Operating Systems	Homework 3
CS 4410	Due Monday, July 31

1. **Big address spaces.** Suppose we have a system with a 40-bit physical address space, a 64-bit logical address space, and a page size of 32kb. We wish to use an inverted page table (IPT).

   (a) How large would a single inverted page table for this system be?

   (b) The IPT is bigger than a page. Design a method for breaking the IPT itself into pages. Carefully describe how a virtual address will be translated into a physical address.

   There are at least two approaches you could take, and some of them are better than others; consider different alternatives and choose a good one. Justify any design decisions you make.

2. **Small address spaces.** Suppose a 16-bit virtual address space, an 8 bit physical address space, hierarchical paging with 16 byte pages. Each page table entry has r/w/x permission bits and a valid bit; these are stored (in this order) in the high-order bits of the entry.

   The entire contents of memory are shown in table 1; the PTBR indicates that the top level page table of the currently running process is in frame 7.

   Suppose the currently running process tries to increment the data in address 0xBEEF. Indicate whether this results in (a) a page fault, (b) a segmentation exception, or (c) success. If (c), indicate what the incremented value is. In any case, list the physical addresses accessed.

3. **Page Replacement.** The program `paging.py` counts the number of page faults incurred by a given sequence of memory accesses and a given page replacement strategy.

   (a) Complete the implementations of the LRU, FIFO, RANDOM, and OPT page replacement strategies. The portions of the code you should modify are marked with `TODO`.

   (b) The files in the `data` folder contain traces of array accesses of various sorting algorithms when run on an input array of size 1024 (the insertion sort and bogosort traces have been truncated to 300,000 accesses). Compare the number of page faults between the different algorithms for a variety of memory sizes (run `paging.py -h` to see the command line parameters). Discuss your results in your written submission.

   (c) Use the provided files to explore the practical effects of Belady's Anomaly.

4. **Monitor.** Add a proof of correctness to your solution to bridge.py from homework 2. Let crossing[i] be the number of threads that have returned from `cross(i)` but have not yet called finished (here i can be either NORTH or SOUTH).

   (a) Clearly define the correctness criteria in terms of these two variables.

   (b) Prove that your code is safe. Be sure to document any invariants you are maintaining on your local state, and to explain why they hold both before you call wait and after you return from a monitor function.

   (c) Argue that your code is live: that it cannot get into a situation where threads are waiting but none are crossing.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
0x00:	0xD9	0xF9	0x46	0x43	0xAD	0xDD	0x8A	0x66
	0x44	0x51	0x3E	0x2D	0x4E	0xDB	0x85	0xCE
0x10:	0xC0	0xB1	0x5B	0x25	0x45	0x3D	0x91	0x0D
	0x21	0x5D	0x9D	0xE8	0xAA	0x0E	0xA7	0x86
0x20:	0xB2	0x60	0x67	0x8F	0x5C	0xFB	0xFA	0xAB
	0xC0	0x60	0xBC	0x3F	0xE2	0x59	0xDA	0xEC
0x30:	0x17	0xB3	0x9E	0x3C	0x05	0x3B	0x29	0xD5
	0xC7	0x0D	0x63	0x76	0x20	0xC1	0xEC	0x00
0x40:	0xEC	0xEB	0xCA	0x97	0x25	0xDF	0xAA	0x82
	0x90	0x03	0x59	0x07	0xF0	0xB5	0x68	0x24
0x50:	0x3A	0x55	0x7B	0x21	0x15	0x1A	0x78	0x25
	0x79	0x43	0x9B	0x71	0x4C	0xA2	0x20	0x75
0x60:	0x34	0x34	0x98	0x7B	0x3A	0x7B	0xBC	0x43
	0x94	0x23	0x60	0xF9	0xBE	0xC9	0x8B	0xEA
0x70:	0xD7	0x7E	0x99	0xF6	0x79	0x1C	0xB5	0xF2
	0x96	0x29	0x39	0xD2	0xC2	0xC0	0xC0	0x92
0x80:	0x1B	0x59	0x46	0x92	0xA5	0xFC	0xEE	0xBF
	0xE5	0x26	0x4D	0xA3	0x06	0x31	0xC7	0x13
0x90:	0x1F	0x3D	0x16	0x69	0xAB	0xC9	0x92	0x32
	0x1C	0x29	0x39	0x44	0x5F	0x42	0xE6	0xA5
0xA0:	0xC2	0x91	0x32	0x11	0x93	0x03	0xA5	0x4A
	0x73	0x33	0x75	0xDD	0x10	0x4E	0x53	0xBE
0xB0:	0x92	0x83	0x5E	0x3B	0x13	0x43	0x09	0xE1
	0x3F	0xD5	0xB7	0x4F	0xD8	0x17	0xC1	0xA7
0xC0:	0x82	0x56	0x02	0xCA	0x6E	0xAD	0x4F	0x8F
	0x66	0x9C	0x78	0x81	0x27	0x18	0x02	0x65
0xD0:	0x34	0x03	0xB6	0xF5	0x6C	0xA2	0x7E	0x8B
	0x7F	0x2F	0x1B	0xB5	0xE4	0xD6	0x7E	0x9D
0xE0:	0x2E	0x6A	0xBA	0x2D	0x42	0x73	0x68	0x7D
	0xFE	0x3A	0xA3	0xA2	0xE0	0x9C	0x6D	0x70
0xF0:	0xF2	0x84	0xD3	0xAF	0xCC	0x8F	0xD6	0x67
	0x8E	0x1C	0x52	0xFB	0xEB	0x55	0x82	0x6D

Table 1: Entire contents of memory. The PTBR is 7.