10420 CS410001 – Computer Architecture 2016

Project #3

1. Project Objective
   a. Based on the single-cycled CPU simulator from project #1, implement a MIPS CPU simulator with memory hierarchy, Translation-Lookaside Buffer (TLB), and virtual page table mechanism. The **memory size**, **page size, cache total size**, **block size** and **set associativity of the cache** should be **configurable.**
   b. Design and submit your own test case to verify the functionality of the memory hierarchy configuration.

2. Project Description
   The simulator is similar to that of project 1 except the following:
   a. Name the executable **CMP** (which stands for Cache_Memory_Page_table).
   b. All data accesses from instructions are using **virtual addressing**
   c. Both instruction cache and data cache have only one level and your simulator should cover both caches.
   d. Cache Organization
      i. Both instruction cache and data cache adopt **write-back/allocate** policy.
      ii. The cache miss replacement policy for both caches: for the cache line under consideration, replace the **least indexed invalid** set if exists; otherwise, replace the **Bits**-**Pseudo LRU** set. Details of Bits-Pseudo LRU can be found in Appendix E.
      iii. The **default** instruction cache is of **16 bytes**, **4-way associative**. The **block size** for **instruction cache** is **4 bytes**.
      iv. The **default** data cache is of **16 bytes**, **direct map**. The **block size** for **data cache** is **4 bytes.**
   e. Cache Initialization
      The valid bit of each cache block is set to be false before the simulation starts. All other contents are initialized as “don’t care” (x’s).
   f. TLB Organization
      i. There should be two TLBs, one for IPageTable and one for DPageTable.
      ii. The TLBs is **fully-associative** and its size is a quarter of the page table size, i.e., #TLB_entries = 1/4*(#page_table_entries).
      iii. TLBs adopt the LRU replacement policy. In other words, replace the least indexed invalid entry if exists; otherwise, replace the LRU entry.
   g. TLB Initialization
      The valid bit of each page table block is initialized to be false before simulation begins. All other contents are “don’t care” (x’s).

h. Page Table Organization
   i. Although theoretically we should have page table cover full 32-bit virtual space, for simplicity of verification you are required to calculate the page table size from the given disk size specified in this project, i.e.,
      #page_table_entries = disk_size / page_size.
   ii. You have to map virtual address (VA) to physical address (PA)
   iii. At page fault, we assume that the virtual address (VA) is exactly the disk address.
   iv. The **default instruction page** size is **8 bytes** and the **default data page** size is **16 bytes**.

i. Page Table Initialization

   The valid bit of each page table block is initialized to be false before simulation begins. All other contents are "don't care" (x's).

j. Memory Organization
   i. Both instruction memory and data memory adopt **write-back/allocate** policy.
   ii. Memory replacement policy for page faults: If memory space is available, place data to the first available page closest to the page #0; otherwise, replace the **LRU** set. Pick the **least indexed** set to be the victim in case of tie.
   iii. The **default instruction memory** size is of **64 bytes** and the **default data memory** size is of **32 bytes.**

k. Memory Initialization

   All initial memory contents are $0x00000000_h$.

l. Disk Initialization
   i. Assume that both the instruction disk and data disk are of **1K bytes size**.
   ii. All other memory contents whose addresses not specified by the image file are assumed to be of value $0x00000000_h$.

m. Configurability

   The executable takes arguments from the command line. **All size related parameters should be of power of two, and the exponent should be great than 1 (2^n, n > 1).** Note that if no command line parameters are set, the default configuration is taken for simulation. Other specifications are the same as *project_1.pdf*. The parameters should be of the following order:
   i. The instruction memory (I memory) size, in number of bytes
   ii. The data memory (D memory) size, in number of bytes
   iii. The page size of instruction memory (I memory), in number of bytes
   iv. The page size of data memory (D memory), in number of bytes
   v. The total size of instruction cache (I cache), in number of bytes
   vi. The block size of I cache, in number of bytes
   vii. The set associativity of I cache

viii. The total size of data cache (D cache), in number of bytes
ix. The block size of D cache, in number of bytes
x. The set associativity of D cache

3. Input and Output Format

   Input: Same as that of Project 1. Please refer to the specification of Project 1 and *Appendix B*, "*Sample Input*."

   Output: For each test case, **report.rpt** and **snapshot.rpt** should be generated.

   a. **snapshot.rpt**

      The requirement is the same as that for project 1. Please refer to *project_1* and *Appendix C-1*, "*Sample Output for Project 1*."

   b. **report.rpt:**

      - For details please refer to *Appendix C-3*, "*Sample Output for Project 3*."
      - report.rpt should contain the following information for total memory access: total hit /miss number of I-cache, D-cache, I-pagetable, D-pagetable, I-TLB, D-TLB.

4. Test Case Design

   a. Design a test case for the default configuration or other legal configuration to verify if your simulator handles every cache event correctly.

   b. The following testcases are **invalid**:

      i. i-memory & d-memory address overflow
      ii. i-memory & d-memory misaligned
      iii. Simulation cycles over 500,000

5. Modularization

   On top of the same design of Project 1, you can add a file named as cmp.c to implement your cache, TLB, page table related functions.

6. Grading Policy

   a. Mostly are same as Project 1, with each testcase (open, hidden, student testcase) testing with **three different configurations**: default configuration and two other configurations.

      i. 10% discount if you fail one of them
      ii. 19% discount if you fail both of them
      iii. 0 point if you fail all of them

   b. Demo will focus on:

      i. Detail design of your program
      ii. Basic organization of TLB, page table, and memory
      iii. Project report

7. **Etiquette**
    a. **Do not plagiarize others' works, or you will fail this course.**
    b. **No acceptance of late homework.**
    c. **Please constantly check the class website announcements for any possible updates.**