CS 356 Virtual Memory Exercises

Redekopp

Name: _____

Score: _____

Enter Answers on Blackboard.

1.) Given a virtual memory system with 32-bit virtual addresses, 16 KB pages, and 36-bit physical addresses answer the following questions:

   a. (4 pts.) Given a single level page table, how much memory would be required to hold just the table assuming each entry in the table requires 4 bytes (this includes the page frame, valid, dirty and other bits).

   b. (16 pts.) Given a three level page table where the 1st level has 32 entries, the 2nd level has 64 entries and the third level contains the rest of the needed entries, show the address bit field breakdown (which bits are used for levels 1, 2, and 3 page tables and which bits are used as the page offset.

   c. (12 pts.) Assuming 4 byte entries in each level of page table, what is the worst case memory usage (in bytes) required for the 3 level page table system described in the previous part if 10 virtual pages are in use.

   d. (12 pts.) Assume a 4-way set associative TLB with 128 total entries. Show the mapping (fields) of the virtual address for accessing the TLB. Include the tag, set and page offset fields.

For questions 2-4, assume a 2-way set associative D-TLB (data only, no code pages) with 64 entries and LRU replacement. Also assume a virtual memory system with 32-bit virtual addresses and 4 KB pages.

2.) (5 pts.) If the D-TLB entries are initially all empty/invalid, how many unique pages could be referenced before a D-TLB entry ***may*** be evicted (replaced). Give your reasoning to receive credit.

3.) (5 pts.) Now assume a program is run and at a certain point in time all D-TLB entries contain valid translations. What is the maximum **amount of memory** (in bytes) that the program can access w/o causing a TLB miss.

4.) Examine the following code operating on three **integer** (word) arrays A, B, and C. Assume the variable i is allocated in a register as is the constant ARRAY_SIZE and neither requires accessing memory. Further assume the arrays are allocated contiguously (B starts after A's last element, etc.) and the right-hand side (RHS) of the assignment is evaluated from left-to-right (A[i] is accessed first, then B[i], etc.)

```
for(i=0; i < ARRAY_SIZE; i++){
   A[i] = A[i] + B[i] + C[i];
}
```

a. (20 pts.) The worst-case scenario is that all three array translations map to the same set. (Show the TLB address mapping.) What size would the arrays have to be (ARRAY_SIZE=?) so that an access to A[i], B[i], and C[i] require different translations but that the translations all map to the same D-TLB set (show the start addresses for A[i], B[i], C[i] to support your answer…There are probably many sizes that would work, pick the smallest. [Remember we are using the configuration described before the previous problems and that each array entry is an integer = word = 4-bytes.]

b. (10 pts.) After the 0^{th} iteration completes which 2 of the three arrays will have translations in the TLB. [Hint: Keep in mind that the D-TLB is 2-way set-associative and uses LRU replacement.]

c. (12 pts.) Given the situation after the 0^{th} iteration, how many D-TLB (translation) misses will be incurred by the next iteration? [Hint: Take into account the evaluation order and what the last values accessed would have been from the previous iteration.]

d. (4 pts.) By simply rewriting/re-ordering the assignment statement, can you reduce the number of D-TLB misses? Hint: remember we said the RHS is evaluated from left to right and then assigned to the left hand side (LHS).