Homework Assignment #1 for CPSC 410 & 611
Fall 2014

• The answers to the questions should be typed using a word processor. Points will be deducted if the answers are hand-written.

• Try to write short answers (one or two paragraphs only for each question). Sometimes one sentence is enough. Do not copy the whole paragraph(s) from the textbook but use your own wording.

• Due date: Friday, September 26 to CSNet.

1. Ex. 1

• What is the purpose of interrupts?

interrupts are used to handle errors and events, it is summoned via the hardware or software. The current state would be saved until the process is completed such as I/O. The system would then resume processing other programs.

• What are the differences between a trap and an interrupt?

A software trap can be implemented by a user in their code to prevent issues. The trap works like and exception that is user defined, the exception can call other system calls to deal with the issue.

• Can traps be generated intentionally by a user program? If so, for what purpose?

Yes, exceptions are usually user defined. Exceptions are used to catch errors that would normally cause the program to crash, catching errors allow the program to deal with the issue.

2. Ex. 2

• What is the purpose of the command interpreter?
command-line interpreter is used to signal system calls from a terminal. The command interpreted has both user level and kernel level. These two levels are separated to prevent system damage.

• Why is it usually separate from the kernel?
The command interpreter it is not part of the kernel because it can be changed.

• Would it be possible for the user to develop a new command interpreter using the system-call interface provided by the operating system?
Yes, it is possible for a user to develop a new command interpreter using systems-calls.

3. Ex. 3

Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in this order), how would each of the algorithms:

(a) first-fit

(b) best-fit

(c) worst-fit algorithms

place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in this order)? Which algorithm makes the most efficient use of memory?

(b) best-fit

First Fit

Memory	Size
	100 KB
212 112	500 KB Remain: 176
	200 KB
	300 KB
417 Leftover: 462	600 KB Total free space: 876 KB

Best Fit

Memory	Size
	100 KB
417	500 KB
112	200 KB
212	300 KB
426	600 KB Total free space: 533 KB

Worst Fit

Memory	Size
100	100 KB
417	500 KB
200	200 KB
300	300 KB
212 112 Leftover: 426	600 KB Total free space: 783 KB

4. Ex. 4

Compare the main memory organization schemes of

• contiguous-memory allocation

• pure segmentation

• pure paging

with respect to the following issues:

(a) external fragmentation

(b) internal fragmentation

(c) ability to share code across processes.

> Contiguous memory allocation scheme has the most external fragmentation since the address spaces are allocated continuously and holes develop as old processes terminate and new processes are created. it is however, allows processes to share code because the process's logical address is not broken into small segments.
>
> Pure segmentation also have external fragmentation as a process is set as contiguously in physical memory and fragmentation would occur as segments of dead processes are replaced by segments of new processes. Segmentation, however, enables processes to share code.
>
> Pure paging has no external external fragmentation, but it has internal fragmentation. Processes are allocated in page granularity and if a page is not completely utilized, it results in internal fragmentation and a corresponding wastage of space. Paging also enables processes to share code at the granularity of pages.

5. Ex. 5

Consider the following segment table:

What are the physical addresses for each of the following logical addresses?

#	Base + offset	
0	649	Legal
1	23100	Legal
2	590	Ilegal
3	1727	Legal
4	2,064	Ilegal

6. Ex. 6

A certain computer provides its users with a virtual-memory space of 2^{32} bytes. The computer has 2^{18} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4096 bytes. A user process generates the virtual address 11123456 (decimal notation).

• Explain how the system establishes the corresponding physical location. You can show it using binary and/or Hex notation.

• What is the page number and page offset for this virtual address?

page size = 2^{12}
2^{32}/2^{12} = 2^{20} = page table size.
Give a virtual address in hex = 0x11123456 consist of the page address + the offset.
The first 20 bits from the left represents the page address, and the remaining represent the offset, which is the physical address. So:

Physical address = 0x456.

7. Ex. 7

Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

1 millisecond = 1,000,000 nanoseconds = 1e6 nanoseconds
EAT = (1-p)*(100) + (p)*(100 + (1-.7)*(8 msec) + (.7)*(20msec))
= 100 - 100p + 100p + (2.4e6)*p + (14e6)*p
= 100 + (16.4e6)*p
200 = 100 + (16.4e6)*p
p = 100/16.4e6

8. Ex. 8

Assume there is an initial 4 KB memory segment where memory is allocated using the Buddy System. Using Figure 9.27, p. 354 as a guide, write which segment sizes are available (you can draw the tree illustrating how the following memory requests are allocated):

- request 240 bytes = 128 bytes + 64 bytes + 32 bytes + 16 bytes
- request 120 bytes = 64 bytes + 32 bytes + 16 bytes + 8 bytes
- request 60 bytes = 32 bytes + 16 bytes + 8 bytes + 4 bytes
- request 130 bytes = 64 bytes + 32 bytes + 16 bytes + 8 bytes + 8 bytes + 2 bytes

In total we need (1 X 128), (3 X 64), (4 X 32), (4 X 16), (4 X 8), (1 X 4), (1 X 2)

Before allocation

4096	0
2048	1
1024	1
512	0
256	1
128	2
64	4
32	4
16	5
8	5
4	1
2	2

After allocation

4096	0
2048	1
1024	0
512	0
256	1
128	1
64	1
32	0
16	1
8	1
4	0
2	1

Next, write which segment sizes are free and available after the following releases of memory. Which segment is still in use? Perform coalescing whenever possible.

• release 240 bytes = 128 bytes + 64 bytes + 32 bytes + 16 bytes

• release 60 bytes = 32 bytes + 16 bytes + 8 bytes + 4 bytes

• release 120 bytes = 64 bytes + 32 bytes + 16 bytes + 8 bytes

In total we have (1 X 128), (2 X 64), (3 X 32), (3 X 16), (2 X 8), (1 X 4).

Before Free

4096	0
2048	1
1024	0
512	0
256	1
128	1
64	1
32	0
16	1
8	1
4	0
2	1

After Free

4096	0
2048	1
1024	0
512	1
256	1
128	0
64	1
32	1
16	1
8	1
4	1
2	1