



Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2002

Handout 16 - Quiz I

All problems, except one, on this quiz are multiple-choice questions. In order to receive credit you must fill in the blank(s) or mark the correct answer or answers for each question. You have 50 minutes to answer this quiz.

Write your name on this cover sheet AND at the bottom of each page of this booklet.

Some questions may be much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.

CIRCLE your recitation section number:

- | | | | |
|--------------|--------------------------|-----------------------------|-------------------------------|
| 10:00 | 1. Balakrishnan/Chambers | 13. Morris+Kaashoek/Gnawali | 11. Amarasinghe/Bhattacharyya |
| 11:00 | 2. Balakrishnan/Salz | 6. Morris+Kaashoek/Chambers | 12. Amarasinghe/Gnawali |
| 12:00 | 5. Ernst/Salz | 14. Witchel/Bauer | |
| 1:00 | 8. Ernst/Yip | 3. Leiserson/Freedman | 10. Teller/Bauer |
| | 7. Saltzer/Vandiver | | |
| 2:00 | 9. Saltzer/Freedman | 4. Leiserson/Vandiver | 15. Teller/Bhattacharyya |

Do not write in the boxes below

1-3 (xx/30)	4-10 (xx/55)	11 (xx/15)	Total (xx/100)

Name:

I Reading questions

1. [10 points]: The Therac-25 (Reading #4) killed patients, while the Therac-20 apparently did not. Which of the following reasons for this difference are supported by the evidence in the Leveson paper?

(Circle ALL that apply)

- A. The Therac-25 did not use any of the same software as the Therac-20
- B. The 20 MeV beam of the Therac-20 does not have enough power to harm a patient
- C. AECL paid more attention to user complaints about the Therac-20 than about the Therac-25
- D. The Therac-25's interlocks did not work as well as the Therac-20's
- E. The Therac-25 turntable had no position sensors

2. [10 points]: The Flash paper (Reading #7) says that Flash's architecture, asymmetric multi-process event-driven (AMPED), provides better performance than a single-process event-driven (SPED) architecture. Which of the following reasons contribute to the performance improvement?

(Circle ALL that apply)

- A. If the workload consists entirely of requests for pages in the application-level cache, AMPED can read those cached pages faster than SPED.
- B. AMPED allows for batching of multiple disk operations.
- C. AMPED incurs fewer thread switches than SPED.
- D. AMPED can read a Web page from the disk at the same time that it reads pages from the application-level cache, but SPED cannot.
- E. AMPED can process a request arriving from one client at the same time that it can send a page to a different client, but SPED cannot.

3. [10 points]: Which of the following would be a fair summary of Mogul and Ramakrishnan's "Eliminating Receive Livelock" paper (Reading #8)?

(Circle ALL that apply)

- A. When overloaded, don't start processing new packets until all processing is complete for previous packets.
- B. When lightly loaded, turn off interrupts and poll for incoming packets.
- C. When overloaded, a router should never discard packets.
- D. When lightly loaded, try to batch as much work as possible in order to minimize latency.
- E. When overloaded, use interrupts to minimize latency.

II More Enforced Modularity

Ben Bitdiddle is so excited about Amazing Computer Company's plans for a new segment-based computer architecture that he takes the job they offered him.

Amazing Computer Company has observed that using one address space per program puts the text, data, stack, and system libraries in the same address space. For example, a Web server has the program text (i.e., the binary instructions) for the Web server, its internal data structures such as its cache of recently-accessed Web pages, the stack, and a system library for sending and receiving messages all in a single address space. Amazing Computer Company wants to explore how to enforce modularity even further by separating the text, data, stack, and system library using a new memory system.

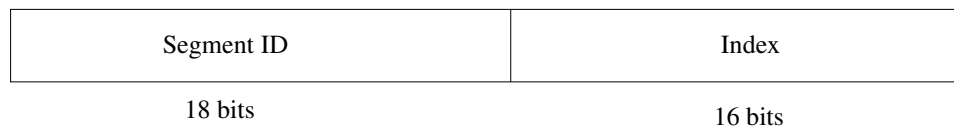
In the document "6.033 Design Project I, 2002", the Amazing Computer Company has asked every designer in the company to come up with a design to enforce modularity further. In a dusty book on Professor Morris's bookshelf about the PDP-11/70, Ben finds a description of a hardware gadget that sits between the processor and the physical memory, translating virtual addresses to physical addresses. The PDP-11/70 used that gadget to allow each program to have its own address space, starting at address 0.

The PDP-11/70 did this through having one *segment* per program. Conceptually, each segment is a variable-sized, linear array of bytes starting at virtual address 0. Ben bases his memory system on the PDP-11/70's scheme with the intention of providing isolation between modules. Ben defines a segment through a segment descriptor:

```
struct segmentDescriptor {
    physicalAddr physAddr;
    int length;
}
```

The `physAddr` field records the address in physical memory where the segment is located. The `length` field records the length of the segment in bytes.

Ben's processor has addresses consisting of 34 bits: (1) 18 bits to identify a segment; and (2) 16 bits to identify the byte within the segment:



A virtual address that addresses a byte outside a segment (i.e., an index larger than the length of the segment) is illegal.

Name:

Ben's memory system stores the segment descriptors in a table, `segmentTable`, which has one entry for each segment:

```
struct segmentDescriptor segmentTable[NSEGMENT];
```

The segment table is shared among all programs, and is stored at physical address 0.

The processor used by Ben's computer is a simple RISC processor, which accesses memory through load and store instructions. The `LOAD` and `STORE` instructions take a virtual address as their argument. Ben's computer has enough memory that all programs fit in physical memory (i.e., Ben doesn't need to develop to a pager).

Ben ports a compiler that translates a medium-level language (C) to generate machine instructions for his processor. The compiler translates the C code to a position-independent machine code: `JUMP` instructions specify an offset relative to the current value of the program counter. To make a call into another segment, it supports the `LONGJUMP` instruction, which takes a virtual address and jumps to it.

Ben's memory system translates a virtual address to a physical address with `translate`:

```
#define MASKSEGID 0x3FFFF
#define MASKINDEX 0xFFFF

physicalAddr translate (virtualAddr addr) {
    int segid = (addr >> 16) & MASKSEGID;
    struct segmentDescriptor segment = segmentTable[segid];
    int index = (addr & MASKINDEX);
    if (index < segment.length) {
        return segment.physaddr + index;
    }
    /* What should we do here? (give your answer in question 6, below) */
    ...
}
```

After successfully computing the physical address, Ben's memory management unit retrieves the addressed data from physical memory and delivers it to the processor (on a `LOAD` instruction) or stores the data in physical memory (on a `STORE` instruction).

4. [5 points]: What is the maximum sensible value of `NSEGMENT`?

(Circle best answer)

- A. 2^{16}
- B. 2^{18}
- C. 2^{32}
- D. 2^{34}

Name:

5. [5 points]: Given the structure of a virtual address, what is the maximum size of a segment in bytes?

(Circle best answer)

- A. 2^{16}
- B. 2^{18}
- C. 2^{32}
- D. 2^{34}

6. [5 points]: How many bits wide must a physical address (`physicalAddr`) be?

(Circle best answer)

- A. 16
- B. 18
- C. 32
- D. 34
- E. The design doesn't require any particular width

7. [10 points]: The code on the “...” line should

(Circle best answer)

- A. signal the processor that the instruction that issued the memory reference has caused an illegal address fault
- B. signal the processor that it should change to user mode
- C. return `index`;
- D. signal the processor that the instruction that issues the memory reference is an interrupt handler

Ben modifies his Web server to enforce modularity between the different parts of the Web server. He allocates the text of the program in segment 1, a cache for recently-accessed Web pages in segment 2, the stack in segment 3, and the system library in segment 4. Segment 4 contains the text of the library program but no variables (i.e., the library program doesn't store variables in its own segment).

8. [10 points]: To translate the Web server (written in C) the compiler has to?

(Circle ALL that apply)

- A. compute the physical address for each virtual address
- B. include the appropriate segment ID in the virtual address used by a LOAD instruction
- C. generate LONGJUMP instruction for calls to functions located in different segments
- D. include the appropriate segment ID in the virtual address used by a STORE instruction

Name:

Ben runs the segment-based implementation of his Web server, and observes to his surprise that errors in the Web server program can cause the text of the system library to be overwritten. He studies his design and realizes that the design is bad.

9. [10 points]: What aspect of Ben's design is bad and can cause the observed behavior?

(Circle ALL that apply)

- A. a STORE instruction can overwrite the segment ID of an address
- B. a LONGJMP instruction in the Web server program may jump to an address in the library segment that is not the start of a function
- C. it doesn't allow for paging of infrequently-used memory to a secondary storage device.
- D. the Web server program may get into an endless loop

10. [10 points]: For each answer that you circled in the previous question, draw an arrow from that answer to each extension of Ben's design that addresses that problem.

(Circle the extension(s) and draw arrow(s))

- A. the processor should have a protected user/kernel mode bit and there should be a separate segment table for kernel and user programs
- B. each segment descriptor should have a protection bit, which specifies if the processor can write or only read from this segment
- C. the LONGJMP instruction should be changed so that it can transfer control only to designated entry points of a segment
- D. segments should all be the same size, as pages in page-based virtual memory systems
- E. change the operating system to use a preemptive scheduler

III Are eventcounts necessary?

The system library for Ben's Web server contains code to send and receive messages. A separate program, the network manager, manages the network card on which messages arrive and are sent. The Web server and the networking manager each have one thread of execution. Ben wants to understand why he needs eventcounts for sequence coordination of the network manager and the web server, so he decides to implement the coordination using eventcounters *and* event variables.

Here are the two versions of the Web server:

Web server using eventcounts

```
eventcount inCnt;
int doneCnt;

while (TRUE) {
    while (inCnt <= doneCnt) {
        wait (inCnt, doneCnt);
    }
    doPacket ();
    doneCnt++;
}
```

Web server using events

```
event input;
int inCnt;
int doneCnt;

while (TRUE) {
    while (inCnt <= doneCnt) { // A
        waitevent (input); // B
    }
    doPacket (); // C
    doneCnt++; // D
}
```

Both solutions use a thread package as described in the class notes in Chapter 2, except for the changes to support eventcounters or events. The eventcounter solution is like the one described in Chapter 2. The `wait` function has the semantics for eventcounts: when the Web server thread calls `wait`, the thread manager puts the calling thread into the “waiting” state unless `inCnt` exceeds `doneCnt`.

The event-based solution is almost identical to the eventcounter one, but has a few changes. An `event` variable is a list of threads waiting for the event. The function `waitevent` puts the current executing thread on the list for the event, records that the current thread is in the “waiting” state, and releases the processor by calling the code for `yield`.

In both solutions, when the web server has completed processing a packet it increases `doneCnt`.

The two corresponding versions of the code for handling each packet arrival in the network manager are:

Network manager using eventcounts

```
inCnt++;
notify (WebServerThread);
```

Network manager using events

```
inCnt++; // E
notifyevent (input); // F
```

The `notify` function wakes up the Web server thread, if it is already asleep. The `notifyevent` function removes all threads from the list of the event and puts them into the “ready” state. The shared variables are stored in a segment shared between the network manager and the Web server.

Name:

Having read the Therac-25 paper, Ben is a bit worried about writing code that involves coordinating multiple activities so he decides to test the code carefully. He buys a computer with one processor, and time-shares the processor between the Web server and the network manager using a *preemptive* thread scheduler. Ben ensures that the two threads (the Web server and the network manager) never run inside the thread manager at the same time by turning off interrupts when the processor is running the thread manager's code (which includes `notify`, `wait`, `notifyevent`, and `waitevent`).

To test the code Ben changes the thread manager to preempt threads frequently (i.e., each thread runs with a short time slice). Ben runs the old code with eventcounts and the program behaves as expected, but the new code using events has the problem that the Web server sometimes delays processing a packet until the next packet arrives.

To help you along, the program statements that relate to the problem are marked with letters in the code of the event-based solution above.

11. [15 points]: Using the letters, give a sequence of statements that creates the problem. (Some statements might have to appear more than once and some might not be necessary to create the problem. It also may take fewer than 10 steps.)

(Write down the event letters in order)

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.

End of Quiz I

Name: