MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.004 Computation Structures
Spring 2009

Quiz #4: April 24, 2009

Name	*Athena login name*	*Score*
Solutions		Avg: 16.5

NOTE: Reference material appears on the backs of quiz pages.

Problem 1 (7 points): **Quickies and Trickies**

(A) (1 point) A Beta processor has an interrupt handler invoked by a periodic 60Hz clock. The handler simply inspects the high-order bit of the XP register to see if it is a 1 or 0. Give your best estimate of the fraction of the time it finds a 0.

Circle best answer: 0% … 50% … [✓ 100%]

(B) (2 points) An application program that stores data in the XP register might fail due to (circle YES or NO for each):

Interrupt handler returns to wrong location: YES .. [✓ NO]

Application data in XP changes unexpectedly: [✓ YES] … NO

Interrupt executes wrong handler: YES .. [✓ NO]

(C) (2 points) Decrementing the saved PC of an interrupted program often indicates (circle YES or NO for each):

re-entrant interrupts: YES .. [✓ NO]

a programming error: YES .. [✓ NO]

a cache miss: YES .. [✓ NO]

"busy waiting": [✓ YES] .. NO

(D) (2 points) Use of a Translation Lookaside Buffer (circle YES or NO for each):

decreases cache hit rate: YES .. [✓ NO]

decreases average address translation time: [✓ YES] .. NO

prevents handler re-entrance: YES .. [✓ NO]

Problem 2 (7 points): **Cache Management**

Four otherwise identical Beta systems have slightly different cache configurations. Each cache has a total of 8 lines each caching a single 32-bit data word, and caches both instruction and data fetches. However, the caches differ in their associativity as follows:

Cache C1: Direct mapped, 8-word cache.
Cache C2: 2-way set associative (4 sets of 2 lines), LRU replacement.
Cache C3: Fully associative, LRU replacement.

Your task is to answer questions about the performance, measured by hit ratio, of these cache designs on the following tiny benchmarks. Note that each benchmark involves instruction fetches starting at location **0** and data accesses in the neighborhood of location **1024 (= 2^{10})**.

```
 .=0    || Benchmark B0
        CMOVE(100, R1)
 LOOP:  LD(R31, 1024, R0)
        SUBC(R1, 1, R1)
        BNE(R1, LOOP)
        HALT()
```

```
 .=0    || Benchmark B1
        CMOVE(100, R1)
 LOOP:  LD(R31, 1024+4, R0)
        SUBC(R1, 1, R1)
        BNE(R1, LOOP)
        HALT()
```

```
 .=0    || Benchmark B2
        CMOVE(100, R1)
 LOOP:  LD(R31, 1024+4, R0)
        LD(R31, 1024+8, R0)
        SUBC(R1, 1, R1)
        BNE(R1, LOOP)
        HALT()
```

```
 .=0    || Benchmark B3
        CMOVE(100, R1)
 LOOP:  LD(R31, 1024+4, R0)
        LD(R31, 1024+8, R0)
        LD(R31, 1024+12, R0)
        SUBC(R1, 1, R1)
        BNE(R1, LOOP)
        HALT()
```

```
 .=0    || Benchmark B4
        CMOVE(100, R1)
 LOOP:  LD(R31, 1024+4, R0)
        LD(R31, 1024+8, R0)
        LD(R31, 1024+12, R0)
        LD(R31, 1024+16, R0)
        SUBC(R1, 1, R1)
        BNE(R1, LOOP)
        HALT()
```

(A) (1 point) Which benchmark yields the best hit ratio with cache **C1**?

(circle one): [✓ B0] … B1 … B2 … B3 … B4

(B) (2 points) Select the value that best approximates the hit ratio with cache **C1** on Benchmark **B1**.

(select approx hit ratio): 0% … 25% … [✓ 50%] … 75% … 100%

(C) (2 points) Which cache yields the best hit ratio with benchmark **B3**?

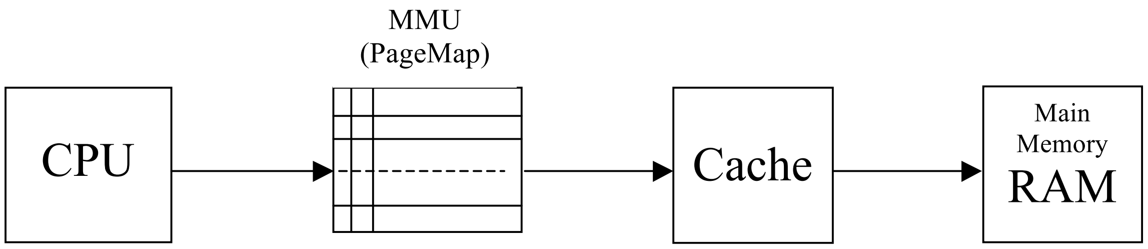
(circle one): C1 … C2 … [✓ C3]

(D) (2 points) Which cache, if any, yields a hit ratio of **zero** (0%) with benchmark **B4**?

(circle one): C1 … C2 … [✓ C3] … NONE

Problem 3 (8 points)**: Memory Systems**

A simple computer system using the Beta processor has memory components as diagrammed below:



Some parameters of the system include:

Cache design: Direct mapped, 4-byte (1 word) block size, **2^{14} words** total data. Cache access (hit) time: **5 ns**
MMU design: single-level PageMap, 2^{30}-byte virtual address space, 2^{28}-byte physical address space, page size 2^{14} bytes. MMU access (translation) time: **5ns.**
RAM (main memory) access (read or write) **time**: **40 ns.**

Assume that other times (e.g. gate delays) are insignificant compared to memory access times. Note that **all memory accesses** — including **LD**s and **ST**s as well as instruction fetches — are made via the above path.

(A) (1 point) Assume that each page map entry includes **Resident** and **Dirty** bits as well as a physical page number. What is the total size, in bits, of the pagemap storage?

Total page map size (bits): $2^{20} = 16 \times 2^{16}$

(B) (2 points) What is the memory access time for a cache hit? A miss?

Access times on HIT: 10 **ns; on MISS:** 50 **ns**

(C) (1 point) Assuming a 90% hit rate, what is the expected average access time?

Average access time for 90% hit rate: 14 **ns**

(D) (2 points) Which, if any, of the following operations **must** the operating system perform on a context switch? **Circle YES or NO for each:**

Clear cache Valid bits? YES … [✓ NO]

Reload PageMap with new context? [✓ YES] … NO

Read pages from disk corresponding to new virtual memory contents? YES … [✓ NO]

Noah Doze, one of the few who actually stayed awake during the entire VM lecture in 6.004, argues that a clever redesign of the memory system using a K-way set associative cache of the same total (**2^{14}-word**) size, for some carefully chosen K, will allow the cache and page map lookups to happen concurrently.

(E) (2 points) Fill in the details of Noah's argument in the blanks below. Continue to assume a 90% hit ratio.

Appropriate (minimal) value of K: 4

New average access time: 9 **ns.**

Problem 4 (3 points): **Mystery Handler**

You are exploring an operating system very much like the one sketched in lecture, and have discovered the following code for a mysterious, undocumented supervisor call handler:

```
Mystery_SVC_handler()
{
    User.Regs[0] = User.Regs[0] – 1;

    if (User.Regs[0] != 0)
    {
        User.Regs[XP] = User.Regs[XP] – 4;
        Scheduler();
    }
}
```

Recall that the C operator `!=` means "not equal".

Your task is to provide documentation explaining, to an application developer, the impact of executing this SVC on a running application and on other processes running on the system.

(A) (3 points) Describe, in few sentences, the effect of executing this mystery SVC by an application. Be sure to mention the role of and impact on any relevant register contents.

This SVC causes the current process to "sleep" (not execute any further instructions) for the next R0 - 1 scheduler cycles. R0 will be set to 0 when the SVC completes. If R0 is 1, it has no effect beyond setting R0 to 0.

We gave full credit (3 points) for all answers that included at least the first sentence.

(Brief documentation of mystery SVC)

END OF QUIZ!
(phew!)