2012/Fall Computer Architecture PhD Qualifying Exam

Student ID _____ Name _____

1. [20pts] Pipelining
   A. What are the CPI of a single-cycle processor, which can execute an instruction every cycle, and the CPI of an ideally pipelined processor? (State the reason for the answer shortly)
   B. What is the speedup of the ideally pipelined processor compared to the single-cycle design? You must add a factor to answer the question. State the missing factor and explain the reason using the execution time decomposition (Execution time = instruction count * CPI * clock cycle time)
   C. Explain two reasons that the clock cycle time of a pipelined processor can be longer than the ideally pipelined processor.
   D. Explain any possible reasons that the CPI of a pipelined processor can be longer than the ideally pipelined processor.

2. [20pts] Suppose you are comparing two processors with different cache designs. Both processors have two-level caches (L1 and L2 caches). Assume a perfect instruction L1 cache. The cache configurations of the two processors are as follows:

   Processor A: L1 cache: 16KB, 1 cycle hit latency, L2 cache: 512KB, 10 cycle hit latency,
   Processor B: L1 cache: 64KB, 3 cycle hit latency, L2 cache: 2MB, 12 cycle hit latency

   For both processors, external memory access latency is 100 cycles. Suppose you will run only two applications with the following characteristics

   Application 1 running on processor A: L1 miss rate = 20%, L2 miss rate = 50%
   Application 2 running on processor A: L1 miss rate = 2%, L2 miss rate = 10%
   Application 1 running on processor B: L1 miss rate = 10%, L2 miss rate = 20%
   Application 2 running on processor B: L1 miss rate = 1%, L2 miss rate= 5%
   Assume that you use the two applications equally. (Each application is used for 50% of the total system run time) Which processor will you choose? Explain the reason with quantitative analysis.

3. [20pts] The problem assumes the following system configurations
   32-bit architecture, which uses 4GB address space for each process
   Page size: 4KB page, the size of each page table entry : 4B

   A. What is the page table size for each process for one-level page table (flat page table)?
   B. What are the minimum and maximum page table sizes for a process, if the system uses two-level page tables? (For the minimum case, a process uses the memory which fits in one page.)
   C. To hide the access latency for TLBs, TLBs may be accessed in parallel with accesses to L1 caches. To support such parallel accesses to TLBs and L1 caches, the organization of L1 caches may be restricted to meet certain conditions. If the maximum associativity is limited to 8 ways, what is the largest cache capacity for such L1 caches? (a physical address must be mapped to only one set in the cache)
   D. Suppose the maximum capacity from the 4.C was X bytes. What if you want to have a cache with 2X bytes capacity? You still want to access TLBs and L1 caches in parallel. What problems will occur and how will you solve the problem?

4. [20pts] Dependency and scheduling for out-of-order processors

```
X1:  add r1, r2, r3        ; r1   r2 + r3
X2:  or  r3, r2, r1        ; r3   r2 + r1
X3:  sub r2, r4, r5
X4:  sw  r2, 0(r6)         ; MEM[r6]   r2
X5:  and r2, r2, r7
X6:  lw  r4, 0(r7)         ; r4   MEM[r7]
```

   A. Find all possible dependencies (RAW, WAR, WAW) from the above instruction sequence.

   B. What is the minimum number of cycles to run the instructions with an ideal out-of-order processor? The execution of an instruction takes one cycle, and the unlimited number of instructions can be executed in parallel, if the instructions do not have dependencies. However, register renaming is not supported. Show your instruction scheduling with a directed graph

   C. What is the minimum number of cycles if unlimited register renaming is supported? Use the same assumption as 1-B. Show your instruction scheduling with a directed graph

5. [20pts] The following code fragment is a spin lock implementation written by a newly hired programmer. It spins until a variable (pointed by R1) becomes zero, and if the variable is zero, it is locked by setting it to 1.

Lock acquire:

```
          li R2,#1
lockit:   lw R3,0(R1)
          bnez R3,lockit
          sw R2,0(R1)
```

   A. Suppose a thread running on Core 0 is in the critical section protected by a spin lock (it is holding the lock). If another thread running on Core 1 tries to acquire the same lock, what instructions is Core 1 continuously executing until the lock becomes free. Is Core 1 doing any useful computation?

   B. Explain why the above code fragment does not implement a spin lock correctly. Describe a scenario when the above implementation can allow multiple threads to enter the critical section

   C. Another programmer fixed the above bug, and implemented a new spin lock with atomic swap instruction (exch) . Describe a scenario why the above spin lock can lead to a performance problem by explaining how the spin lock execution interacts with cache coherence. Re-write the code to fix the performance problem.

```
          li R2,#1
lockit:   exch R2,0(R1) ;atomic exchange
          bnez R2,lockit
```