 8], [A + 16], [A + 24], [A + 32], [A + 40], [A + 48],
A, [A + 8], [A + 16], [A + 24], [A + 32], [A + 40], [A + 48]

(d) A stride prefetcher (that works on cache block addresses).

Circle which of the cache block addresses are prefetched by the prefetcher:

A, A + 8, [A + 16], [A + 24], [A + 32], [A + 40], [A + 48],

A, A + 8, [A + 16], [A + 24], [A + 32], [A + 40], [A + 48],

A, A + 8, [A + 16], [A + 24], [A + 32], [A + 40], [A + 48]

(e) A Markov prefetcher with a correlation table of 8 entries (assume each entry can store one next address).

Circle which of the cache block addresses are prefetched by the prefetcher:

A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,

A, [A + 8], [A + 16], [A + 24], [A + 32], [A + 40], [A + 48],

[A], [A + 8], [A + 16], [A + 24], [A + 32], [A + 40], [A + 48]

(f) A Markov prefetcher with a correlation table of 4 entries (assume each entry can store one next address).

Circle which of the cache block addresses are prefetched by the prefetcher:

A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,

A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48,

A, A + 8, A + 16, A + 24, A + 32, A + 40, A + 48

None.

(g) What is the minimum number of correlation table entries that should be present for the Markov prefetcher to attain the same accuracy and coverage as the Markov prefetcher with infinite number of correlation table entries? Show your work to get partial credit.

7, since there are only 7 patterns to remember.

(h) Based on the above access stream, which prefetcher would you choose if the metric you are optimizing for were memory bandwidth? Why?

Markov, because it has the highest accuracy (i.e. least number of prefetches wasted). Stride prefetch was accepted if a good argument was made in favor of it.

(i) Which prefetcher would you choose if you wanted to minimize hardware cost while attaining reasonable performance benefit from prefetching? Explain your reasoning in no more than 20 words.

Stride prefetcher. Simple to implement, and has the highest accuracy and coverage apart from Markov, which has more complicated hardware.

3. Out-of-Order Execution [45 points]

In this problem, we will give you the state of the Register Alias Table (RAT) and Reservation Stations (RS) for an out-of-order execution engine that employs Tomasulo's algorithm. Your job is to determine the original sequence of **five instructions** in program order.

The out-of-order machine in this problem behaves as follows:

- The frontend of the machine has a one-cycle fetch stage and a one-cycle decode stage. The machine can fetch one instruction per cycle, and can decode one instruction per cycle.
- The machine dispatches one instruction per cycle into the reservation stations, in program order. Dispatch occurs during the decode stage.
- An instruction always allocates the first reservation station that is available (in top-to-bottom order) at the required functional unit.
- When a value is captured (at a reservation station) or written back (to a register) in this machine, the old tag that was previously at that location is *not cleared*; only the valid bit is set.
- When an instruction in a reservation station finishes executing, the reservation station is cleared.
- Both the adder and multiplier are fully pipelined. An add instruction takes 2 cycles. A multiply instruction takes 4 cycles.
- When an instruction completes execution, it broadcasts its result. A dependent instructions can begin execution in the next cycle if it has all its operands available.
- When multiple instructions are ready to execute at a functional unit, the *oldest* ready instruction is chosen.

Initially, the machine is empty. Five instructions then are fetched, decoded, and dispatched into reservation stations. When the final instruction has been fetched and decoded, one instruction has already been written back. Pictured below is the state of the machine at this point, after the fifth instruction has been fetched and decoded:

RAT

Reg	V	Tag	Value
R0	1		13
R1	0	A	8
R2	1		3
R3	1		5
R4	0	X	255
R5	0	Y	12
R6	0	Z	74
R7	1		7

ADD

	Src 1			Src 2		
	Tag	V	Value	Tag	V	Value
A	-	1	5	Z	0	-
B						
C						

MUL

	Src 1			Src 2		
	Tag	V	Value	Tag	V	Value
X	A	1	8	-	1	7
Y	X	0	-	-	1	13
Z	-	1	3	-	1	8

(a) Give the five instructions that have been dispatched into the machine, in program order. The source registers for the first instruction can be specified in either order. Give instructions in the following format: "opcode destination ⇐ source1, source2."

ADD R1 ⇐ R2, R3

MUL R4 ⇐ R1, R7

MUL R5 ⇐ R4, R0

MUL R6 ⇐ R2, R1

ADD R1 ⇐ R3, R6

(b) Now assume that the machine flushes all instructions out of the pipeline and restarts fetch from the first instruction in the sequence above. Show the full pipeline timing diagram below for the sequence of five instructions that you determined above, from the fetch of the first instruction to the writeback of the last instruction. Assume that the machine stops fetching instructions after the fifth instruction.

As we saw in class, use "F" for fetch, "D" for decode, "En" to signify the nth cycle of execution for an instruction, and "W" to signify writeback. You may or may not need all columns shown.

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction: ADD R1 ⇐ R2, R3	F	D	E1	E2	W									
Instruction: MUL R4 ⇐ R1, R7		F	D		E1	E2	E3	E4	W					
Instruction: MUL R5 ⇐ R4, R0			F	D					E1	E2	E3	E4	W	
Instruction: MUL R6 ⇐ R2, R1				F	D	E1	E2	E3	E4	W				
Instruction: ADD R1 ⇐R3, R6					F	D				E1	E2	W		

Finally, show the state of the RAT and reservation stations after **10 cycles** in the blank figures below.

RAT

Reg	V	Tag	Value
R0	1		13
R1	0	A	8
R2	1		3
R3	1		5
R4	1	X	56
R5	0	Y	12
R6	1	Z	24
R7	1		7

ADD

	Src 1			Src 2		
	Tag	V	Value	Tag	V	Value
A	-	1	5	Z	1	24
B						
C						

MUL

	Src 1			Src 2		
	Tag	V	Value	Tag	V	Value
X						
Y	X	1	56	-	1	13
Z						

4. Amdahl's Law [40 points]

Consider the following three processors (X, Y, and Z) that are all of varying areas. Assume that the single-thread performance of a core increases with the square root of its area.

Processor X	**Processor Y**	**Processor Z**
Core Area = A	Core Area = $4A$	Core Area = $16A$

(a) You are given a workload where S fraction of its work is serial and $1 - S$ fraction of its work is **infinitely** parallelizable. If executed on a die composed of 16 Processor X's, what value of S would give a speedup of 4 over the performance of the workload on just Processor X?

$\frac{1}{\left(S+\frac{1-S}{16}\right)} = 4$

$S = .2$

(b) Given a homogeneous die of area $16A$, which of the three processors would you use on your die to achieve maximal speedup? What is that speedup over just a single Processor X? Assume the same workload as in part(a).

Processor Y. Speedup of 5.
We know that using Processor X gives us a speedup of 4.
Since Processor Z would take up the full area and the performance increases with the square root of its area, we know that Processor Z gives us a speed up of 4.

Processor Y has a single-threaded speedup of 2 over Processor X.
$speedup = \frac{1}{\left(.2+\frac{.8}{4}\right)} \times 2$
$speedup = 5$

We would use processor Y and get a speedup of 5.

(c) Now you are given a heterogeneous processor of area $16A$ to run the above workload. The die consists of 1 Processor Y and 12 Processor X's. When running the workload, all sequential parts of the program will be run on the larger core while all parallel parts of the program run exclusively on the smaller cores. What is the overall speedup achieved over a single Processor X?

Let n be the speedup of Processor Y over Processor X
$speedup = \frac{1}{\left(\frac{.2}{n}+\frac{.8}{16-n^2}\right)}$
$speedup = \frac{1}{\left(\frac{.2}{2}+\frac{.8}{12}\right)}$
$speedup = 6$

(d) One of the programmers decides to optimize the given workload so that it has 4% of its work in serial sections and 96% of its work in parallel sections. Which configuration would you use to run the workload if given the choices between the processors from part (a), part (b), and part (c)?

part (a) $speedup = \frac{1}{(.04 + \frac{.96}{16})} = 10$
part (b) $speedup = \frac{1}{(.04 + \frac{.96}{4})} \times 2 = 7.14$
part (c) $speedup = \frac{1}{(\frac{.04}{2} + \frac{.96}{12})} = 10$
You would use the processor from part (a) because it gives you the maximum speed up while being less complex to implement than the heterogeneous set up.

(e) What value of S would warrant the use of Processor Z over the configuration in part (c)?

Again, the speedup of Processor Z over one Processor X is 4. $\frac{1}{(\frac{S}{2} + \frac{1-S}{12})} = 4$
$S \geq .4$

(f) Typically, for a realistic workload, the parallel fraction is not infinitely parallelizable. What are the three fundamental reasons why?

1. Synchronization
2. Load imbalance
3. Resource contention

(g) Name a technique we discussed in class that takes advantage of the heterogeneous architecture to minimize the effect of one of the above reasons?

Accelerated Critical Sections or Bottleneck Identification and Scheduling

5. Cache Coherence [40 points]

We have a system with 4 byte-addressable processors. Each processor has a private 256-byte, direct-mapped, write-back L1 cache with a block size of 64 bytes. Coherence is maintained using the MESI protocol we discussed in class. Accessible memory addresses range from 0x30000000 - 0x3FFFFFFF. We show below the initial tag store state of the four caches.

Initial State

Cache 0		
	Tag	*MESI State*
Set 0	0x3FFFFF	S
Set 1	0x000000	I
Set 2	0x330000	E
Set 3	0x300000	I

Cache 1		
	Tag	*MESI State*
Set 0	0x3FFFFF	S
Set 1	0x300000	E
Set 2	0x300004	M
Set 3	0x330000	E

Cache 2		
	Tag	*MESI State*
Set 0	0x3FFFFF	S
Set 1	0x330000	E
Set 2	0x32FFFF	M
Set 3	0x32FFFF	E

Cache 3		
	Tag	*MESI State*
Set 0	0x000000	I
Set 1	0x000000	I
Set 2	0x000000	I
Set 3	0x000000	I

After 5 memory instructions are executed in this system, we find the final tag store state of the four caches to be as follows:

Final State

Cache 0		
	Tag	*MESI State*
Set 0	0x3FFFFF	I
Set 1	0x000000	I
Set 2	0x330000	S
Set 3	0x300000	I

Cache 1		
	Tag	*MESI State*
Set 0	0x3FFFFF	I
Set 1	0x300000	E
Set 2	0x330000	S
Set 3	0x330000	E

Cache 2		
	Tag	*MESI State*
Set 0	0x3FFFFF	I
Set 1	0x330000	E
Set 2	0x32FFFF	M
Set 3	0x32FFFF	M

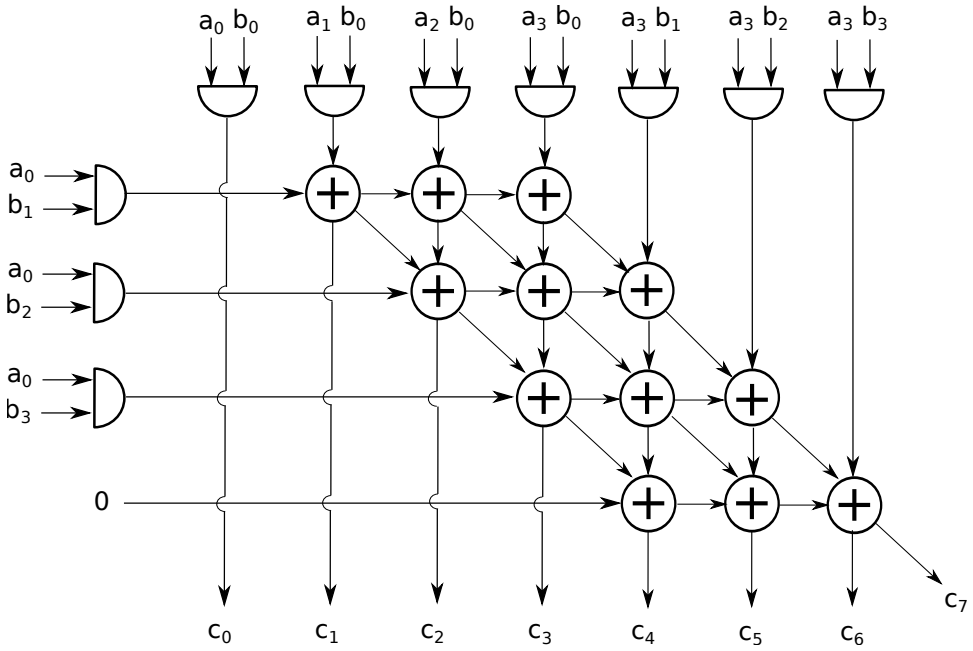
Cache 3		
	Tag	*MESI State*
Set 0	0x333333	E
Set 1	0x000000	I
Set 2	0x000000	I
Set 3	0x32FFFF	I

Assume that the offset within a cache block is 0 for all requests. In the below diagram, you are given partially what happened with each of the 5 memory instructions. In particular, we show you whether the instruction caused an eviction or a write-back in any cache and which cache the instruction was serviced by. We ask you to complete three pieces of information: 1) address of each request (Address), 2) whether the request is a Read or a Write (R/W), 3) which caches the request caused an eviction from if an eviction occurred due to the request (Evicted from cache 0, 1, 2, 3).

Instruction			Evicted from cache						Request served by cache			
Order	**Address**	**R/W**	Eviction?	**0**	**1**	**2**	**3**	Write back?	0	1	2	3
1	0x3FFFFF00	W	Yes	X	X	X		No				X
2	0x33000080	R	Yes		X			Yes		X		
3	0x32FFFFC0	R	No					No				X
4	0x32FFFFC0	W	Yes				X	No			X	
5	0x33333300	R	Yes				X	Yes				X

6. Systolic Arrays [35 points]

The following diagram is a systolic array that performs the multiplication of two 4-bit binary numbers. Assume that each adder takes one cycle.



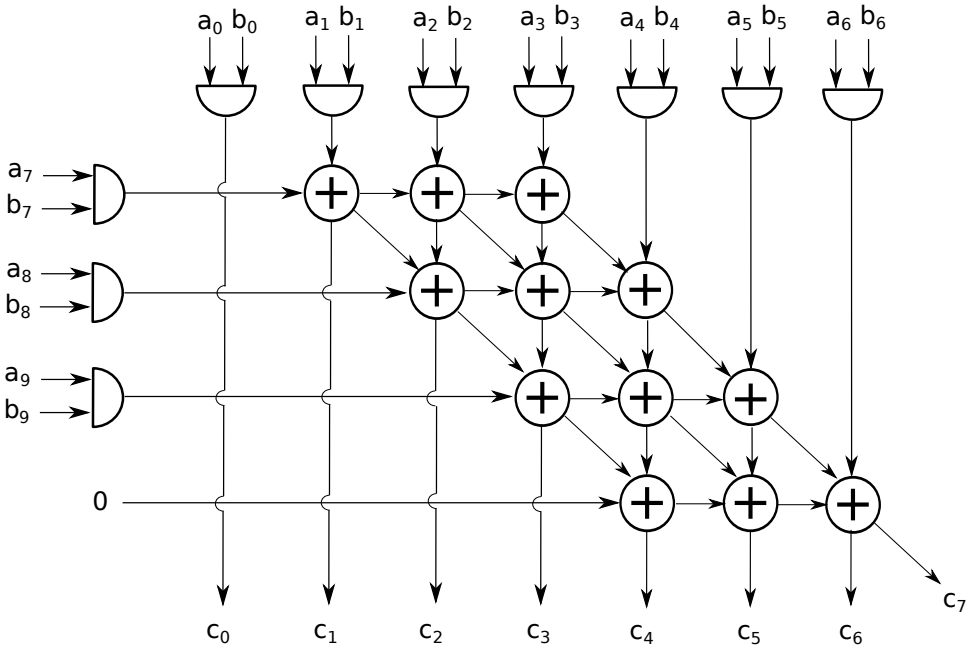
(a) How many cycles does it take to perform *one* multiplication?

9 cycles

(b) How many cycles does it take to perform *three* multiplications?

13 cycles.
Note that some input ports have to hold their values for an extra cycle because their nodes need to wait for inputs from the previous nodes. You cannot completely overlap latencies of different nodes. However, because no one was aware of this, we accepted 11 cycles as another possible answer.

(c) Fill in the following table, which has a list of inputs (a 1-bit binary number) such that the systolic array produces the following outputs, in order: $5 * 5$, $12 * 9$, $11 * 15$. Please refer to the following diagram to use as reference for all the input ports.



$5x5 = 101x101$ $12x9 = 1100x1001$ $11x15 = 1011x1111$

Cycles	Row Inputs														Column Inputs					
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	b_0	b_1	b_2	b_3	b_4	b_5	b_6	a_7	a_8	a_9	b_7	b_8	b_9
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
2	0	0	1	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0
4	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	1	0
5	0	1	0	1	0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	0
6	0	1	0	1	0	0	0	0	1	1	1	0	0	0	1	1	1	1	0	0
7	0	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1
8	0	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0	1	1
9	1	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
10	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	1
11	0	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	1	0	0	1
12	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
13	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7. Cache Mystery [40 points]

A byte-addressable system with 32-bit addresses ships with a two-way set associative, write-back, sectored cache with perfect LRU replacement. Each block consists of four sectors. The tag store requires a total of 4224 bits of storage. What is the block size of the cache? (Hint: $4224 = 2^{12} + 2^7$)

Answer:

2^{17} bytes

Show all your work.

Assume t tag bits, n index bits and b block bits.
$t + n + b = 32$
LRU = 1 bit per set
Valid Bit = 1 bit per sector = 4 bits total
Dirty Bit = 1 bit per sector = 4 bits total
Tag bits = t bits per block
Number of sets = 2^n
Number of blocks = $2 * 2^n$ (2-way associative)
Tag store size = $2^n + 2 * 2^n * (8 + t) = 2^n * (17 + 2t)$
We get $2^n * (17 + 2t) = 2^7 + 2^{12}$
$2^n * (17 + 2t) = 2^7 * 33$
So, $n = 7$ and $t = 8$. As a result, $b = 17$
Therefore, the block size is 2^{17}, and a sector is 2^{15} bytes.

8. Potpourri [85 points]

(a) **Philosophy (The Final Exam Remix Version)** [30 points]

In class, we have discussed many times the tradeoff between programmer versus microarchitect.

Among the following sets of two concepts, circle the one that makes programmer's life harder and **at the same time** microarchitect's (or hardware designer's) life easier. Interpret "programmer" broadly as we did in class, e.g., a compiler writer is a programmer, too. If the two options are equal, select NEITHER; if you cannot decide which option is better based on information provided in the question, select UNCLEAR. Assume the goal is to have high performance and all else is equal in a system where you are considering the two concepts.

Questions	Choices	
i	Hardware based cache coherence	[Software based cache coherence]
	NEITHER	UNCLEAR
ii	Sequential consistency	[Weak consistency]
	NEITHER	UNCLEAR
iii	Snoopy cache coherence	Directory based cache coherence
	[NEITHER]	UNCLEAR
iv	Accelerated critical sections	[No acceleration of critical sections]
	NEITHER	UNCLEAR
v	QoS-aware memory scheduler	[FR-FCFS memory scheduler]
	NEITHER	UNCLEAR
vi	[In-order execution]	Out-of-order execution
	NEITHER	UNCLEAR
vii	Precise exceptions	[Imprecise exceptions]
	NEITHER	UNCLEAR

Initials: _____

Questions	Choices	
viii	[Static branch prediction]	Dynamic branch prediction
	NEITHER	UNCLEAR
ix	Delayed branching	Predicated execution
	NEITHER	[UNCLEAR]
x	Virtually indexed physically tagged caches	Physically indexed physically tagged caches
	NEITHER	[UNCLEAR]
xi	Instruction scheduling using superblocks	Instruction scheduling using hyperblocks
	NEITHER	[UNCLEAR]
xii	DRAM that supports RowClone	DRAM that does not support RowClone
	NEITHER	[UNCLEAR]
xiii	Runahead execution based prefetching	[Helper thread based prefetching] (using software helper threads)
	NEITHER	UNCLEAR
xiv	Markov prefetching	Stream prefetching
	[NEITHER]	UNCLEAR
xv	[Fewer architectural registers]	More architectural registers
	NEITHER	UNCLEAR

Some other answers were also accepted as correct.

.

(b) **Performance Analysis [10 points]**

Your friend is comparing the performance of a program on two systems.

1. System I has 64 simple symmetric cores.
2. System II has 64 large symmetric cores.

He runs the program with 64 threads on both machines and concludes that the obtains the following execution times:

On System I: 1000 seconds

On System II: 333 seconds

What would you conclude about the speedup of System II over System I for this program?

We do not know.

Explain your reasoning:

We do not know if it is a fair comparison because we do not know how many threads is the best number for each architecture.

(c) **Buffering [10 points]**

Assume a friend you have is designing the next generation interconnect for the next generation multi-core processor that consists of 4096 cores. She has designed a topology where each router has 6 input links and 4 output links. She is considering using either deflection routing or packet dropping and retransmission to eliminate the need for buffers in the routers. What would you suggest to your friend? Which strategy should she choose to keep the routers bufferless?

Circle one:

Deflection routing | **Packet dropping and retransmission** | Does not matter

Explain your reasoning:

Cannot do deflection routing because number of input ports is > number of output ports.

(d) **Interconnect Design [15 points]**

Assume we have a 2D mesh interconnect connecting 256 nodes, where each node consists of a core, a cache, and part of the shared physical memory, of a shared memory multiprocessor. The network supports all types of communication (for the purposes of this question, the type of communication does not matter). You are told that the "load" on this network is very low. In other words, there is very little traffic injected into the network at any given point of time. When communication happens, it is usually point to point, i.e., one node sends data to another for a prolonged amount of time, and only those two nodes are communicating for that duration (for the most part).

Given this information, pick the best design choices for this network. Your goal is to maximize performance and minimize energy efficiency and design complexity, all at the same time, if possible (the Holy Grail, yes!).

Which of the following would you choose? Circle one in each sub-question and explain.

i) Choose your switching strategy in each router:

[Circuit switching] Packet switching Does not matter

Pick one above, and explain.

Most traffic is point to point and the network has low load

ii) Choose the type of your routing algorithm:

[Deterministic routing] Oblivious routing Adaptive routing Does not matter

Pick one above, and explain.

No need to make the routing algorithm complex because contention is low in the network.

iii) Choose your buffering strategy in each router. When two packets contend for the same output port:

Buffer one [Deflect one] Drop one and ask for retransmission Does not matter

Pick one above, and explain.

Network load is low → deflection can efficiently handle contention.
Alternatively, buffering (with a very small number of buffers) is also acceptable if you argue that it does not have to deal with the livelock problem

(e) **Multiprocessors [10 points]**

"On a sequentially-consistent multiprocessor, the user-visible (or, software-visible) outputs of the same program are guaranteed to be the same across different times when the program is executed with the same input set."

Is the above statement True? Circle one: YES [NO]

Explain:

Sequential consistency does not dictate anything about different runs of a program.

"On a cache-coherent multiprocessor, the user-visible (or, software-visible) outputs of the same program are guaranteed to be the same across different times when the program is executed with the same input set."

Is the above statement True? Circle one: YES [NO]

Explain:

Cache coherence does not dictate anything about different runs of a program.

(f) **Pointers [10 points]**

Assume we have a byte-addressable computer with a 64-bit virtual address space and a 4KB page size. Assume that the data is laid out such that it is aligned at the data type size boundaries. The following shows the contents of the valid translations cached in the TLB for the running program:

VPN	PFN
0x008deadbeef12	0x0dd75
0x0feed0beef012	0x00254

The following shows the contents of a 64-byte L3 cache block starting at address 0x254900. The L3 cache is physically addressed.

feed0beef056adb8	00008deadc450000	000000dd7533f070	00254000ffde7800
[008deadbee88f000]	[0feed0beef008000]	0feed0beef54328a	00008deadbeef070

Which one of the values are likely pointers? Circle above clearly.

9. [BONUS] DRAM Refresh [30 points]

A memory system has the following configuration:

- There are two DRAM channels.
- Each channel has two ranks of DRAM chips.
- Each memory channel is controlled by a separate memory controller.
- Each rank of DRAM contains eight banks.
- Each bank contains 32K rows.
- Each row in one bank is 8KB.

Assume the minimum retention time among all DRAM rows in the system is 64 ms. In order to ensure that no data is lost, every DRAM row is refreshed once per 64 ms. Every DRAM row refresh is initiated by a command from the memory controller which occupies the command bus on the associated memory channel.

Let us consider a 1.024 second span of time.

We define *utilization* (of a resource such as a bus or a memory bank) as the fraction of total time for which a resource is occupied by a refresh command.

For each calculation in this section, you may leave your answer in *simplified* form in terms of powers of 2 and powers of 10.

(a) How long does each refresh command occupy the command bus (in ns) such that across all memory channels, the command bus utilization due to refreshes is 8.192%? (Hint: $8.192 = 2^{13}/1000$)

10 ns

(b) How long does each refresh command occupy the DRAM banks (in ns) such that across all the banks, the bank utilization due to refreshes is 8.192%?

160 ns

(c) What data bus utilization, across all memory channels, is directly caused by DRAM refreshes?

0%

(d) How many refreshes are performed by the memory controllers during the 1.024 second period in total across both memory channels combined?

$2^{15+1+3+4}$ refreshes per channel; 2^{24} = 16M across 2 channels.

(e) The system designer wishes to reduce the overhead of DRAM refreshes in order to improve system performance and reduce the energy spent in DRAM. A key observation is that not all rows in the DRAM chips need to be refreshed every 64 ms. In fact, rows need to be refreshed only at the following intervals in this particular system:

Required Refresh Rate	Number of Rows for the entire DRAM
64 ms	2^4
128 ms	2^8
256 ms	all other rows

Given this distribution, if all rows are refreshed only as frequently as required to maintain their data, how many refreshes are performed by the memory controllers during the 1.024 second period in total across both memory channels combined?

$(2^4 * 16 + 2^8 * 8 + (2^{20} - 2^4 - 2^8) * 4)$

(f) The system designer wants to achieve this reduction in refresh overhead by refreshing rows less frequently when they need less frequent refreshes. In order to implement this improvement, the system needs to track every row's required refresh rate. What is the minimum number of bits of storage required to track this information without any information loss?

2 Mbits (2 bits per row)

Stratchpad

Stratchpad

Stratchpad