CE-4
August 2017 QE

(1) (24 points) **Processor [13 minutes]**

(a) (12 points) In a processor with out-of-order issue, register renaming, branch prediction, and precise interrupts, give **two** reasons why a physical register mapped to an architectural destination register of instruction *A* cannot be released (to be remapped to another architectural register) immediately upon completing instruction *B, given* that B is the last use of *A*'s destination register in program order.

(b) (12 points)

(i) Many branch prediction schemes use a branch history register to hold previous branch outcomes. What is the problem for branches occurring in close succession in the instruction stream when branch resolution (i.e., knowing the outcome) occurs later in the pipeline?

(ii) How do modern pipelines solve this problem?

(2) (24 points) **Memory hierarchy [13 minutes]**

(a) (12 points) Consider a cache and memory where increasing parallelism -- the number of concurrent in-flight requests that can be handled **(x)** -- **increases** the latency of hits and misses. Specifically, the latency of cache hits and misses are given by:

$$Time_{Hit} = 10ns + x^2$$

$$Time_{Miss} = 190ns + x^2$$

Given a 50% cache hit rate, how many concurrent requests maximizes **throughput**?
Hint: Recall the quotient rule

$$f(x) = \frac{g(x)}{h(x)} \qquad f'(x) = \frac{g'(x)h(x) - h'(x)g(x)}{h(x)^2}$$

(b) (12 points) Modern operating systems provide support for multiple page sizes to be used concurrently within an application (e.g., 4-KB regular pages and 4-MB super pages).

(i) What is the problem for TLBs in supporting two page sizes simultaneously?

(ii) What is a potential solution for the problem (restricted to two page sizes)?

(3) (22 points) **Multicore [12 minutes]**

Consider a sequentially consistent (SC) system with an invalidate-based coherence protocol. Assume that before the following code segment runs X=Y=0.

Thread 1	Thread 2
X <= 10 Print Y	Y <= 10 Print X

To improve performance, an out-of-order-issue processor issues and completes **loads that are later in program order** (i.e., the memory value is loaded into the register file) even though **earlier stores in program order** have not issued to the memory system (i.e., the stores have not been sent to the L1 cache).

(a) (4 points) Does this out-of-order issue maintain SC? **Explain why.**
(b) (18 points) If not, show the violation(s) of SC for the above example. Your answer should (a) provide **all** outputs that are impossible under SC but possible under the above scheme, and (b) explain the sequences of actions that lead to the violations.

(4) (30 points) **Fundamentals [22 minutes]**

You are required to design a new architecture for a new **sequential** application. You are to use a new CMOS-like technology that affords high clock speeds and billions of transistors but provides especially slow and power-hungry wires. The architecture employs a three-level cache hierarchy with appropriate latencies. The latency and energy costs of moving data from level *i* to level *i-1* are **significantly higher** than those of accessing the level i. The new application has extremely phased memory behavior. During some phases, the working set fits entirely in the L1. In other phases, the working set fits entirely in the L2 (while thrashing the L1), while in other phases the working set fits in the L3 and the L2 is thrashed. The ordering of the phases is **entirely arbitrary** as to in which cache the next phase would fit (an example ordering could be L2, L1, L3, L1, L2, L3, L2, L3, L1, etc). Further, the phases are fairly **long**. The software **cannot** be changed.

(a) (6 points) Describe a basic architecture to improve energy and performance for the given workload. Provide only an **organizational description of the hardware**. How the software runs on the hardware comes next.
(b) (6 points) As in CMOS, slower logic is more energy-efficient in this technology. Describe a key feature of the processor architecture to exploit this fact.
(c) (18 points) Explain the key actions needed to implement your **execution strategy** for the given workload:
    (i) (10 points) As execution proceeds, what key events would you look for? How would you detect those events? Be specific. Keep in mind the arbitrary ordering of the phases. A simple scheme that is somewhat inefficient is acceptable.
    (ii) (8 points) If the caches are write-back and inclusive, what correctness issues may arise? How would you solve them?

Write in Exam Book Only