WASHINGTON STATE UNIVERSITY

EE524/CPTS561 Mid term exam

Friday, 9 October 2020. Submit by 8:00 am, 10 October 2020

Name: **ID:**

FOR FULL CREDIT:

- Please show your work, justify the claims you make and point to a source if you are using any. You may lose points if you use a resource without appropriate references.
- Please, **DO NOT** collaborate on your answers. You should work on the exam questions by yourself.

Please type or hand-write your answers and upload them to Blackboard. Please convert your files or images to **PDF** format and upload the PDF. It makes annotations and returning feedback to you much easier.

Submit your exams by 8:00 am on Saturday, 10 October 2020. Late submissions will be penalized at 10 points/hour for each hour of delay.

Question 1 [10/20 points]

You are given the task to choose the number of cores in the next generation of a multicore chip. Your teammates collected the data summarized below:

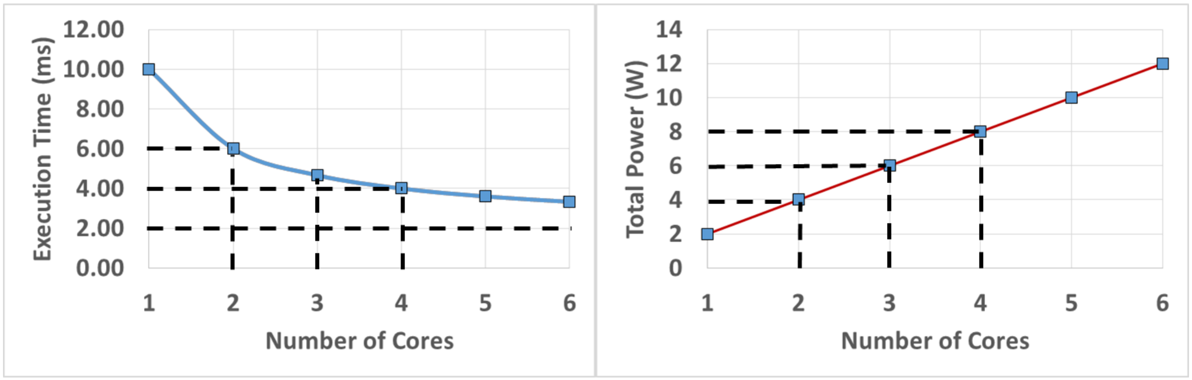


Figure 1: Execution time and power consumption with respect to the number of cores

Looking at this data, assume the following:

- The power consumption can be expressed as: $P(n) = nP(1)$, where $P(1)$ is the total power consumption of a single core. Further assume a fixed C_{dyn} and $I_{leakage}$ throughout this problem, and operating voltage V is proportional to the frequency (i.e., you can assume $V = af$, where $a > 0$).
- The execution time can be expressed as: $t_{exe} = t_c + \frac{t_s}{n}$, where t_c is a constant term which does not change with the number of cores, and $\frac{t_s}{n}$ denotes the time that scales with number of cores. Both t_c and t_s are inversely proportional with frequency.

(a) What is the **energy-optimal** number of cores? Write *one-sentence* intuition for this answer. Recall that energy is the product of power and execution time.

(b) Suppose that 10 ms execution time provides sufficient performance for your target market. Propose a design technique to reduce the energy consumption compared to the minimum energy you found in part (a). You need to quantify the new energy consumption and show that it is smaller than the result in part (a). You can assume that $P_{dynamic} > P_{leakage}$.

Question 2 [5/7/7/6 points]

As part of optimizing the cache design, you recently switched to using a two-way set associative cache, as opposed to a direct-mapped cache. However, adding associativity to a cache increases the access time. One of the ways to reduce the access time is to use a way predicting cache. On every cache access, we make a prediction about the way in which the data is present and select the way. If the way is correctly predicted, it generates a fast access, similar to a direct-mapped cache. If the prediction is incorrect, the other way is accessed to check if the data is present in the second way. If the data is not present in the cache, then a normal cache miss occurs. This process is described in the figure below:

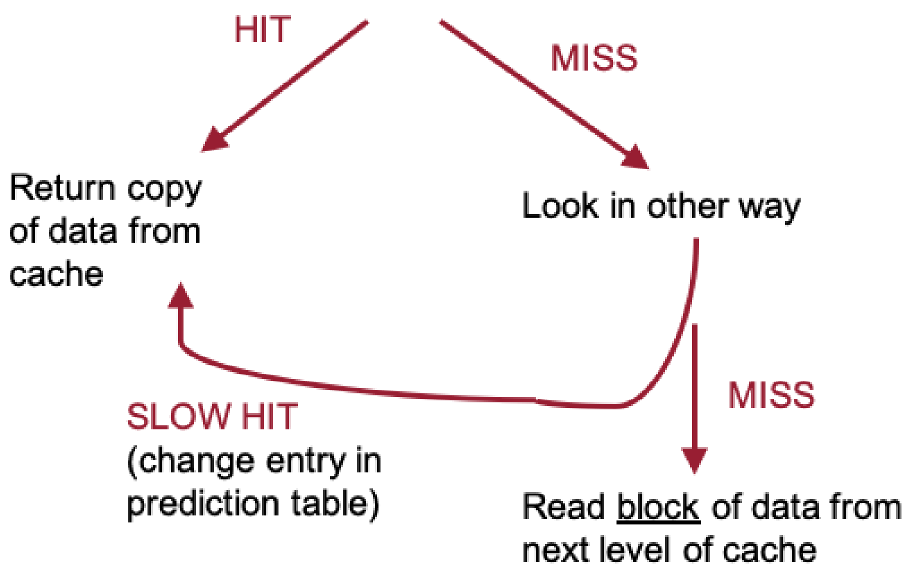

HIT
MISS
Return copy of data from cache
Look in other way
SLOW HIT (change entry in prediction table)
MISS
Read block of data from next level of cache

Figure 2: Way prediction scheme

The way prediction consumes one bit, since there are two ways present in the cache. Moreover, the bit value is used as the prediction. For this question, you can assume that a prediction is always available on a cache access, and it is not part of the critical path of the cache access.

(a) First, we will determine the cache parameters for our 2-way set associative cache. Assuming a 64-bit address, 64-byte blocks, and 12-bit cache index, complete the following table. You can use the next page to show your work.

Table 1: Cache parameters

	2-way set associative cache
Tag bits	
Block offset bits	
No. of sets	
Cache size (KB)	

We will use these cache parameters for the rest of this question.

(b) Next, we will determine the cache hit time by looking at the critical path in the cache hit. The critical path is the sequence of steps that have to be done for a fast access. For this purpose, we will use the figure below that shows the architecture of the way predicting cache.

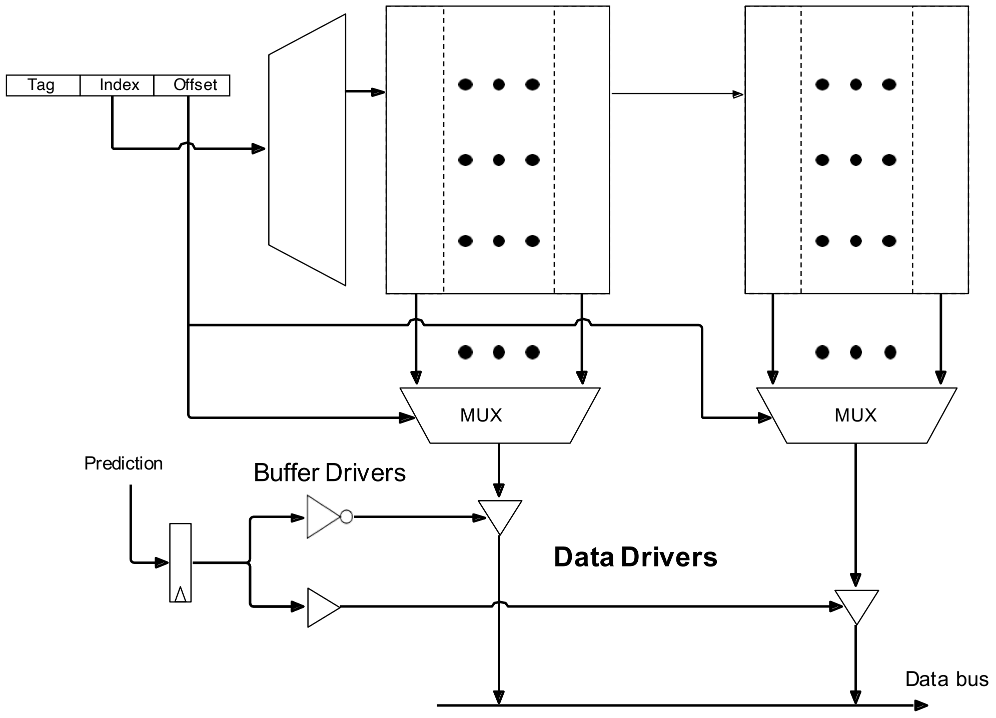


Figure 3: Architecture of way-predicting cache

Using the figure and the table below, calculate the delay of the critical path. You can assume that the prediction bit generation is not on the critical path as it is readily available for every cache access. (*Hint: To determine the critical path, you need to check the sequence of operations that have to be done for a fast cache access. Modules not in the direct path do not contribute to the critical path delay.*)

Table 2: Critical path calculation

Component	Equation for delay (ps)	Total (ps)
Decoder	$20\cdot(\#\text{ of index bits}) + 60$	
Memory array	$10\cdot \log_2(\#\text{rows}) + 10\cdot \log_2(\#\text{ bits in each row}) + 200$	
N to 1 Mux	$50\cdot \log_2 N + 200$	
Buffer driver	160	
Data output driver	$90\cdot(\text{associativity}) + 100$	
Critical Path Delay		

(c) Way-predicting caches can potentially offer advantages in instruction caches. Assume a prediction scheme where each set has a prediction bit that keeps track of the last way accessed for the set. Next, assume that the cache uses a least recently used replacement policy and the parameters in Table 1 whenever there are conflicts. In what situations would the predictions would be accurate? Under what conditions will the way prediction scheme result in mispredictions?

(d) For this part, assume that the page size is 8 KB for the machine on which the way-prective cache is implemented. The cache parameters remain the same as in Table 1. If the cache is physically-tagged and virtually-indexed, does the way-predicting cache suffer from aliasing? Why or why not? Explain your answer.

Question 3 [10 points]

You are trying to optimize the classical five-stage pipeline used in your company. The current pipeline uses forwarding to avoid stalls as much as possible. After doing some profiling, you discover that the ALU stage of the pipeline is taking the longest to execute, which slows down the clock. As a result, the execution time of the applications is higher. In order to address this issue, you propose to split the ALU into two pipeline stages. With the ALU split into two pipeline stages, you now have a six-stage pipeline. Note that the result of the ALU is **only** available after the second ALU stage and the data after the first ALU state is not useful. Describe how this change of adding a pipeline stage affects Read after write (RAW) hazards. How would you resolve the new complications for RAW hazards that result from the six-stage pipeline? Please list changes and/or hardware needed for resolving the RAW hazards. You may use two consecutive ALU instructions with a RAW hazard for the purposes of this question.

Question 4 (Short answer questions) [20 points]

(a) Amdahl's law provides a very useful metric to measure speedup when we make applications parallel using more resources. However, Amdahl's law's basic form makes an assumption about the application that may not hold in real-world scenarios. State the assumption made by Amdahl's law and describe conditions that may break the assumption.

(b) What type of misses are reduced by each of these cache optimization techniques. List *all* types of misses reduced for full credit. In addition, list the possible disadvantages of using the optimization technique.

- Data and instruction prefetching:

- Pipelined cache accesses:

- Higher associativity:

- Larger cache capacity:

(c) How does each of these changes in an ISA or pipelining change the number of instructions in a program and the cycles per instruction. Provide reasoning for your answers.

- Changing the ISA from load-store to register-memory.

- Merging pipeline stages.

- Using branch instructions for jumps instead of having dedicated jump instructions.

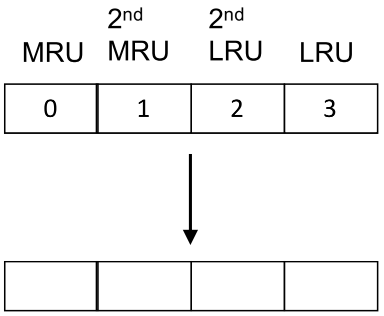
- Increasing the number of registers from 32 to 64.

(d) Many processors optimize their interrupt handling performance by taking only the bare minimum action on an interrupt and queuing the bulk of processing for a later time. Can the same optimization apply to exceptions? Explain your reasoning.

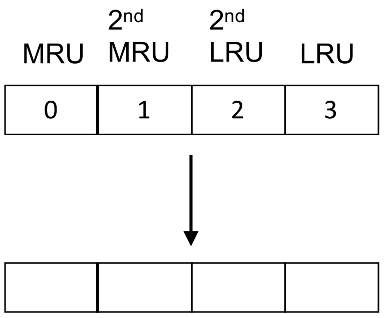
Question 5 [15 points]

Consider a four-way set associative cache for the following questions. We explore different replacement and insertion policies with this cache.

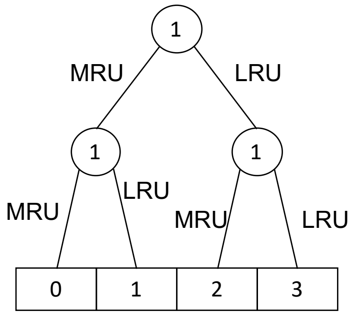
(a) To implement a least recently used (LRU) replacement policy, we form an array of 4 indices for each set to represent the line reference orders of the set. Assume the initial reference order of a particular set is 0, 1, 2, and 3 (lines 0, 1, 2, and 3 are most recently used (MRU), 2nd MRU, 2nd LRU, and LRU, respectively), as shown in the diagram. Moreover, whenever a cache miss happens, the new line is inserted in the most recently used position. After a miss, a hit to line 1, and then a hit to line 0, what is the new reference order? (Please fill in the up-to-date array of indices)



(b) Next, we make a modification to the insertion policy of the cache. Instead of inserting the new line to the MRU position, we insert it to the LRU position. It is promoted to the MRU position *only when* it is reused. Given this insertion policy, what is the reference order after 2 misses, hit to line 3, and hit to line 1.



(c) A Partial LRU (or Pseudo LRU) replacement algorithm can be described with a binary tree. That is, we form a binary tree of 4 leaf nodes for each set to represent the 4 lines in the set. In the non-leaf node, a binary bit is used to show the LRU or MRU order of the two branches (if the left branch is LRU (MRU), the bit is 0 (1)). Assume the initial reference order of a particular set is (1, 1, 1), i.e., the 3 bits at nonleaf nodes A, B, and C of the figure are 1, 1, and 1 respectively, which indicates the pair of lines 0 and 1 is accessed more recently than the pair of lines 2 and 3, line 0 is accessed more recently than line 1, and line 2 is accessed more recently than line 3.



After a miss, a hit to line 1, and then a hit to line 0, what is the new reference order in the tree:

(A, B, C) = (,   ,   )