OS Midterm Solution

2013/11/18

Statistics

	2013	2012
Midterm Highest	98	98
Midterm Average	60	65 / 100
HW AVG	90	84
Final Average	?	80 / 112

Year	Range	100~90	90~80	80~70	70~60	60~50	50~40	>40
2012	#people	12	12	27	20	17	9	12
	Percentage	10%	20%	45%	60%	80%	85%	100%
2013	#people	5	17	14	29	17	10	17
	Percentage	3.5%	20%	33%	60%	75%	85%	100%

Be aware if your midterm score is BELOW 40!

Statistic

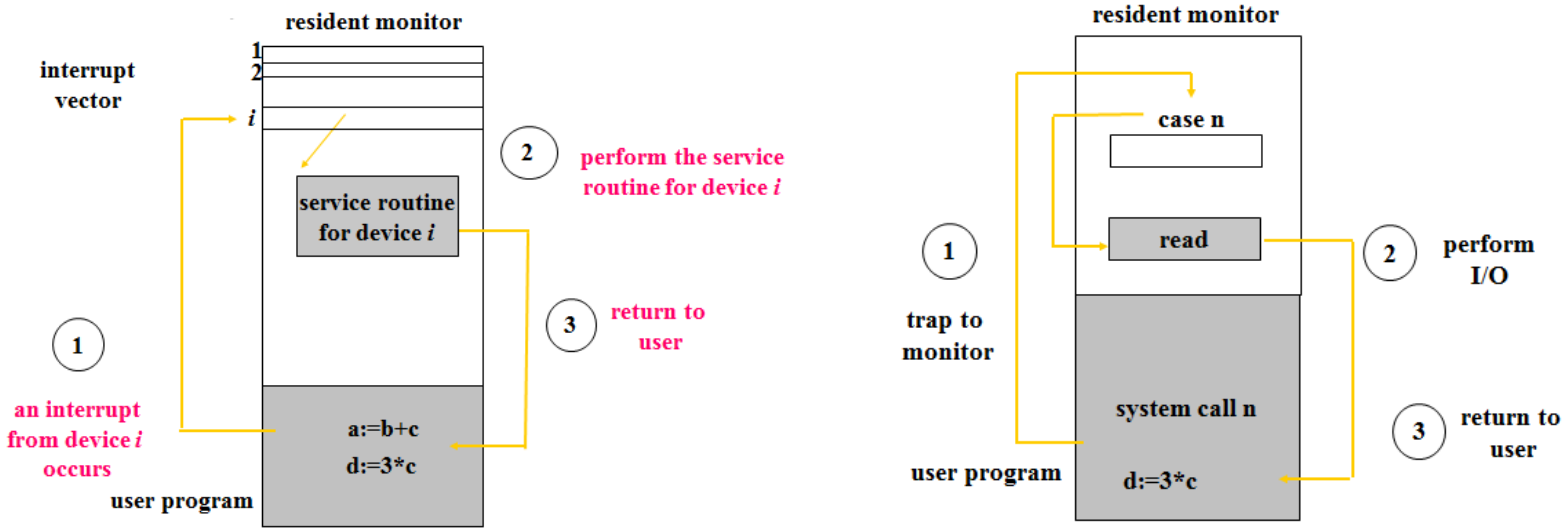
Q#	Average Score	Total Score	Correct Rate	Difficulty
1	9.2	12	76.5	Basic
2	4.2	5	84.0	Basic
3	0.7	4	**16.2**	Med
4	2.7	5	53.5	Hard
5	5.0	7	70.6	Basic
6	2.3	5	45.6	Med
7	3.0	5	59.1	Basic
8	4.4	10	43.5	Med
9	5.2	8	64.5	Basic
10	10.2	12	84.7	Basic
11	6.8	10	68.1	Basic
12	1.9	5	38.7	Hard
13	4.9	12	41.0	Hard

Level	Def.	Grades	AVG Score	%
Basic	Copy & Paste	59	43.6	74%
Med	Digest& Understand	19	7.4	39%
Hard	Digest & Apply	22	9.5	43%

1. (12%) Briefly explain the definition of following terminologies, and do a simple comparison in terms of their strength and weakness: (4 pt. each)

- Layered OS structure vs. Microkernel
  - Layer: subsystem in **level N can only call the function from components in level N-1**
  - Microkernel: move **nonessential OS component** to **user space**, use **message passing** for communication
  - Layer is easy for debugging, but hard to separate levels.
  - Microkernel is highly modulized, but communication overhead too much
- Message passing communication vs. shared memory communication
  - Message passing: **copy memory** content for communication
  - Shared memory: **access the same memory address space** for communication
  - Message passing is slower, but easier for scaling.
  - Shared memory is faster, but could have synchronization issue.
- Compile-time address binding vs. Runtime address binding
  - Compile time binding: determine the memory address after compilation
  - Runtime binding: **translate memory address by MMU** after sending from CPU
  - **Compile time binding is slower and can't relocate memory.**
  - Runtime binding allows more flexible memory management techniques, but require **HW support and result slightly slower memory access time.**

2. (5%) Use a simple diagram to illustrate the key steps for handling an interrupt.

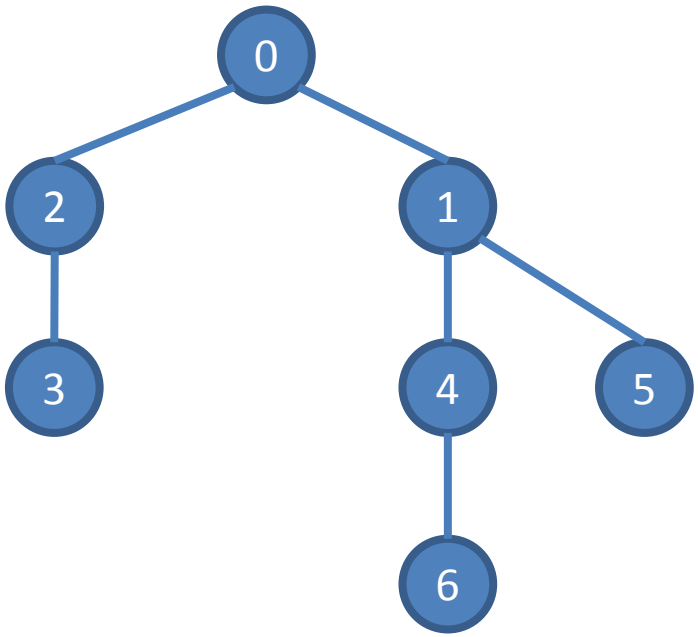


3. (4%) Briefly explain why context switch will cause system performance degradation in terms of **program execution time** and **memory access time**.

a. Execution time increases because no process can utilize CPU during context switch

b. Memory access time increases because TLB needs to be flushed

4. (5%) How many processes are created in the following program? (You must plot the process tree with process ID to explain your answer. You can assume the process ID is assigned in increasing order from 0.)




```
int main() {
  int pid = fork();
  if (pid==0) {
        for (int i=0; i<2; i++) {
              pid = fork();
        }
        execlp("\bin\ls");
  } else {
        pid = fork();
  }
  if (pid == 0) fork();
}
```

5. a. (3%) Explain what is many-to-one multithreading model?

b. (4%) Give two examples in which multithreading does NOT provide better performance than a single-threaded solution.

a. All the user threads of a process are mapped to a single kernel thread.

b. (1) On a single core computer; (2) Create way too many threads than the physical cores; (3) Synchronization overhead;

6. (5%) Briefly explain why the modern general purpose OS uses both segmentation and paging to manage physical memory?

- Use segmentation because it is closer to the user's view of a process memory content.
- Use paging because it is easier for the MMU to allocate and manage physical memory space.

7. a. (3%) Explain what is a TLB (Translation Lookaside Buffer)?

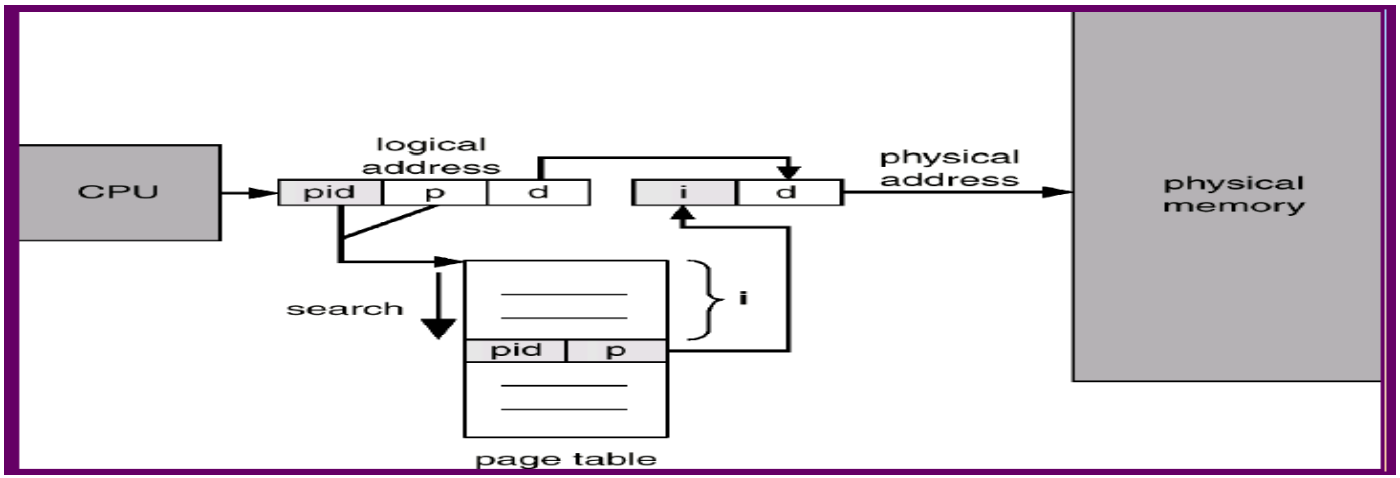
b. (2%) Why TLB is often implemented by associative memory?

a. TLB is a cache for page table

b. Associative memory allows all table entries to be checked at the same time. So the lookup time can be reduced.

8. a. (4%) Use an example to explain inverted page table mechanism. b. (6%) Discuss the difference between inverted page table and the traditional single-level page table in terms of their memory access time, page table size, and page sharing mechanism.

a.



b.

- Inverted page table could have longer memory access time because it needs to scan page table during address translation
- Inverted page table could have smaller page table size because the table is bounded by the physical memory size
- Inverted page table is more difficult to support page sharing

9. a.(3%) What is a page fault? b. (5%) Describe the steps to handle a page fault.

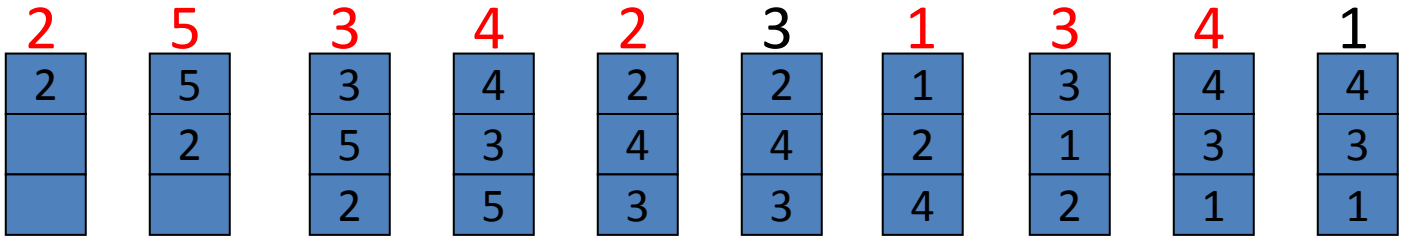
a. Page fault occurs when a process access its page which is stored in disk not in memory

b.

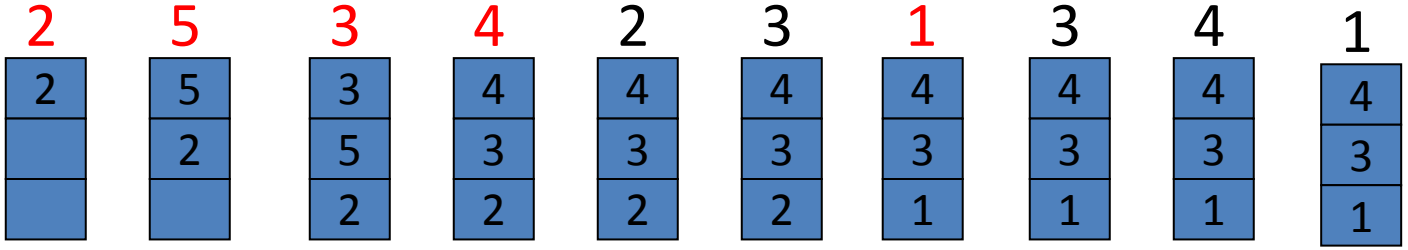
1. OS looks at the internal table (in PCB) to decide
   - Invalid reference ➔ abort
   - Just not in memory ➔ continue
2. Get an empty frame
3. Swap the page from disk (swap space) into the frame
4. Reset page table, valid-invalid bit = 1
5. Restart instruction

10. (12%) Consider a computer with 3 memory frames to handle a reference string 2, 5, 3, 4, 2, 3, 1, 3, 4, 1. How many page faults are generated by the following code, using LRU replacement?

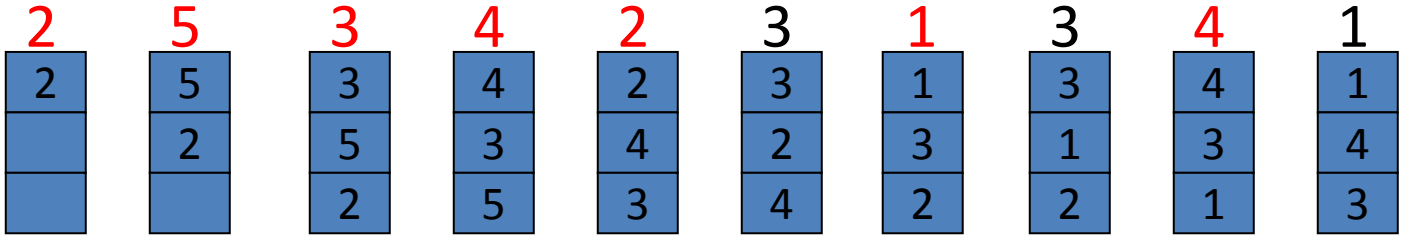
FIFO=8



OPTIMAL=5



LRU=7



11. a. (4%) Describe the steps that would cause trashing.

b. (4%) Explain how to use working-set solution to solve it.

a. processes queued for I/O to swap (page fault)

➔ low CPU utilization

➔ OS increases the degree of multiprogramming

➔ new processes take frames from old processes

➔ more page faults and thus more I/O

➔ CPU utilization drops even further

b.

WSSi : working-set size for process i

D = $\sum$ WSSi (D = total nunber of pages demanded by all processes)

If D > m (m = the total number of frames) $\Rightarrow$ thrashing

The OS monitors the WSSi of each process and allocates to the process enough frames

- if D << m, increase degree of MP
- if D > m, suspend a process

12. (5%) How many page faults are generated by the following code, using LRU replacement?

Memory reference for loading the first instruction code

Memory reference for array element:

A[0][0]; A[2][0]; A[4][0]; …… ; A[48][0];

Total: 1 + 25 = 26

13. Consider a byte-addressable computer system with a 16-bit virtual address, total physical memory size 4KB, page size is 256 Bytes, the maximum size of a segment is 1KB. Given the following segment table, page table and a 16 bits hexadecimal logical address "0C42", complete the address translation diagram below.

(i) (2%) segment table index: segment offset = 10bits ➔ segment number = 6 bits ;

logical address = 0000,1100,0100,0010 ➔ segment number = 000011 = 3

(i) (4%) linear address: 0110,0111,0111 + 00,0100,0010 = 0110,1011,1001

(ii) (2%) page table index: page number = 4 bits ➔ page number = 0110 = 6

(iii) (4%) physical address: 0111,0000,0000 + 1011,1001 = 0111,1011,1001