CoE 1110 Introduction to Computer Architecture

Official due date: Friday, 10 December 2021
Accepted until Sunday, 12 December 2021

1. A recent paper proposed eliminating paging and using segmentation instead. Their reasoning was that memories can be made large enough to avoid using virtual memory in most cases. Suppose we adopt this philosophy and require that all memory references refer to a segment descriptor. Give a design for such a descriptor. What changes would you make to the RISC-V instruction set and architecture?

2. Here's another simulation problem. You are provided with a address "trace file" of the SPICE circuit simulator running on a uniprocessor. These addresses will be interpreted within the standard RV32 context: A two level page table and a 12 bit offset. You will need to write at least one program to answer these questions:

   (a) How many first-level pages are needed?

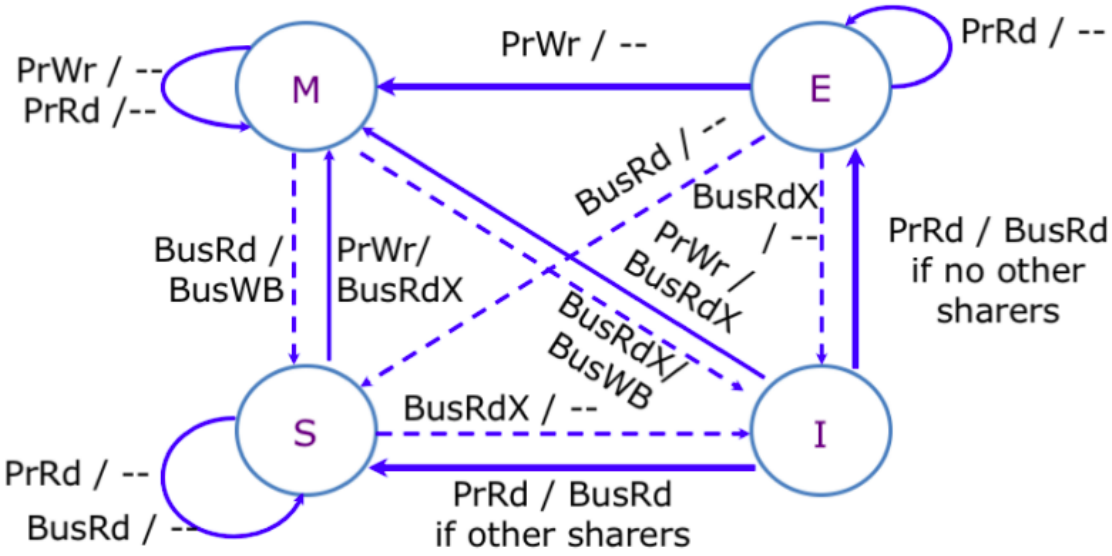
   (b) How many second-level pages are needed?

   (c) (The hard part) Implement a simulator for a 32 entry full-associative TLB. You can use any replacement policy you'd like (hint: random would be easiest). Credit will be given for complex replacement policies and flexible TLB size. You need to report the hit rate for your TLB.

   You should assume that the addresses in the trace are virtual addresses and don't worry about the physical page numbers!

   The trace format is as follows: Two fields are examined per line: access type and address. The access type is numeric: 0 for read, 1 for write, 2 for instruction fetch, 3 for miscellaneous. The address is hexadecimal, beginning with an optional "0x" or "0X". Fields are separated by white space (space or tab), and everything following the first two fields of a line is ignored.

3. A multiprocessor with 4 processors implements a write-back cache using the MESI algorithm for cache coherency as shown below:



Assume that location x1A0 is not in any cache at the start of the following sequence of memory transactions. Show the state (M, E, S or I) for each cache line containing location 0x1A0 in each processor cache and the state in main memory after each operation. Also note any transfers to or from memory.

(a) Processor 0 reads from location 0x1A0.

(b) Processor 0 writes to location 0x1A0.

(c) Processor 1 reads from location 0x1A0.

(d) Processor 2 reads from location 0x1A0.

(e) Processor 1 writes to location 0x1A0.

(f) Processor 3 writes to location 0x1A0.

Action	P0	P1	P2	P3	Memory state	Memory Transfer
P0 reads						
P0 writes						
P1 reads						
P2 reads						
P1 writes						
P3 writes						