Dan Otieno.

CPE434-01.

HOMEWORK 3.

03/01/23.

1. **Multi-level page tables. Assume a virtual address machine with a 32bit address. Assume the address is divided into 4 parts a,b,c,d with the first three parts used to index into a three-level page table, and the fourth is used to index into the location on the page. Assume the values for a,b,c,d are 4,6,6,16 respectively resulting in a 64 KB sized page. What is the total size in bytes of the page tables for the smallest program, which is a program containing a single page of text and heap, starting at location 0, and a separate page for the stack, starting at location 0xFFFFFFFF and running downwards? Assume all page table entries are 4 bytes each.**
   - *Each table entry = 4bytes.*
   - *Size of page = 64KiB = 2^{16} bytes.*
   - *Virtual address space = 2^{32}.*
   - *Number of entries is therefore 2^{32} / 2^{16} = 2^{16}entries.*
   - *Total size of the smallest program = 2 × 2^{16} = 2^{17} or 131072 bytes.*

2. **What is the effect of allowing two entries in a page table to point to the same page frame in memory? Explain how this effect could be used to decrease the amount of time needed to copy a large amount of memory from one place to another. What effect would updating some byte on one page have on the other page?**

   When two entries in a page table are allowed to point to the same page frame in memory, users can share code and data. If the code is reusable, such that multiple programs can use it simultaneously, in the case where those programs are larger in size, a lot of memory space can be saved through the shared use. We can decrease the amount of time needed to copy large amounts of memory by having different page tables point to the same memory location. However, if the code cannot be used by multiple programs at the same time, then sharing it means any user with access can modify it, and those modifications will be reflected on the other page, or the copy, i.e. and updated byte on one page will be updated on the other page as well.

3. **Given six memory partitions of 120 KB, 700 KB, 350 KB, 200 KB, 150 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)?**

   a. **First Fit:**
      - *115KB into 120KB. (**5KB**, 700KB, 350KB, 200KB, 150KB, 125KB).*
      - *500KB into 700KB. (5KB, **200KB**, 350KB, 200KB, 150KB, 125KB).*
      - *358KB ==> Cannot allocate, no partition is large enough to accommodate.*
      - *200 KB into 200KB. (5KB, **0KB**, 350KB, 200KB, 150KB, 125KB).*
      - *375KB ==> Cannot allocate, no partition is large enough to accommodate.*

b. **Best Fit:**
   - *115KB into 120KB. (5KB, 700KB, 350KB, 200KB, 150KB, 125KB).*
   - *500KB into 700KB. (5KB, 200KB, 350KB, 200KB, 150KB, 125KB).*
   - *358KB ==> Cannot allocate, no partition is large enough to accommodate.*
   - *200 KB into 200KB. (5KB, 0KB, 350KB, 200KB, 150KB, 125KB).*
   - *375KB ==> Cannot allocate, no partition is large enough to accommodate.*

c. **Worst Fit:**
   - *115KB into 700KB. (120KB, 585KB, 350KB, 200KB, 150KB, 125KB).*
   - *500KB into 585KB. (120KB, 85KB, 350KB, 200KB, 150KB, 125KB).*
   - *358KB ==> Cannot allocate, no partition is large enough to accommodate.*
   - *200 KB into 350KB. (120KB, 85KB , 150KB, 200KB, 150KB, 125KB).*
   - *375KB ==> Cannot allocate, no partition is large enough to accommodate.*

4. **Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers): (1KB = 2^{10} = 1024, to determine page number, we divide by 1024, to determine offset we perform modular arithmetic on 1024).**
   a. **3085**
   b. **42095**
   c. **215201**
   d. **650000**
   e. **2000001**

	PAGE NUMBER	*OFFSET*
3085	*3*	*13*
42095	*41*	*111*
215201	*210*	*161*
650000	*634*	*784*
2000001	*1953*	*129*

5. **Consider the page table for a system with 12-bit virtual and physical addresses and 256-byte pages.**

Page	Page Frame
0	–
1	2
2	C
3	A
4	–
5	4
6	3
7	–
8	B
9	0

The list of free page frames is D, E, F (that is, D is at the head of the list, E is second, and F is last). A dash for a page frame indicates that the page is not in memory. Convert the following virtual addresses to their equivalent physical addresses in hexadecimal. All numbers are given in hexadecimal.

- **98F.**
- **121.**
- **404.**
- **7FE.**
    - ***98F----------->Page no. 9, Page Frame 0: Physical Address = 08F.***
    - ***121----------->Page no. 1, Page Frame 2: Physical Address = 221.***
    - ***404----------->Page no. 4, Page Frame D: Physical Address = D04.***
    - ***7FE----------->Page no. 7, Page Frame E: Physical Address = EFE.***