CSE 378 - Machine Organization & Assembly Language - Winter 2009

HW #4

1. Here is a series of address references given as word addresses: 7, 13, 1, 28, 25, 17, 48, 49, 3, 13, 1, 9, 4, 27, 28, and 25.
    a. Assuming a Direct-mapped cache with 16 one-word blocks that is initially empty, label each reference in the list as a hit or a miss.
    b. Now assume that the cache is a 2-way set associative, one-word blocks, total 16-word capacity.

    If there is a cache-miss, indicate which of the following types of miss it is (compulsory/conflict **(with what)**/capacity). As a brief refresher, here are the definitions:

    compulsory miss - miss caused by the first access to a block that has never been in the cache.
    conflict miss - miss where multiple blocks compete for the same set.
    capacity miss - miss caused when the cache cannot contain all the blocks needed during execution of the program.

    For each of your answers to both parts a & b fill in the table shown below. Show the final contents of the cache in each case.

    Assume a Least Recently Used replacement policy where applicable.

    (a)

Address	Hit/Miss	Type
7		
13		
1		
28		
25		
17		
48		
49		
3		
13		
1		
9		
4		
27		
28		
25		

(b)

Address	Hit/Miss	Type
7		
13		
1		
28		
25		
17		
48		
49		
3		
13		
1		
9		
4		
27		
28		
25		

2. You've been tasked to upgrade a mission critical function in a visiting computational linguistics professor's code provided here as a tarball (or here if you prefer ZIP files). The function (as defined in find_letter.h) searches for a string with a given letter somewhere in it. The current version runs too slowly. We think it may have something to do with locality, but we're not certain.

   Your tasks:
   1. Rewrite the function to run faster.
   2. Explain why the old function was slow.
   3. Explain why yours is faster.
   4. Explain in what circumstances your code would be slower than the provided code.

   Notes on the provided code:
   Provided is some code to load the specially formatted files that the professor likes to use in his research. You are not expected to grok any code outside of the function you are to rewrite. Also provided is a simple main function for testing purposes.

   The file format is as follows:

   int (describes how long the longest string is)
   words
   words
   ...
   lastword

   The file words.txt is a simple example of this type of file.

   Due to the...odd string format that the professor uses, you may not use any of the standard c string library functions. The format he uses allows embedded nulls for some reason.

3. You are the lead engineer on CezeCorp's debug team, and are testing the company's first processor that uses data caching. After finding that the processor fails to meet its performance targets, you discover that the wires carrying the 32-bit memory address from the Memory Access pipeline stage to the cache have been reversed:
   pipeline_address[31] connects to cache_address[0]
   pipeline_address[30] connects to cache_address[1]
   pipeline_address[29] connects to cache_address[2]
   …
   pipeline_address[0] connects to cache_address[31]

   Since this reversal is consistent for loads and stores, the processor continues to function correctly (although with intolerably slow performance). Why does this wiring mistake decrease the processor's performance?

4. As we saw in class, page tables require fairly large amounts of memory, even if most of the entries are invalid. One solution is to use a hierarchy of page tables. The virtual page number can be broken up into two pieces, a "page table number" and a "page table offset," which is described in the figure below.

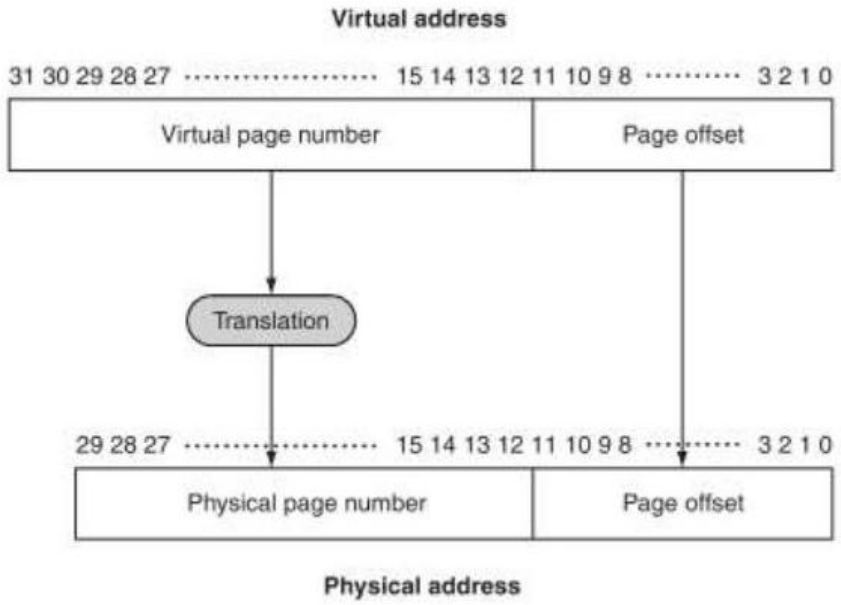


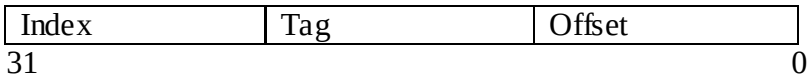
FIGURE 7.20 Mapping from a virtual to a physical address. The page size is 2^{12} = 4 KB. The number of physical pages allowed in memory is 2^{18}, since the physical page number has 18 bits in it. Thus, main memory can have at most 1 GB, while the virtual address space is 4 GB.

(From Hennessy and Patterson's **Computer Organization and Design, 3rd Edition**, pg 513)

The page table number can be used to index a first-level page table that provides a physical address for a second-level page table. The page table offset is used to index into the second-level page table to retrieve the physical page number.
One way to arrange such a scheme is to have the second-level page tables occupy exactly one page of memory.

Assuming a 32-bit virtual address space with 4KB pages and 4 bytes per page table entry, how many bytes will each program need to use to store the first-level page table (which must always be in memory)? Provide information on how to decode the address (i.e. what bits for the page table number, pagetable offset and physical page offset). Explain your solution.

5. Do you see any problems with the indicated decoding of addresses for caches? If so, what is it?



6. With respect to TLBs, indicate whether the three fields (Tag, Index and Offset) shown here should be Physical or Virtual addresses and also state the reason. For your answer, fill in the table provided below.

Tag	Index	Offset

Table:

Field	**Physical/Virtual**	**Reason**
Tag		
Index		
Offset		

7. **Virtual Memory and TLBs**:

a. Given the following stream of ~~16-bit~~ 32-bit virtual addresses, update the TLB and Page Table shown below. Assume 4 KB pages, a four-entry fully-associative TLB, and true LRU replacement. If pages must be brought in from disk, increment to the next largest page number.

Addresses: 4095, 31272, 15789, 15000, 7193, 4096, 8912

TLB

Valid	Tag	Physical Page
1	11	12
1	7	4
1	3	6
0	4	9

Page Table

Valid	Physical page or in Disk
1	5
0	Disk
0	Disk
1	6
1	9
1	11
0	Disk
1	4
0	Disk
0	Disk
1	3
1	12
0	Disk
1	3
1	7
0	Disk

b. Repeat the question, but this time assume that pages are 16 KB (instead of 4 KB).

c. What would be an advantage of having a larger page size? Are there any disadvantages?

d. Show the final contents of the TLB if it is two-way set associative (and pages are 4 KB).

8. **I/O:** Given the following two applications 1) A database for storing large image files, 2) a video game, answer these questions about appropriate disk usage:

   a. Is decreasing the sector size likely to help or hurt this application? (explain)

   b. Would increasing disk rotation speed help or hurt this application?(explain)

9. **I/O**: Assume a hard disk has the following specifications:
   - An average seek time of 11 ms
   - A 7200 RPM rotational speed
   - A 30MB/s average transfer rate
   - 3 ms of overheads for making a request

   a. Given that sectors are 1024 bytes, what is the average time to read or write a 1024-byte sector?

   b. Now assume that sectors are 2048 bytes. What is number of random 2048-byte sectors that can be read in one second?

10. **Performance**: Our two favorite programs have the following distribution of instructions of type A, B, C, and D, and our current processor has the following CPI for those instructions:

	A	B	C	D
Fraction of program 1	20%	15%	55%	10%
Fraction of program 2	30%	30%	25%	15%
CPI	5	4	3	2

a. We have heard there is a new processor that implements Type B instructions blazingly fast – each one now executes in only one cycle. What is maximum performance improvement we might expect from this new processor for program 1 and program 2?

b. What would be our performance improvement if instead we just made our original processor run twice as fast? For program 1? For program 2?