CS 520: Introduction to Operating Systems

Homework Assignment #5 (due November 28, 2019)

Read Chapters 9, and 10 and make sure that you understand everything covered in Lectures 6 and 7—leave no rock unturned!

1. Write a program that implements the Buddy System algorithm for memory management. For input the program accepts 1) the amount of total available memory and 2) the sequence of integers with allocation (+) or return (-) requests. As output, the program must print the table showing the state of the memory (in the same way it is demonstrated in Lecture 6 after each allocation or return. Test the program with the following input parameters: Two megabytes of available memory and the following sequence: *(A: +20K), (B: +35K), (C: +90K), (D: +40K), (E: +240K), (D: -40K), (A: -20K), (C: -90K), (B: -35K), (E: -240K).* Turn in the listing of the program and the printout. **(35 points)**

2. A computer with a 32-bit address uses a two-level page table. Virtual addresses are split into a 9-bit top-level page table field, an 11-bit second-level page table field, and an offset. How large are the pages and how many are there in the address space? (1**5 points)**

3. Below is an execution trace of a program fragment for a computer with 512-byte pages:

   Load word *8118* (a value of a procedure input parameter) into register *0*
   Push register *0* onto the stack
   Call a procedure at *4120* (the return address is stacked)
   Remove one word (the actual parameter) from the stack and
   Compare the actual parameter to constant *7*
   If equal, jump to *5000*

The values of *PC* and *SP* are, respectively, *2020* and *10192*. (The stack growth down—toward *0.*) Each instruction occupies four bytes . Produce the page reference string generated by this program. (**10 points)**

4. 1) Explain the value and drawbacks of “non-traditional” hardware architectures for supporting virtual memory (i.e., *inverted page table* and *TLB-only*).

   2) Using the *inverted* page table below,

0	**2**	**Process 1**
1	**3**	**Process 2**
2	**1**	**Process 1**
3	**0**	**Process 2**
4	**0**	**Process 1**

give the physical page frame number corresponding to virtual page numbers 0, 1, and 2 of *Process 1*. **(15 points)**

5. Provide a proof that, with three-frame memory, MRU will result in *3(l+1)* page faults, and LFU will result in *3[min(k,l)+1]* page faults, but FIFO and LRU will result only in six page faults on a reference string $\omega = (1,2,3)^k (4,5,6)^l$. (**25 points**)