Craig Huff
Homework5
ID #: 009841390

1. **Given resource type X with two instances, resource type Y with a single instance, and three threads. As illustrated in the resource allocation graph,**
   a. **Are any Coffman conditions true in this state?**
   Mutual Exclusion: Yes
   Non-preemption: Yes
   Hold and Wait: Yes
   Circular Wait: No. (Thread 1 is not waiting for any resources to be released)

   b. **Is there any deadlock? Why or why not?**
   Since not all of the Coffman conditions have been met, there is no deadlock.

2. **Both segment table and page table are used to translate from logical address to physical address. But the structures of these tables are different; Each entry in a segment table is {base, length} while each entry in a page table is{frame#}. Why the differences? Why there is no length in a page table? Can we record base in a page table?**

   A segment table is used with elements of variable sizes, which is specified in each segment. For instance, one segment can be 1000 bytes long, while another is only 256. This is different from a page table which has a fixed length for each frame, most commonly 4096 bytes. The fixed frame size is the reason a page table doesn't need a length in addition to the frame #. The table knows exactly where the frame begins and ends. In a segment table, we specify the start and end location as the base and the length, which are recorded into the page table.

   It is possible to store the base from a segment table in a page table, however, since bases will generally be larger than the size of the frame it is inefficient to do so.

3. **Consider the following page reference string: 7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4 Assuming demand paging with 3 frames, fill in the table to indicate pages in the frames, page fault if any, and total number of page faults, for the following page replacement algorithms.**

   a. **LRU -> # of Page Faults:** 13

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reference	7	2	3	1	2	5	3	4	6	7	7	1	0	5	4
Frame 0	7	7	7	1	1	1	3	3	3	7	7	7	7	5	5
Frame 1	x	2	2	2	2	2	2	4	4	4	4	1	1	1	4
Frame 2	x	x	3	3	3	5	5	5	6	6	6	6	0	0	0
Page Fault?	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y

b. **FIFO -> # of Page Faults:** 12

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reference	7	2	3	1	2	5	3	4	6	7	7	1	0	5	4
Frame 0	7	7	7	1	1	1	1	1	6	6	6	6	0	0	0
Frame 1	x	2	2	2	2	5	5	5	5	7	7	7	7	5	5
Frame 2	x	x	3	3	3	3	3	4	4	4	4	1	1	1	4
Page Fault?	Y	Y	Y	Y	N	Y	N	Y	Y	Y	N	Y	Y	Y	Y

4. **On a machine with 16 bytes page size, given the following page table for a process, and four of these 8 entries are mapped to page frames. Frame 0 starts at physical address 0.**

Page Number	Frame Number	Virtual Address Ranges
0	3	0 - 15
1	0	16 - 31
2	Not in main Memory	32 -47
3	Not in main Memory	48 - 63
4	2	64 - 79
5	Not in main Memory	80 - 95
6	1	95 - 111
7	Not in main Memory	112 - 127

a. **Make a list of all virtual address ranges (in decimals, byte-level) that would cause page faults.**

Page 2: 32 – 47
Page 3: 48 – 63
Page 5: 80 – 95
Page 7: 112 – 127

b. **What are the corresponding physical addresses (in decimals, byte-level) of the following virtual addresses (in decimals, byte-level)**

- **0** :(16 x 3) + 0 = 48 | page 0 > frame3
- **17** :(16 x 0) + 1 = 1 | page 0 > frame 0
- **31** :(16 x 0) + 15 = 15 | page 0 > frame3
- **32** : page 2 >> Page Fault
- **100** :(16 x 1) + 4 = 20 | page6 > frame1

5. **Assume that a system has a 32-bit virtual address with N-KB page size (where N >= 1, and 1KB = 1024 bytes). Write a C program that accepts two command line parameters, the first one being the value of N (in decimal notation) and the second one being a virtual address in decimal notation, and have it output the page size, the page number and offset for the given virtual address**

```
Craigs-MacBook-Pro:vaddr craig$ ./vaddr 1 19981
Virtual address translation by CraigHuff390
Page size = 1024, virtual address 19981 => page number = 19, offset = 525
Craigs-MacBook-Pro:vaddr craig$ ./vaddr 2 19982
Virtual address translation by CraigHuff390
Page size = 2048, virtual address 19982 => page number = 9, offset = 1550
Craigs-MacBook-Pro:vaddr craig$ ./vaddr 4 19984
Virtual address translation by CraigHuff390
Page size = 4096, virtual address 19984 => page number = 4, offset = 3600
Craigs-MacBook-Pro:vaddr craig$ ./vaddr 8 19988
Virtual address translation by CraigHuff390
Page size = 8192, virtual address 19988 => page number = 2, offset = 3604
```