| **Final Exam**  ***\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_ \_\_\_\_ \_\_\_\_\_\_\_\_\_\_ \_\_\_***  ***| \_\_\_\_| \_\_\_\_/ \_\_\_/ \_\_\_| |\_\_\_ /\_\_\_ / \_ \***  ***| \_| | \_|| | \\_\_\_ \ |\_ \ / / | | |***  ***| |\_\_\_| |\_\_| |\_\_\_ \_\_\_) | \_\_\_) |/ /| |\_| |***  ***|\_\_\_\_\_|\_\_\_\_\_\\_\_\_\_|\_\_\_\_/ |\_\_\_\_//\_/ \\_\_\_/***  **EECS 370 Fall 2024: 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 丄 if you are at the end of the row) | | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
| Uniqname of person sitting to your ***Left***  **(**Write 丄 if you are at the end of the row) | | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |

**Exam Directions:**

* You have **120 minutes** to complete the exam. There are **7** questions in the exam on **18** pages (double-sided). **Please flip through your exam to ensure you have all pages.**
* 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 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Point Value* | 15 | 15 | 14 | 14 | 14 | 14 | 14 |

# THIS PAGE INTENTIONALLY LEFT BLANK

# 

| **1.** |  | **Multiple Choice** | **15 points** |
| --- | --- | --- | --- |
|  |  | Red are correct answers. Orange are correct answers but we did not take off points if you did not select it. | |

Completely shade the boxes with the best answer. **PROBLEMS 1-5 CONTAIN POTENTIALLY MULTIPLE CORRECT ANSWERS. SELECT ALL THAT APPLY** **[2 points each]**

1. Which of the following are registers in a pipelined datapath that we have discussed in class? **(FILL IN ALL THAT APPLY)**
   * ID/EX
   * Mem/WB
   * EX/WB
   * PC
   * FWD/WWF
2. Which of the following methods that we have discussed in class could potentially help avoid data hazards (**FILL IN ALL THAT APPLY)**
   * Detect and Stall
   * Detect and Forward
   * Detect and Predict
   * Avoidance through programming and/or compilation
3. Which of the following methods that we have discussed in class could potentially help avoid control hazards? **(FILL IN ALL THAT APPLY)**
   * Delayed Branch
   * Branch Prediction
   * Detect and Stall
   * Detect and Forward
4. Which of the following combinations are legitimate cache configurations? **(FILL IN ALL THAT APPLY)**
   * Write back + Write allocate
   * Write through + Direct mapped
   * Direct mapped + Fully associative
   * Write back + Fully associative
5. What are the general benefits of virtual memory (**FILL IN ALL THAT APPLY)**
   * Transparency (No knowledge of how other programs are using memory)
   * Caches (Programs are able to use caches)
   * Protection (Programs can’t modify each other’s data)
   * Capacity (Programs not limited by DRAM sizes)

**PROBLEMS 6-10 ONLY HAVE 1 CORRECT ANSWER [1 point each]**

1. Consider a program in which 50% of the instructions are loads and every load is immediately followed by a dependent ALU instruction. Also assume that there are no control hazards and all memory accesses complete in one cycle. What would be the CPI of this program run on the pipelined processor discussed in class if “Detect and Forward” is implemented?
   * 3/2
   * 5/2
   * 1
   * 5
2. A branch predictor is effective because branch instructions:
   * usually (but not always) follow a predictable pattern of outcomes of taken/not-taken
   * always follow a predictable pattern of outcomes of taken/not-taken
   * usually (but not always) ‘hit’ in the cache
   * always ‘hit’ in the cache
   * always resolve before write-back
3. Which one is true? Here we use FA to refer to Fully Associative cache, DM to refer to Direct Mapped cache, and SA to refer to a Set Associative cache that is neither FA nor DM:
   * SA always achieves a higher cache hit rate than DM or FA.
   * DM always achieves a higher cache hit rate than SA or FA.
   * FA always achieves a higher cache hit rate than DM or SA.
   * None of the above.
4. Which of the following cache designs (assuming equal total size and number of entries) is expected to run at the lowest clock frequency if they consume the same power?
   * Fully-associative
   * Direct-mapped
   * Set-associative
   * They can all be expected to run at the same frequency
5. How many memory accesses will each load instruction require in a system employing virtual memory and a single-level page table, but has no TLB or caches? You may ignore memory accesses involved in fetching the instruction.
   * 1
   * 2
   * 3
   * Impossible to have virtual memory without TLB or caches. Nice try!

| **2.** |  | **Short Answers** | **15 points** |
| --- | --- | --- | --- |
|  |  |  | |

1. Consider two cache designs:
2. A 1 Kilobyte 4-way associative cache with 16 byte blocks (Kilobyte = 2^10 bytes)
3. A 1 Kilobyte direct mapped cache with 16 byte blocks

Assuming the cache starts out empty, what is the smallest number of memory accesses that could result in a **hit in cache A**, but the exact same accesses would result in a **miss in cache B**. Write "N/A" if this situation is impossible **[4 points]**

(E.g. if the 3 accesses "100, 200, 100" resulted in a hit in A but a miss in B, and no shorter set of accesses did so, you would write "3"). You don't need to specify what addresses cause this behavior)

Number of accesses:

| 3 |
| --- |

2. Consider the 1-bit ‘last time’ branch predictor discussed in lecture. The predictor is initialized to "Not Taken". We observe the following execution sequence for a single branch: "T T N T N N N T". What is the accuracy of this predictor (provide your answer as a percentage)? Note: ‘T’ means Branch Taken and ‘N’ means Branch Not Taken. **[3 points]**

T T N T N N N T

N T T N T N N N

⅜ = 0.375 = 37.5%

3. Select **all** types of cache misses that apply to the following statements **[5 points]**:

| **Statement** | **Compulsory** | **Capacity** | **Conflict** |
| --- | --- | --- | --- |
| It's possible for \_\_\_ misses to be reduced by increasing the size of the cache, while keeping the block size the same. | ◯ | ◯Correct | ◯Optional |
| It's possible for \_\_\_ misses to be reduced by increasing the block size, while keeping the total cache size constant | ◯Correct | ◯Optional | ◯Optional |
| \_\_\_ misses can occur in a fully associative cache | ◯Correct | ◯Correct | ◯ |
| \_\_\_ misses can occur in a direct mapped cache | ◯Correct | ◯Correct | ◯Correct |

4. Select which type of page table (compared to the other) best describes each situation **[3 points]**:

| **Statement** | **Single-level** | **Multi-level** |
| --- | --- | --- |
| \_\_\_\_ page tables are expected to provide translations with higher access latency | ◯ | ◯Correct |
| \_\_\_\_ page tables are expected to require more memory in the worst case | ◯ | ◯Correct |
| \_\_\_\_ page tables are expected to require more memory in the typical case | ◯Correct | ◯ |

| **3.** |  | **Data Hazards** | **14 points** |
| --- | --- | --- | --- |
|  |  |  | |

Consider the program below:

lw 0 3 four // lw1 [*Line 1*]

lw 0 1 one // lw2 [*Line 2*]

start add 1 1 1 // [*Line 3*]

beq 3 1 end // beq1 [*Line 4*]

sw 0 1 100 // sw [*Line 5*]

beq 0 0 start // beq2 [*Line 6*]

end halt // [*Line 7*]

one .fill 1 // [*Line 8*]

four .fill 4 // [*Line 9*]

Recall the pipeline from lecture: **IF → ID → EX → MEM → WB**

Suppose you use a modified version of the lecture pipeline. To improve cycle time, you create a new pipeline that has two **MEM stages, MEM1 and MEM2, and two EX stages, EX1 and EX2.** Now, the pipeline is:

**IF → ID → EX1 → EX2 → MEM1→ MEM2 → WB.**

* When forwarding is allowed, data is always forwarded to **EX1**.
* When forwarding is allowed, internal forwarding from **WB** to **ID** is also present i.e. values written to a register in the WB are immediately available to reads in ID in the same cycle.
* **lw** will finish at the end of **MEM1** (i.e. when forwarding is allowed, the **lw** data can be forwarded from the next pipeline stage **MEM2**).
* **sw** will finish at the end of **MEM2**.
* **add** instructions will finish at the end of **EX2** (i.e. when forwarding is allowed, the **add** data can be forwarded from thenext pipeline stage **MEM1**).
* Branch address is computed in **EX1** and resolved (passed to PC) in **EX2**.

1. Select all registers that are involved in a RAW (Read After Write) data hazard. Assume data hazards and control hazards are both resolved using **detect and stall (i.e. no forwarding is allowed here)**. Remember, data hazards can only occur from instructions that are executed. If a register is involved in multiple hazards, list them individually. **You may not need all the lines provided below. [5 points]**

| **Register** | **Line # of producer** | **Line # of dependent consumer** |
| --- | --- | --- |
| **1** | **2** | **3** |
| **1** | **3** | **4** |
| **3** | **1** | **4** |
| **1** | **3** | **5** |
| **1** | **2** | **4** |
| **1** | **2** | **5** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

2. Using **detect and stall** to resolve data hazards, and **detect and stall** to resolve control hazards **(again, no forwarding)**, what is the number of stalls due to **data hazards**? **[4 points]**

Note: The old solution had errors and we have below the updated solution. We provide both below - we will accept both solutions. We will regrade for everyone.

New solution: 12 if internal forwarding, 15 if no internal fwd

Old solution: 16 if internal forwarding, 20 if no internal fwd

3. Using **detect and forward** **(i.e. here forwarding is allowed)** to resolve data hazards, and **predict taken for branches** to resolve control hazards. Answer the statements below.

i. How many instructions are executed to completion in this program? **[2 point]**

New solution: 9

Old solution: 13

ii. How many cycles does the program take to run? **[3 points]**

New solution: 22

Old solution: 30

| **4.** |  | **Caches Misses** | **14 points** |
| --- | --- | --- | --- |
|  |  |  | |

Consider a byte-addressable system, and a **64-byte** (excluding overheads like tags / valid bits), **4-way** set-associative cache with **8-byte** blocks. The cache uses **LRU** or Least Recently Used replacement. With LRU, the way to be evicted within a set is the one that was least recently referenced. In the table, an LRU value of 0b00 denotes the least recently used way. The cache is preloaded as shown. Assume: (a) before preloading all cache entries were invalid, (b) during preloading all blocks in Set 0 were accessed before blocks in Set 1, and no evictions occurred.



From this point, the following addresses are referenced. Determine whether each reference is a hit or a miss, and categorize the type of each miss (Compulsory / Capacity / Conflict). The given first reference is not reflected in the cache, but should be updated before completing the below.

| Reference | Tag | Set Idx | B.O | Hit/Miss | Type of Miss (N/A if hit) |
| --- | --- | --- | --- | --- | --- |
| 0x03 | 0x0 | 0 | 3 | Miss | Compulsory |
| 0x3A | 0x3 | 1 | 2 | Hit | N/A |
| 0x1B | 0x1 | 1 | 3 | Miss | Compulsory |
| 0x16 | 0x1 | 0 | 6 | Hit | N/A |
| 0x49 | 0x4 | 1 | 1 | Hit | N/A |
| 0x2C | 0x2 | 1 | 4 | Miss | conflict |
| 0x34 | 0x3 | 0 | 4 | Hit | N/A |
| 0x59 | 0x5 | 1 | 1 | Miss | Conflict |
| 0xA0 | 0xA | 0 | 0 | Hit | N/A |

| **5.** |  | **Branch Prediction** | **14 points** |
| --- | --- | --- | --- |
|  |  |  | |

Consider the following LC2K assembly program running on our 5-stage pipelined datapath from lecture. This program **counts the number of zeros** in the binary array **Arr** and stores that count in **r5**. Assume all registers are initialized to zero. Answer the following question

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | lw 0 1 One //r1 = 1  lw 0 2 Len //r2 = Len  lw 0 3 Zero //r3 = loop counter i  loop beq 3 2 end //loop terminates if i==Len  lw 3 4 Arr //load Arr[i]  beq 4 1 cont //if Arr[i]==1, skip Line 7  add 5 1 5 //increment r5  cont add 3 1 3 //increment i  beq 0 0 loop  end halt  Zero .fill 0  One .fill 1  Len .fill 6  Arr .fill 1  .fill 0  .fill 1  .fill 1  .fill 0  .fill 1 |
| --- | --- |

| 1. | Write the sequence of branch decisions for each **beq** instruction, using **T** to represent “taken” and **N** to represent “not taken.” (You might not need all the boxes.) **[8 points]** | | |
| --- | --- | --- | --- |

| Line 4 **beq** (Loop termination condition) | **N1** | **N4** | **N7** | **N10** | **N13** | **N16** | **T19** |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Line 6 **beq** (If-Condition) | **T2** | **N5** | **T8** | **T11** | **N14** | **T17** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Line 9 **beq** (Loop condition) | **T3** | **T6** | **T9** | **T12** | **T15** | **T18** |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Combined sequence, as seen globally | **N** | **T** | **T** | **N** | **N** | **T** | **N** | **T** | **T** | **N** | **T** |
|  | **T** | **N** | **N** | **T** | **N** | **T** | **T** | **T** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

| 2. | Assume that you are given a program with 3 branch instructions. The branch instructions follow an alternating sequence of “B1, B2, B3, B1, B2, B3, B1, B2, B3, …….”. The “B1 B2 B3” sequence repeats 10 times. The 3 branch instructions exhibit the following branching patterns (note that T means Taken and N means Not Taken):  **B1: T N T N T T N T N N**  **B2: T T T N N N T T T N**  **B3: N T T T T N N T T N**  What percentage of the total branch decisions are correctly predicted when using the following prediction schemes? (You can leave your answers as fractions.) **[1+2+3 = 6 points]** | |
| --- | --- | --- |
|  | Predict always taken (T) | \_\_\_\_17/30\_\_\_\_\_\_\_\_\_\_ |
|  | Predict B1 always taken (T), B2 and B3 always not taken (N) | \_\_\_13/30\_\_\_\_\_\_\_\_\_\_ |
|  | Predict using a local 2-bit branch predictor initialized to the starting state “weakly taken”. Note: by ‘local branch predictor’, we mean that each branch instruction has its own predictor (as discussed in lecture). | \_\_\_14/30\_\_\_\_\_\_\_\_\_\_\_ |

2-bit predictor = 5/10 + 5/10 + 4/10

| **6.** |  | **Page Tables: Leveling Up** | **14 points** |
| --- | --- | --- | --- |
|  |  |  | |

Consider a 48-bit byte-addressable system that supports virtual memory with the following specifications (Note KB = Kilobyte = 2^10 bytes, GB = Gigabyte = 2^30 bytes):

* A page is **8 KB**.
* The page table is hierarchical with **3 levels**.
* The first-level occupies **4 pages** of memory.
* Each second-level occupies **8 pages** of memory.
* A page table entry is **16 B** and is the same for all levels.
* **64 GB** of physical memory is installed.

You may leave your answers as exponents when appropriate **[6 points]**

1. How many bits are used for the page offset? \_\_\_\_\_13\_\_\_\_\_\_\_\_\_\_\_
2. How many virtual pages exist in this system? \_\_\_\_\_\_2^35\_\_\_\_\_\_\_\_\_
3. How many physical pages exist in this system? \_\_\_\_\_\_2^23\_\_\_\_\_\_\_\_\_\_
4. How many bits in a virtual address are used to index …
   1. the first-level page table? \_\_\_\_11\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   2. a second-level page table?  \_\_\_\_12\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. a third-level page table? \_\_\_\_12\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. How much memory does this page table system occupy in the worst case? Express your final answer **in bytes** and in powers of 2. Do not simplify (e.g. if your answer is 2x + 2y, leave it at that). **[3 points]**

1st PT = 2^15 B and 2^11 entries

2nd PT = 2^11\*2^16 = 2^27 B and 2^11\*2^12 = 2^23 entries

3rd PT = 2^23\*2^16 = 2^39 B

Total size = 2^15 + 2^27 + 2^39 Bytes

1. If this system used a single-level page table instead of a three-level page table (each page table entry is still 16 bytes), how much memory would it occupy? Express your final answer in bytes and as a sum of powers of 2. Do not simplify (e.g. if your answer is 2x + 2y, leave it at that). **[3 points]**

2^35 entries, so 2^39 Bytes.

1. What specific situation must occur in order for the single-level page table to use less memory than the three-level page table? Reference specific values in your answer. Limit your answer to 2 sentences. **[2 points]**

Almost all or all 3rd level tables must be allocated.

| **7.** |  | **Reads and Writes to Memory** | **14 points** |
| --- | --- | --- | --- |
|  |  |  | |

You are required to run a program that generates 1 Billion loads and 3 Billion stores, each accessing data that is 8 bytes in size. Your system can use a data cache with a 64-byte block which gets a 90% hit rate on that program (with some more specifications which are detailed below). Note that we are not concerned with instruction fetches or an instruction cache in this question.

1. How many bytes would be read from memory and written to memory if you did not use the data cache (write down each separately)? **[3 points]**

8 billion bytes read (1Billion \* 8 bytes)

24 billion bytes written (3Bil \* 8 bytes)

Total = 32 billion Bytes

2. How many bytes of memory would be read from memory and written to memory if you had a **write-through** data cache with a **no-write allocate** policy (write down each separately)? **[3 points]**

Reads occur only on load misses and will be at the granularity of a cache block:

1Billion \* 10% \* 64-bytes = 6.4 billion Bytes

Writes occur on all stores and will be at the granularity of a store

3 Billion \* 8 bytes = 24 billion Bytes

Total = 30.4 billion Bytes

3. How many bytes of memory would be read from memory and written to memory if you had a **write-back** data cache with a **write-allocate** policy (write down each separately)? Assume 20% of all misses result in a dirty eviction. **[3 points]**

Reads occur on all cache misses both loads and stores, at granularity of block:

4Billion\*10%\*64B = 25.6B

Writes occur only on write back, at granularity of block:

4Billion\*10%\*20%\*64B = 5.12B

Total = 30.72 billion Bytes

4. You are allowed to make one of the two changes to the **write back** data cache (with **write-allocate**) to reduce the total bytes transferred between the cache and memory (i.e. total memory accesses). You could a) improve the data cache hit rate to 95%, or b) Reduce the dirty eviction rate on stores to 5% (loads remain at 20%). Which would you choose and why? Show calculations comparing the two changes. **[5 points]**

(a) 95% hit rate:

Total memory access in bytes = 4Billion \* 64Bytes \* 0.05 \*(1 + 0.2) = 15.36 Billion bytes

(b) Dirty eviction of stores is 5%:

Total memory access in bytes = 4Billion \* 64Bytes \* 0.1 + 1Billion\* 64Bytes\*10%\*20% + 3Billion \* 64Bytes\*10%\*5% = 25.6 + 1.28 + 0.96 = 27.84 Billion bytes.

We would choose (a) because lower memory accesses in bytes.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK