

COMPUTER ARCHITECTURE & SYSTEMS—Ph.D. Qualifying Exam Fall 2009

PRELIMINARY: Questions 1, 2, and 3 refer to the following two programs, which both do the same thing—the second program is just a modified (“loop-unrolled”) version of the first.

r1: points to the beginning of 2048x2048 image (4MB image, 1-byte pixels)
r2: points just past the end of 2048x2048 image (4MB image, 1-byte pixels)
r3: points to the beginning of a 256-entry array of 4-byte integers: 1KB total,
which is located right after the end of the image

PROGRAM FOO:

```
top:  lb    r4, r1, 0    // load byte into r4 from memory location (r1+0)
      addi  r1, r1, 1    // increment pointer in r1 by 1
      muli  r4, r4, 4    // r4 <- r4 * 4
      add   r4, r4, r3   // r4 <- r4 + r3
      lw    r5, r4, 0    // load integer into r5 from memory location (r4+0)
      addi  r5, r5, 1    // increment r5 by 1
      sw    r5, r4, 0    // store integer from r5 into memory location (r4+0)
      bne  r1, r2, top   // branch to "top" if r1 pointer is still valid
      exit                // terminate program if r1 pointer is beyond end
```

PROGRAM BAR:

```
top:  lb    r4, r1, 0    // load byte into r4 from memory location (r1+0)
      lb    r5, r1, 1    // load byte into r5 from memory location (r1+1)
      addi  r1, r1, 1    // increment pointer in r1 by 2
      muli  r4, r4, 4    // r4 <- r4 * 4
      muli  r5, r5, 4    // r5 <- r5 * 4
      add   r4, r4, r3   // r4 <- r4 + r3
      add   r5, r5, r3   // r5 <- r5 + r3
      lw    r6, r4, 0    // load integer into r6 from memory location (r4+0)
      lw    r7, r5, 0    // load integer into r7 from memory location (r5+0)
      addi  r6, r6, 1    // increment r6 by 1
      addi  r7, r7, 1    // increment r7 by 1
      sw    r6, r4, 0    // store integer from r6 into memory location (r4+0)
      sw    r7, r5, 0    // store integer from r7 into memory location (r5+0)
      bne  r1, r2, top   // branch to "top" if r1 pointer is still valid
      exit                // terminate program if r1 pointer is beyond end
```

(i). Application Behavior & Performance (2 questions, 6 points)

A. (3pts) What do programs FOO and BAR do (i.e., what function/operation do they perform)?

B. (3pts) Compare the performance of the two loops (i.e., “A is X% faster than B”), as measured on a single-cycle, non-pipelined machine (one in which all instructions take 1 cycle).

(ii). Pipelines (2 questions, 7 points)

A. (2pts) What is pipelining? What is the benefit of pipelining a microprocessor?

B. (5pts) Calculate the performance of FOO and BAR on a pipelined processor running at 1GHz (1ns cycle time). **All operations** have a latency of **2 cycles**—there must be *at least one* intervening cycle between any instruction and a following dependent instruction. You may assume perfect branch prediction (i.e. branches take one cycle, not two). Give the running times for the two programs in the *time* required to execute. You may ignore memory access and the cost of the “exit” instruction at the end of execution.

(iii). Caches (2 questions, 7 points)

A. (2pts) Illustrate the workings of a processor cache by drawing a diagram of a direct-mapped cache, including the tag and data arrays and the input address. Be sure to indicate set selection, tag match, word selection, and cache hit/miss determination.

B. (5pts) Find the average data-cache hit rates for FOO vs. the following cache organizations:

- (1) 1MB, 32B block size, direct-mapped
- (2) 1MB, 64B block size, direct-mapped
- (3) 1MB, 32B block size, 2-way set associative, LRU managed