CSE 410 - Computer Systems
Homework 7

Assigned: Friday, May 28, 2004

Due: Friday, June 4, 2004
At the start of class

Your name: $\boxed{\text{Solution}}$ DWJ

1. Imagine that two students are working together on a software development project. All the files they need are located on a remote server that is accessible to both of the students.

Being good software developers, the students are keeping a bug database updated as they work on their code. This requires that they have both the source code file and the database file open in read/write mode when they want to do any work. Only one person can open a file in read/write mode at a time.

a. Late Sunday night student Ann decides to work on the project. First, she opens the database with read/write access. Then she opens the source file, also with read/write access, and she starts working. At 1:30 Monday morning, student Billy decides to do some work also. Billy attempts to open the database in read/write mode, but access is denied because Ann has the file open in read/write mode. Because Billy needs both the database and the source file open in read/write mode to do any development, he cannot start working until Ann closes the files.

Are Billy and Ann stuck in a deadlock situation? No. Ann is working.

b. If yes, describe how each of the four necessary conditions for deadlock is true in this case. If no, describe one condition that is not true in this case.

Circular wait is not true in this case. Ann is working, she is not waiting for Billy.

Note that the potential for deadlock does exist in this scheme. If Ann opened the database and Billy opened the source file, then they would be stuck in a deadlock. The only thing that prevents deadlock in this design is that they open the resources in the same order, thereby avoiding circular waiting.

2. The drawing on the next page shows a 2-level page table for a particular process and the associated physical memory. One address is shown already filled in. The program address 0xFFFC points to an entry in the stack (highest numbered addresses). The top two bits of the address are used to select the Level 1 entry of interest that points to the associated level 2 page table. The level 2 page table contains the physical page number where the data is actually stored.

a. How many pages of physical memory are there? Give your answer in decimal base 10.

$FF_{16}+1 = 100_{16} = 256_{10}$

b. What is the maximum number of physical memory pages that any one process can occupy, given the addressing scheme shown in the drawing?

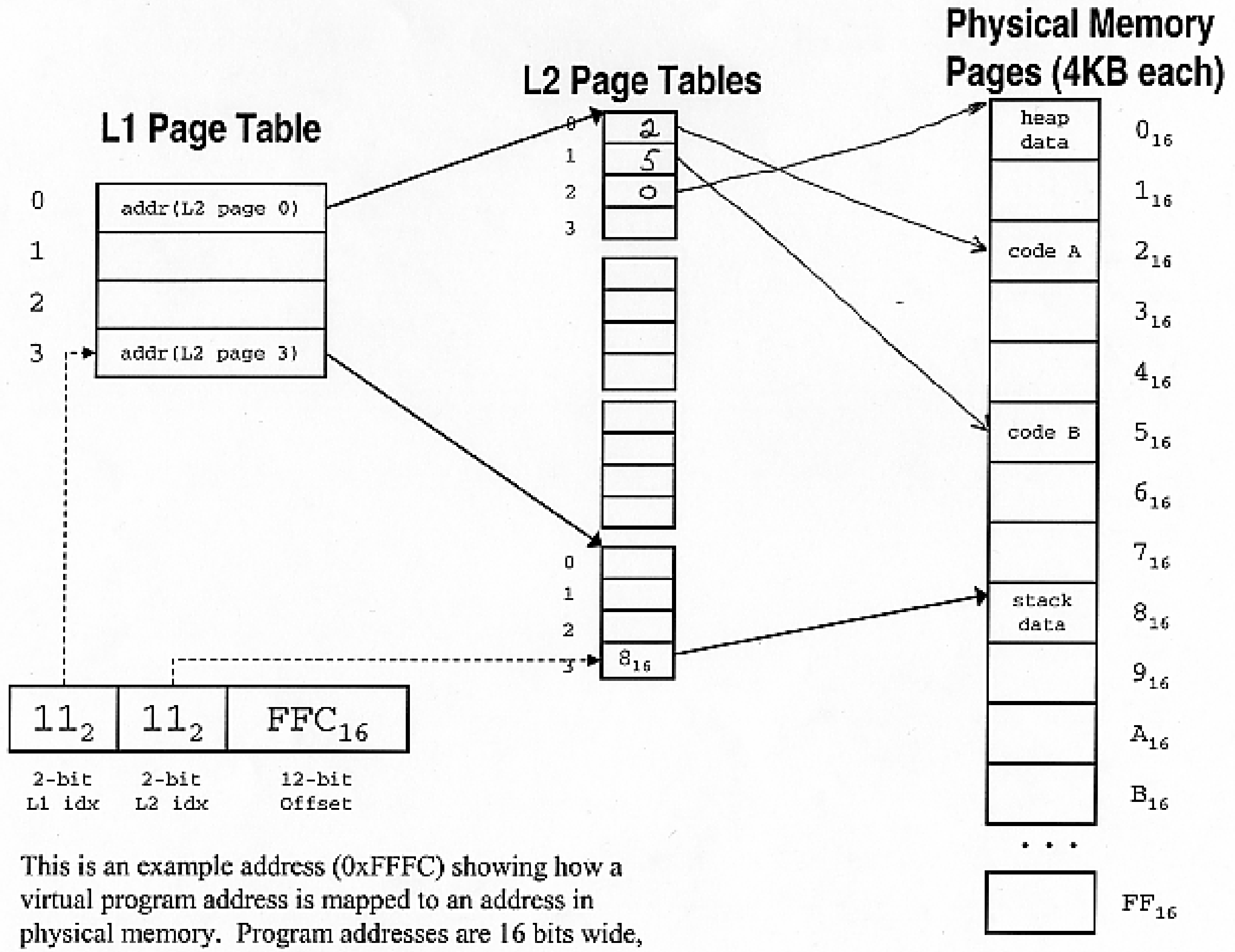
16_{10}. There are 4 entries in the L1 table. Each entry points to an L2 table with 4 entries. $4 \times 4 = 16$

c. The logical address space for a process is determined by the width of the program addresses, in this case 16 bits. The physical address space is determined by the width of the physical addresses, in this case 20 bits. This is a factor of 16 ($2^4=16$). Is it possible to fit more than 16 processes in memory at once on this system? If so, under what conditions? If not, why not?

Yes. Many processes do not actually occupy all of the locations that are addressable by their program addresses. The program that is shown in the dwg, for example, is only using 2 code pages and 2 data pages, 4 pages of the possible 16.

d. The program shown is fairly small. There are just two pages allocated for program code. Assume that the physical memory containing code page A is mapped to program addresses 0x0000 through 0x0FFF and the physical memory containing code page B is mapped to program addresses 0x1000 through 0x1FFF. Fill in the Level 2 page table entries that would make this happen and draw the associated arrows.

e. There is one page allocated for heap data. Assume that the physical memory containing the heap data is mapped to program addresses 0x2000 through 0x2FFF. Fill in the Level 2 page table entry that would make this happen and draw the associated arrow.



This is an example address (0xFFFC) showing how a virtual program address is mapped to an address in physical memory. Program addresses are 16 bits wide, physical addresses are 20 bits wide.

3. Fragmentation in a memory system refers to the waste of memory space either because it cannot be allocated (external) or because it is allocated and won't be used (internal).

a. In a system using paged memory and a page size of 4096 bytes (4KB) as in problem 2, what type of fragmentation will we see?

Internal

b. What is the maximum possible amount of wasted space due to fragmentation in a single page on this system?

4095 bytes. ie a page may be allocated but hold only 1 byte of data. All the rest of the page is wasted space.

4. Consider a system using demand paging to manage memory allocation to processes.

a. What is the meaning of the term "page fault" on this system?

The program tries to access a valid program address (logical address) but the page containing that address is not in memory. The system has to read in the page from disk before the program can continue.

b. When a page fault occurs, what is the most time consuming portion of the work that the operating system must do before the user process can start running again?

Read the page into memory from disk.

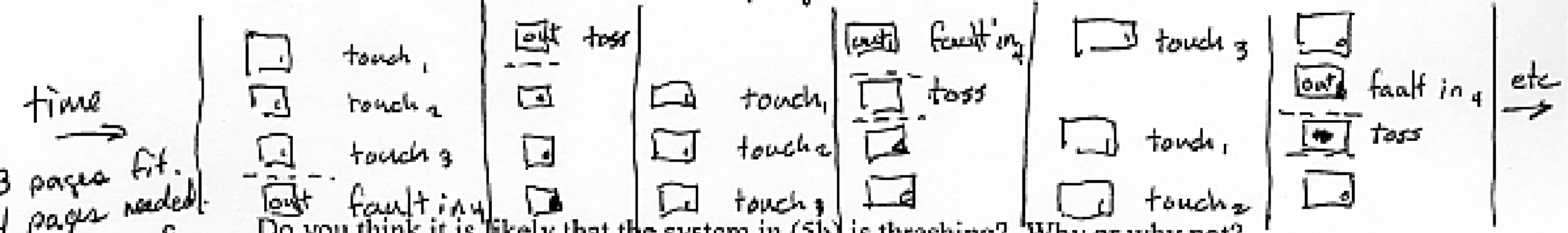
5. When memory is full and a new page is needed, the OS must select a page to evict from memory to make room for the incoming page.

a. The OPT page replacement algorithm is provably the best algorithm. What is the characteristic of the OPT algorithm that makes it impossible to implement?

It needs to select the page that will not be used for the longest time in the future. We don't know and cannot calculate future usage.

b. The second chance (or clock algorithm) degenerates to the First-In-First-Out FIFO algorithm if all the reference bits are set when a page fault occurs. Describe in general terms how a single program might cause this to happen on a system. (for example: describe the program memory size and reference locality characteristics.)

Assume the program requires more pages than fit in memory. Imagine that the program has low locality of reference and is only touching one entry per memory page each time through a loop. Depending on the pattern of references, it would be possible for most or all of the in-memory pages to be touched between each page fault.



c. Do you think it is likely that the system in (5b) is thrashing? Why or why not?

Yes. The references to the memory pages happen very quickly, then there is a long pause while the missing page is read in from disk. Then there is a very short CPU burst and another long wait for the disk. The program is essentially doing disk I/O, with very little actual time available for calculations.