University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Spring 2017                                                              Ion Stoica

## Second Midterm Exam
March 21, 2017
CS162 Operating Systems

| Your Name: | |
|---|---|
| **SID AND 162 Login:** | |
| **TA Name:** | |
| **Discussion Section Time:** | |

General Information:
This is a **closed book and one 2-sided handwritten note** examination. You have 80 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points for that question. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. *Make your answers as concise as possible.* If there is something in a question that you believe is open to interpretation, then please ask us about it!
## Good Luck!!

| QUESTION | POINTS ASSIGNED | POINTS OBTAINED |
|---|---|---|
| **1** | 24 | |
| **2** | 20 | |
| **3** | 20 | |
| **4** | 22 | |
| **5** | 14 | |
| **TOTAL** | **100** | |

**P1** (24 points total) True/False and Why? **CIRCLE YOUR ANSWER**. For each question: 1 point for true/false correct, 2 point for explanation. An explanation cannot exceed 2 sentences.

a) A multilevel page table hierarchy will always take less storage space than a single level page table, given the same virtual address space size.

TRUE                            FALSE

Why?

b) Forcing all threads to request resources in the same order (e.g., resource A before B, B before C, and so on) will prevent deadlock.

TRUE                            FALSE

Why?

c) Increasing the size of a cache always decreases the number of total cache misses, all things held constant (replacement policy, workload, associativity).

TRUE                                    FALSE

Why?

d) Adding a TLB may make process context switching slower.

TRUE                                    FALSE

Why?

e) It is not possible to share memory between two processes when using multiple-level page tables.

TRUE                                    FALSE

Why?

f) Round-robin scheduling has always a higher average response time than shortest-job-first.

TRUE                                    FALSE

Why?

g) One way to respond to thrashing is to kill a process.

TRUE                                    FALSE

Why?

h) With uniprogramming, applications can access any physical address.

TRUE                                    FALSE

Why?

**P2** (20 points) **Demand Paging:** Consider the following sequence of page accesses:

A, B, D, C, B, A, B, A

a) (10 points) Fill in the table below assuming the FIFO, LRU, and MIN page replacement policies. There are 3 frames of physical memory. The top row indicates the frame number, and each entry below contains the page that resides in the corresponding physical frame, after each memory reference (we have already filled in the row corresponding to accessing A). For readability purposes, **only fill in the table entries that have changed and leave unchanged entries blank.** If there are several pages that meet the replacement criteria, break ties using the lexicographical order, i.e., A takes priority over B, B over C, and C over D.

|  | FIFO | | | LRU | | | MIN | | |
|---|---|---|---|---|---|---|---|---|---|
|  | F1 | F2 | F3 | F1 | F2 | F3 | F1 | F2 | F3 |
| A | A |  |  | A |  |  | A |  |  |
| B |  | B |  |  | B |  |  | B |  |
| D |  |  | D |  |  | D |  |  | D |
| C | C |  |  | C |  |  |  |  | C |
| B |  |  |  |  |  |  |  |  |  |
| A |  | A |  |  |  | A |  |  |  |
| B |  |  | B |  |  |  |  |  |  |
| A |  |  |  |  |  |  |  |  |  |
| # Page Faults | 6 | | | 5 | | | 4 | | |

b) (4 points) Give a sequence of page accesses in which LRU will generate a page fault on every access.

c) (3 points) Demand paging can be thought of as using main memory as a cache for disk. Fill in the properties of this cache:

    a. Associativity: _____

    b. Write-through/write-back? : _____

    c. Block Size (assume 32-bit addresses, 4-byte words, and 4KB pages) ? : _____

d) (3 points) The Nth chance replacement algorithm relies on a parameter $N$. Why might one choose a large $N$ value? Why might one choose a small $N$ value?
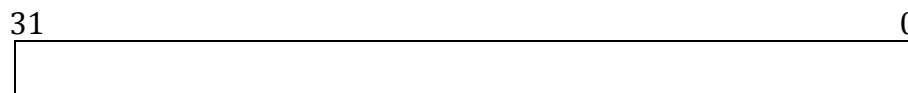
**P3** (20 points) **Caching:** Assume an 8KB cache with 32B blocks, on a machine that uses 32-bit virtual and physical addresses.

a) (6 points) Specify the **size** and **name** of each field (i.e., cache tag, cache index, and byte select / offset) in the physical address for the following cache types:
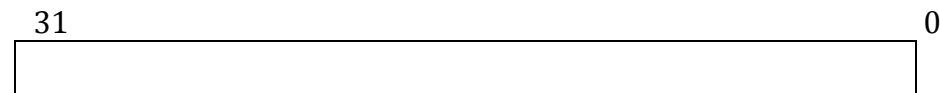
    a. Direct Mapped

31                                                       0
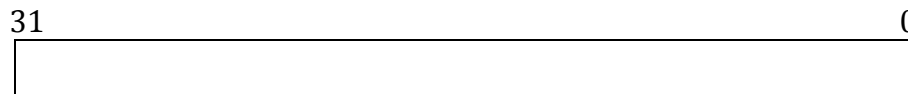
    b. Fully associative

31                                                       0

    c. Four-way associative

31                                                     0

b) (11 points) You've finished implementing your cache, which ended up being a 2-Way Set Associative cache that uses an LRU replacement policy. To test it out, you try the following sequence of reads and writes:

```
read  from   0x705F3140
write 0x1 to 0x705F3140
write 0x2 to 0x705F3150
write 0x2 to 0x705F3148
write 0x3 to 0x707A2150
write 0x3 to 0x035F2154
read  from   0x705F3140
```

Recall that caching happens at the block level, i.e., the unit of transfer between cache and main memory is one block. Also, assume that a write leads to caching the data, the same as a read. Initially, assume the cache is empty. Please answer the following questions:

a. (5 points) How many misses does the above access pattern exhibit?

b.  (3 points) Assume the cache is a write-through cache. How many writes are taking place between cache an memory?

c.  (3 points) Assume that cache is a write-back cache. How many writes are taking place between cache and memory?

c)  (4 points) Now consider that your system has 5ns of latency when accessing cache and 70ns when accessing main memory. What should your application's average hit rate be in order to have an average memory access latency of 15ns?

**P4** (22 points) **Address Translation:** Consider a computer with **16 bit** virtual and physical addresses. Address translation is implemented by a two-level scheme combining segmentation and paging. The page size is **256 bytes**.

Virtual address format:

| 2b (segment #) | 6b (page #) | 8b (offset) |
|----------------|-------------|-------------|

Segment table (Base Address specifies the address of the page table associated with the segment, and Limit specifies the total number of bytes in the segment):

| Segment ID | Base Address | Limit |
|------------|--------------|--------|
| 0x0 | 0x4000 | 0x1000 |
| 0x1 | 0xC000 | 0x2000 |
| 0x2 | 0x8000 | 0x0200 |

Page Table start address: 0x4000

| Page # |
|--------|
| 0x20 |
| 0x30 |
| 0x31 |

Page Table start address: 0x8000

| Page # |
|--------|
| 0xC0 |
| 0xC1 |

Page Table start address: 0xC000

| Page # |
|--------|
| 0x40 |
| 0x44 |

a)  (6 points) What is physical address corresponding to virtual address 0x4125 ?

b)  (12 points) Consider the following assembly code computing a string's length:

```
0x8150 .data str: .asciiz "Hello World"
0x01F4 main: li $t1, 0        # $t1 is the counter; set it to 0
0x01F8       la $t0, str      # load address (la) of str into $t0
0x01FC cnt:  lb $t2, 0($t0)   # Load first byte in $t2 from address
0x0200       beqz $t2, end    # if $t2 == 0 then goto label "end"
0x0204       add $t0, $t0, 1  # else increment the address
0x0208       add $t1, $t1, 1  # and increment the counter
0x020C       j cnt            # goto "cnt"
0x0216 end:
```

Fill in the following table after executing each of the first 8 instructions in the above code. The program counter (PC) is initialized to 0x01F4, i.e., the execution starts with instruction li $t1, 0.

| Instruction | Physical address of instruction | Content of $t0 (after executing instruction) | Content of $t1 (after executing instruction) | Content of $t2 (after executing instruction) |
|---|---|---|---|---|
| li $t1, 0 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
|---|---|---|---|---|
|  |  |  |  |  |

c) (4 points) Assume that instead of "Hello World" we have a 200-byte long string. Are there any changes we need to make to the segment or page tables to support the 200-byte string? If yes, use one sentence to describe the change. (The code is unchanged and addresses for each instruction and data remain the same.)

**P5** (14 points) **Banker's Algorithm:**

a)  (6 points) Suppose that we have the following resources: A, B, C and threads T1, T2, T3, T4. The total number of instances for each resource is:

| Total | | |
|---|---|---|
| A | B | C |
| 11 | 21 | 19 |

Further, assume that the threads have the following maximum requirements and current allocations:

| Thread ID | Current Allocation | | | Maximum Allocation | | |
|---|---|---|---|---|---|---|
| | A | B | C | A | B | C |
| T1 | 2 | 3 | 10 | 4 | 10 | 19 |
| T2 | 1 | 6 | 3 | 5 | 9 | 5 |
| T3 | 4 | 7 | 3 | 9 | 13 | 6 |
| T4 | 2 | 3 | 1 | 4 | 5 | 3 |

Is the system in a safe state? If "yes", show a non-blocking sequence of thread executions. Otherwise, provide a proof that the system is unsafe. Show your work and justify each step of your answer.

b)  (4 points) Repeat question (a) if the total number of B instances is 20 instead of 21.

c)  (4 points) Give one reason using no more than two sentences of why you might decide **not** to use Banker's algorithm to avoid deadlock.