Final Exam

```
    ____ ____ ___ ____   _______ ___
   / ___/ ___/ __/ ___/  /__ /__  / _ \
  / _/ / _// /   \___ \  /_ \ / / / // /
 / /___/ /__/ /___ ___) / ___) |/ / /_/ /
/_____/_____\____/____/ /____//_/  \___/
```

EECS 370 Fall 2023: Introduction to Computer Organization

You are to abide by the University of Michigan College of Engineering Honor Code. Please sign below to signify that you have kept the honor code pledge:

I have neither given nor received aid on this exam,
nor have I concealed any violations of the Honor Code.

Signature: ______________________________

Name: ______________________________

Uniqname: ______________________________

Uniqname of person sitting to your ***Right***
(Write ⊥ if you are at the end of the row) ______________________________

Uniqname of person sitting to your ***Left***
(Write ⊥ if you are at the end of the row) ______________________________

Exam Directions:

- You have **120 minutes** to complete the exam. Don't spend too much time on a problem.
- There are **6** questions in the exam on **13** pages (double-sided). **Please flip through your exam to ensure you have all pages.**
- You must show your work to be eligible for partial credit!
- Write ***legibly*** and ***dark*** enough for the scanners to read your answers.
- **Write your uniqname on the line provided at the top of each page.**

Exam Materials:

- You are allotted **one 8.5 x 11 double-sided** note sheet to bring into the exam room.
- You are allowed to use calculators that do not have an internet connection. All other electronic devices, such as cell phones or anything or calculators with an internet connection, are strictly forbidden.

Problem	1	2	3	4	5	6
Pages	2	3-5	6	7	8-10	11-13
Point Value	15	20	10	10	20	25

Problem 1: I guess they sometimes miss, huh? 15 points

For this problem, consider a cache with the following specifications:

- Cache Size is 32 Bytes
- Addresses are 20 bits
- Cache is 2-way set associative
- Cache block size is 8 Bytes
- Write Back policy
- No Allocate on Write

1. How many sets does the cache have? ____________

2. How many bits of the address would be for the cache tag? ____________

3. Say the cache begins empty, and the system makes the accesses below. For each row, fill in the bubble if the access to that address results in a cache hit or miss.

Row	Instruction	Address	Cache Hit or Miss
1	load	0x3B8D9	◯ Hit ◯ Miss
2	load	0x3B8D7	◯ Hit ◯ Miss
3	**store**	0x3967A	◯ Hit ◯ Miss
4	load	0x3967E	◯ Hit ◯ Miss
5	**store**	0x39678	◯ Hit ◯ Miss
6	load	0xE1F3D	◯ Hit ◯ Miss
7	load	0x3B8D4	◯ Hit ◯ Miss
8	**store**	0x3B8DC	◯ Hit ◯ Miss
9	**store**	0x3B8DF	◯ Hit ◯ Miss

4. If the cache were instead to have a write-through policy, which access would change whether or not main memory is accessed? (Select and fill in the bubble for one.)

   ◯ Row 3 ◯ Row 5 ◯ Row 8 ◯ Row 9

5. If the cache were instead to be fully associative, which access would turn from a cache miss into a hit? (Select and fill in the bubble for one.)

   ◯ Row 2 ◯ Row 6 ◯ Row 7 ◯ Row 8

Problem 2: Pipe Dreams 20 points

Fill in the bubble for the best answer in each of the following questions.

1. Given a multi-cycle processor and a pipelined processor implementing the same ISA and with the same clock frequency, we would expect:
   - ◯ The multi-cycle would have a higher CPI than the pipelined processor.
   - ◯ The multi-cycle and pipelined processor would have the same CPI.
   - ◯ The multi-cycle would have a lower CPI than the pipelined processor.
2. Given 2 multi-cycle processors with the same clock period, but one which uses a RISC ISA and one which uses a CISC ISA, we would expect:
   - ◯ The processor with the RISC ISA will have a lower CPI.
   - ◯ The two processors will have similar CPIs.
   - ◯ The processor with the CISC ISA will have a lower CPI.
3. Given 2 equivalent pipelined processors, one which requires avoidance, and one which uses detect-and-stall to resolve all hazards, we would expect:
   - ◯ The pipeline with detect-and-stall will have a lower CPI since it will execute fewer instructions.
   - ◯ The two pipelines will have the same CPI since they have the same cycle and instruction counts.
   - ◯ The pipeline with detect-and stall will have a higher CPI since it will run for more cycles.
   - ◯ The pipeline with detect-and-stall will have a higher CPI since it will execute fewer instructions.
4. Given 2 equivalent pipelined processors, except that one which uses detect-and-stall to solve data hazards, and one which uses detect-and-forward where possible to solve data hazards, we would expect:
   - ◯ The pipeline with detect-and-stall will have a lower CPI since it will execute more instructions.
   - ◯ The two pipelines will have the same CPI because of control hazards.
   - ◯ The pipeline with detect-and-stall will have a higher CPI since it will run for more cycles.
   - ◯ The pipeline with detect-and-stall will have a higher CPI since it will execute fewer instructions.
5. Given 2 equivalent pipelined processors, one which uses detect-and-stall to solve control hazards, and one which uses speculate-and-squash and a branch predictor with 10% correct prediction rate, we would generally expect:
   - ◯ The pipeline with detect-and-stall will have a higher CPI.
   - ◯ The two pipelines will have the same CPI.
   - ◯ The pipeline with detect-and-stall will have a lower CPI.

6. Say we took the 5-stage LC2K pipeline discussed in class and shortened the pipeline by combining the EX and MEM stages into 1 stage. We would generally expect:
   - ◯ The shorter pipeline would have a higher CPI and higher or same clock period.
   - ◯ The shorter pipeline would have a higher CPI but a lower or same clock period.
   - ◯ The shorter pipeline would have a lower CPI but a higher or same clock period.
   - ◯ The shorter pipeline would have a lower CPI and a lower or same clock period.

7. Say we added an instruction cache to a processor that didn't previously have one. Which of the following is most true?
   - ◯ A program with few branches and also a program with many loops would both benefit from the I cache primarily due to temporal locality.
   - ◯ A program with few branches and also a program with many loops would both benefit from the I cache primarily due to spatial locality.
   - ◯ A program with few branches would benefit from the I cache primarily due to temporal locality, and a program with many loops would benefit from the I cache primarily due to spatial locality.
   - ◯ A program with few branches would benefit from the I cache primarily due to spatial locality, and a program with many loops would benefit from the I cache primarily due to temporal locality.

For the next parts, consider the given LC2K program and pipeline. Assume we run the program on the pipeline. In the given table, indicate the opcode of the instructions that are in the **WriteBack** stage of the pipeline during the specified cycle until the processor halts. Use `noop` to indicate no instruction in the stage. Inclusion of the `halt` opcode is optional. You might not need all cycles. The first cycle is done for you.

8. Fill out the table for the below hazard-free program run on the 5-stage pipeline discussed in class.

```
        lw    0     1     num1
        nor   0     0     2
        noop
        noop
        add   1     1     3
        halt
num1    .fill 100
```

Cycle	1	2	3	4	5	6	7	8	9	10
WB Opcode	noop									

9. Fill out the table for the below program run on the 5-stage pipeline from lecture which uses detect-and-stall to solve data hazards.

```
      lw    0     4     num2
      sw    4     0     0
      halt
num2  .fill 280
```

Cycle	1	2	3	4	5	6	7	8	9	10
WB Opcode	noop									

10. Fill out the table for the below program run on the 5-stage pipeline from class which uses detect-and-forward to solve data hazards, as far as possible.

```
      lw    0     5     num3
      lw    5     6     0
      sw    0     6     num3
      halt
num3  .fill 370
```

Cycle	1	2	3	4	5	6	7	8	9	10
WB Opcode	noop									

11. Fill out the table for the below program run on the 5-stage pipeline from class which uses detect-and-forward to solve data hazards as far as possible, and speculate-and-squash with a 1-bit branch predictor per branch initialized to Not Taken to solve control hazards. For simplicity, assume the BTB/branch predictor are read in the IF stage, and updated in the same cycle as when a branch is resolved.

```
      add   0     0     7
b1    beq   0     7     skip
      halt
skip  nor   0     0     7
      beq   0     0     b1
```

Cycle	1	2	3	4	5	6	7	8	9	10
WB Opcode	noop									

Cycle	11	12	13	14	15	16	17	18	19	20
WB Opcode										

Problem 3: Branch Predictor Matching **10 points**

Consider a program in C, with the corresponding LC2K code:

```
int main(void) {
      int sum = 0;
      for(int i = 3;
                     i <= 6;
                             ++i) {
         // Take branch if i == 7

            if (i % 4 != 0)
         // Take branch if i % 4 == 0
                  sum += i;
      }
         // Take branch unconditionally
}
```

```
      lw    0    4    mask
      add   0    0    5
      lw    0    2    i
      lw    0    3    max
      lw    0    1    one
loop  beq   2    3    end
      nor   2    2    6
      nor   4    6    6
      beq   6    0    skip
      add   5    2    5
skip  add   2    1    2
      beq   0    0    loop
end   halt
mask  .fill -4
i     .fill 3
max   .fill 7
one   .fill 1
```

Say we run the above program on processors with different branch predictors. Match the below branch predictors to their misprediction counts by placing the corresponding letter into the blank next to the **number of mispredictions across all branches**. Each branch predictor will have a unique number of mispredicts. Assume the branch target buffer is preloaded with the target PC of all three branch instructions. You may use the table below for scratch work.

A. Always Not Taken
B. Forward branches predicted Not Taken, Backward branches predicted Taken
C. Last-Time (1-Bit) Predictor per branch, initialized to Not Taken
D. Saturated (2-Bit) Predictor per branch, initialized to Weakly Not Taken
E. Saturated (2-Bit) Predictor per branch, initialized to Weakly Taken

Branch Predictor (A-E)	Total Misprediction Count
	2
	3
	4
	5
	6

(Branch outcome scratch work, not graded)

Problem 4: Wolverine Accesses 10 points

The University has had trouble keeping Wolverine Access working efficiently on course registration days, and has tasked you to help optimize their code to avoid slowdowns and improve quality of life. Consider the following C program fragment run on a **64-bit, memory-aligned system** with a cache that has a **block size of 8 bytes**. Then answer the following questions. Assume `UM_catalog` begins at address 0x1000.

```
typedef struct course{
      char[5] department;
      short course_number;
      unsigned short credits;
      char* professor;
      float median;
      bool graded;
} course;

course UM_catalog[32];

int weighted_grade() {
      float grade = 0;
      unsigned short total_credits = 0;
      for(unsigned int i = 0; i<CATALOG_SIZE; ++i) {
            if(UM_catalog[i].graded) {
                  grade += UM_catalog[i].median * UM_catalog[i].credits;
                  total_credits += UM_catalog[i].credits;
      }     }
      return grade / total_credits;
}
```

Assume the following data type sizes:	
bool	1 byte
char	1 byte
short	2 bytes
int	4 bytes
float	4 bytes
double	8 bytes

1. What is the size of the above course struct? (Select and fill in the bubble for one.)

   ◯ 22 B ◯ 24 B ◯ 32 B ◯ 40 B

2. Provide a reordering of the `course` struct that minimizes its size **AND** minimizes the number of cache misses when the `weighted_grade` function is run.

List the member identifiers in order:

3. Say the cache block size was decreased to 4 bytes. If `weighted_grade` was run once, we would generally expect the new block size to: (Fill in the bubble for all that apply.)

   ◯ Increase Hits ◯ Increase Compulsory Misses ◯ Increase Other Misses

Problem 5: Virtually Correct 20 points

1. In 1-2 sentences, give a disadvantage of using a physically addressed cache, and a disadvantage of using a virtually addressed cache. You should reference the TLB somewhere in your answer.

____

____

____

____

You're working on developing a LC2K operating system and are trying to debug store instructions in virtual memory. Recall that LC2K is a word-addressable architecture. Since it's *Little* Computer 2000, there's not a lot of memory to work with. Virtual addresses are **16 bits** in length, while there are only **4 pages of physical memory**. Each page is **256 words**.

Current State of Physical Memory:

Physical Page 0	Reserved for OS and Page Tables
Physical Page 1	Least Recently Used Data Page (VPN ?)
Physical Page 2	Unused/Free
Physical Page 3	Most Recently Used Data Page (VPN 0, Including Text Section & `.fills`)

After all physical pages are used, virtual pages are replaced based on true LRU policy, ignoring the reserved page.

The system uses a 2-level page table, with bits **15-12** of the virtual address representing the first level index and bits **11-8** representing the second level index. The page table base address is **0xE0**.

The 1st level entries are encoded as follows:

Bits 31-11	Bit 10	Bits 9-0
Unused (all 0)	Valid	Start Address of 2nd Level Page Table

The 2nd level entries are encoded as follows:

Bits 31-3	Bit 2	Bits 1-0
Unused (all 0)	Valid	Physical Page Number

2. How many entries are in the 1st level page table? _______________

3. How many entries are in a 2nd level page table? _______________

Say we execute the following LC2K program in virtual memory on our machine:

```
      lw     0     1     val   // Load val into r1
      sw     1     1     20480 // VMem[r1 + 0x5000] = r1
      add    1     1     2     // r2 = 2 * r1
      sw     2     1     20480 // VMem[r2 + 0x5000] = r1
      halt
val   .fill        1801        // 0x709 in hex
```

The below table shows a small snippet of physical memory as the program begins, from 0xE0 to 0x10F.

Addr	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0x0E0	430	0FC	164	2C3	106	4F0	387	12D	077	26d	1BA	324	013	3B8	337	3EE
0x0F0	003	002	000	002	001	003	002	005	003	001	000	003	002	003	001	002
0x100	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000

4. For the first store, what is the destination virtual address?
   - ◯ 0x0709
   - ◯ 0x5000
   - ◯ 0x5709
   - ◯ 0x6801
5. For the first store, what is first level page table index?
   - ◯ 0x2
   - ◯ 0x5
   - ◯ 0x6
   - ◯ 0xA
6. For the first store, what are the initial contents of the second level page table entry?
   - ◯ 0x1
   - ◯ 0x3
   - ◯ 0x5
   - ◯ Cannot be determined from the given information
7. For the first store, what is the final destination physical address?
   - ◯ 0x009
   - ◯ 0x109
   - ◯ 0x209
   - ◯ 0x309

(Snippet repeated for ease.)

Addr	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0x0E0	430	0FC	164	2C3	106	4F0	387	12D	077	26d	1BA	324	013	3B8	337	3EE
0x0F0	003	002	000	002	001	003	002	005	003	001	000	003	002	003	001	002
0x100	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000

8. For the second store, what is the destination virtual address?
   - ◯ 0x5E0E
   - ◯ 0x5E12
   - ◯ 0x5E18
   - ◯ 0x8602
9. For the second store, what is the first level page table index?
   - ◯ 0x2
   - ◯ 0x4
   - ◯ 0x5
   - ◯ 0xA
10. For the second store, what are the initial contents of the second level page table entry?
    - ◯ 0x1
    - ◯ 0x3
    - ◯ 0x5
    - ◯ Cannot be determined from the given information
11. For the second store, what is the final destination physical address?
    - ◯ 0x112
    - ◯ 0x212
    - ◯ 0x312
    - ◯ Cannot be determined from the given information
12. The store instructions in the program will result in exactly 2 updates to the memory values **in the above snippet**. Since the machine code is valid in physical memory, you may **ignore page table updates from the load, and from instruction fetches**. For each value in the above table that is modified, write below the physical address receiving the update and the new value. Put all values in Hexadecimal.

Physical Address (in Hex, in the range 0xE0 to 0x10F)	New Value (in Hex)

Problem 6: Cost-Benefit Analysis 25 points

You are doing an internship as a computer architect and were asked to evaluate a set of architectural trade-offs. They want to implement the features that provide the best performance-cost trade-off for their system. Your task is to predict the benchmarks' performance and determine which configurations are optimal for their applications.

This baseline in-order pipelined processor was designed to have no stalls for data hazards of any type. You are given a simulator and collect the following measurements:

Total instructions	**35,000,000**
Memory instructions	16,000,000
Branch instructions	7,100,000
I-cache hit rate	98%
D-cache hit rate	91%
Branch predictor accuracy rate	92%

Unfortunately, the simulator they gave you did not compute cycles or CPI. Your first task is to figure these out for the two benchmarks. After some digging, you will find the following facts from the simulator's configuration parameters:

- **The branch misprediction penalty is 7 cycles**
- **I and D cache miss penalties are 20 cycles**

1. You first compute the cycles spent on misprediction and miss penalties for the benchmark.

Branch misprediction cycles	
I-cache miss cycles	
D-cache miss cycles	

2. Then, produce a summary of total cycles and CPI (rounded to 2 decimal places). You may ignore the few cycles to empty or fill the pipeline.

Total cycles	
CPI	

(Original metrics repeated for ease)

Total instructions	**35,000,000**
Memory instructions	16,000,000
Branch instructions	7,100,000
I-cache hit rate	98%
D-cache hit rate	91%
Branch predictor accuracy rate	92%

- **The branch misprediction penalty is 7 cycles**
- **I and D cache miss penalties are 20 cycles**

3. Your manager asks you to decide between two features, each adding equal processor area:
   - A branch predictor that increases prediction accuracy to 95%
   - A better data cache, which increases the D-cache hit rate to 96%

Each feature adds about 25% to the processor area. Since the company doesn't want to add too much extra cost to the processor (and cost is proportional to area), they would like to pick between the two options. You start eyeballing the benchmark breakdown from before and quickly realize that the choice is clear.

Choose the best and fill in the bubble:	◯ Better D-cache	◯ Better branch predictor

Please explain your reasoning in 1-2 sentences. You do not need to make specific calculations, but your answer should include comparisons between relevant metrics.

____

____

____

____

____

4. The better D-cache and branch predictor were deemed too expensive for the product, so you are asked to evaluate another idea instead. This new idea is an accelerator engine that can accelerate certain sets of instructions but which adds a 10% area cost. You look at the accelerator's new custom instructions and note that using them results in the execution of 8,000,000 fewer arithmetic instructions (the number of branch and memory instructions executed stays the same).

You produce the following results:

Total instructions	**27,000,000**
Cycles Reduced from the initial architecture	

5. In a discussion with your team, it comes out that a decision has to be made between the initial architecture and the accelerator. The former two options were deemed to be too expensive for the product.

The main objective is to **minimize the cost** of the device, and due to system constraints, they can only run the processor at a maximum of 1.5 GHz. (1 GHz = 10^9 Hz)

They further explain that the use case they are designing this processor for is a sensor that must be able to run the benchmark at least 10 times per second. They want to leave headroom for future applications, so you **should plan** for 2x capacity (i.e., plan for enough performance headroom for the benchmark to be able to run 20 times per second).

Choose the best and fill in the bubble:	◯ Initial Architecture	◯ Accelerator

Please explain your reasoning in 1-2 sentences, using calculations to support it.

____

____

____

____

____

THIS PAGE INTENTIONALLY LEFT BLANK.