

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

Spring 1999

Anthony D. Joseph

**Midterm Exam**  
March 3, 1999  
CS162 Operating Systems

<b>Your Name:</b>	
<b>SID:</b>	
<b>TA:</b>	
<b>Discussion Section:</b>	

General Information:

This is a **closed book** examination. You have two hours to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question; there are 100 points in all. 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 (you needn't cover every available nano-acre with writing). If there is something in a question that you believe is open to interpretation, then please go ahead and interpret, **but** state your assumptions in your answer.

**Good Luck!!**

<b>Problem</b>	<b>Possible</b>	<b>Score</b>
<b>1</b>	13	
<b>2</b>	24	
<b>3</b>	18	
<b>4</b>	30	
<b>5</b>	15	
<b>Total</b>	<b>100</b>	

## 1. Threads (13 points total):

- a. (6 points) What state does a thread share with other threads in a process and what state is private/specific to a thread? Be explicit in your answer.

- i) *Contents of memory (global variables, heap)*
- ii) *I/O state (file system)*
- iii) *CPU registers (including, program counter)*
- iv) *Execution stack*

- b. (7 points) Draw a picture of the three states of a thread and label the transitions between the states:

## 2. Context switching and CPU scheduling (24 points total):

- a. (3 points) What state is saved on a context switch between threads?

*CPU registers (including, program counter and stack pointer)*

- b. (6 points) List two reasons why Nachos disables interrupts when a thread/process sleeps, yields, or switches to a new thread/process?



c. (15 points) Consider the following processes, arrival times, and CPU processing requirements:

Process Name	Arrival Time	Processing Time
1	0	3
2	1	5
3	3	2
4	9	2

For each of the following scheduling algorithms, fill in the table with the process that is running on the CPU (for timeslice-based algorithms, assume a 1 unit timeslice):  
 For RR, assume that an arriving thread is scheduled to run at the beginning of its arrival time.

Time	FIFO	RR	SRTCF
0	1	1	1
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
Average response time			

## 3. Concurrency control (18 points total):

- a. (6 points) Match the terms in column A with the most appropriate definition from column B.

<u>Column A</u>	<u>Column B</u>
1. Synchronization	a. Piece of code that only one thread can execute at once
2. Mutual exclusion	b. Ensuring that only one thread does a particular thing at a time
3. Critical section	c. Isolating program faults to an address space
	d. Using atomic operations to ensure cooperation between threads

1.

2.

3.

that is, sometimes works and sometimes doesn't. If the implementation does not work or is dangerous, explain why (there may be several errors) and show how to fix it so it does work. Note that `ThreadFork` does the obvious thing.

```
BankServer() {
    while (TRUE) {
        ReceiveRequest(&op, &acctId1, &acctId2, &amount);
        if (op == transfer) ThreadFork(Transfer(acctId1, acctId2, amount));
        else if ...
    }
}

Transfer(acctId1, acctId2, amount) {
    account[acctId1]->Lock();
    acct1 = GetAccount(acctId1); /* May involve disk I/O */
    account[acctId2]->Lock();
    acct2 = GetAccount(acctId2); /* May involve disk I/O */
    if (acct1->balance < amount) return ERROR;
    acct1->balance -= amount; acct2->balance += amount;
    StoreAccount(acct1); /* Involves disk I/O */
    StoreAccount(acct2); /* Involves disk I/O */
    account[acctId1]->Unlock(); account[acctId2]->Unlock();
    return OK;
}
```

*Transfer is broken: Deadlock! I/O between lock acquires. Also, failure to release lock, when an error occurs. Solution: Compare acctIDs and change order of lock acquisition.*

```

Transfer(acctId1, acctId2, amount) {
    if (acct1 > acct2) {
        account[acctId2]->Lock();
        account[acctId1]->Lock();
    } else {
        account[acctId1]->Lock();
        account[acctId2]->Lock();
    }
    acct1 = GetAccount(acctId1); /* May involve disk I/O */
    acct2 = GetAccount(acctId2); /* May involve disk I/O */

    if (acct1->balance < amount) {
        account[acctId1]->Release();
        account[acctId2]->Release();
        return ERROR;
    }
    acct1->balance -= amount;
    acct2->balance += amount;
    StoreAccount(acct1); /* Involves disk I/O */
    StoreAccount(acct2); /* Involves disk I/O */
    account[acctId1]->Release();
    account[acctId2]->Release();
    return OK;
}

```

4. Memory management (30 points total):

- a. (6 points) Consider a memory system with a cache access time of 100ns and a memory access time of 1200ns. If the effective access time is 10% greater than the cache access time, what is the hit ratio  $H$ ? (fractional answers are OK)

$$\begin{aligned}
 1.1 \times T1 &= T1 + (1-H)T2 \\
 (0.1)(100) &= (1-H)(1200) \\
 H &= 1190/1200
 \end{aligned}$$

- b. (6 points) Assuming a page size of 4 KB and that page table entry takes 4 bytes, how many levels of page tables would be required to map a 32-bit address space if every page table fits into a single page? Be explicit in your explanation.

*Since each PTE is 4 bytes and each page contains 4KB, then a one-page page table would point to 1024 or  $2^{10}$  pages, addressing a total of  $2^{10} * 2^{12} = 2^{22}$  bytes. Continuing this process:*

<i>Depth</i>	<i>Address Space</i>
<i>1</i>	<i><math>2^{22}</math> bytes</i>
<i>2</i>	<i><math>2^{32}</math> bytes</i>

- c. (18 points) Consider a multi-level memory management using the following virtual addresses:

Virtual seg #	Virtual Page #	Offset
---------------	----------------	--------

Each virtual address has 2 bits of virtual segment #, 8 bits of virtual page #, and 12 bits of offset. Page table entries are 8 bits. *All values are in hexadecimal.*

Translate the following virtual addresses into physical addresses:

Virtual Address	Physical Address	Virtual Address	Physical Address
0x204ABC	0x46ABC	0x23200D	7400D
0x102041	0x10041	0x1103DB	Error!
0x304F51	Error!	0x010350	0x16350

Segment Table

Start	Size	Flags
0x2004	0x40	Valid, read only
0x0000	0x10	Valid, read/write
0x2040	0x40	Valid, read/write
0x1010	0x10	Invalid

Physical Memory

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0x0000	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
0x0010	1E	1F	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D
....																
0x1010	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
....																
0x2000	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
0x2010	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21
0x2020	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31
0x2030	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41
0x2040	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51
0x2050	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61
0x2060	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71
0x2070	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	80	81

## 5. Design tradeoffs (15 points total):

A hardware designer asks for your input on the design of a new processor and computer. You can spend \$1500 dollars on the following components:

Item	Latency	Minimum Size	Cost
TLB	10ns	16 entries	\$20/entry
main memory	200ns	16MB	\$2/MB
disk	10ms (10M ns)	2GB	\$0.20/MB

The page size is fixed at 16KB. Assume you want to run 3 – 4 applications simultaneously. Each application has an overall maximum size of 64 MB and a working set size of 256KB. TLB entries do not have Process Identifiers. Discuss how you would divide the available funds across the various items to optimize performance.