10220 CS410001 – Computer Architecture 2014

Project #3

1. Project Objective
   a. Based on the single-cycled CPU simulator from project #1, implement a 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. About the Simulator
   The simulator is similar to that of project 1 except the followings:
   a. The executable should be named **CMP** (which stands for Cache_Memory_Pagetable).
   b. All instructions accesses data using **virtual addressing**
   c. Both instruction cache and data cache have 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 **LRU** set..
      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 cares” (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 same replacement policy as that for cache. In other words, replace the least indexed invalid entry if exists; otherwise, replace the LRU entry.
   g. 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 default disk size 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** **data page** size is **16 bytes** and the **default** **instruction page** size is **8 bytes**.

h. 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 cares" (x's).

i. 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**.

j. Memory Initialization

i. All initial memory contents are $0x0000000_h$.

k. 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 $0x0000000_h$.

For configurability, the executable takes arguments from the command line. **All size parameters should be of multiple of four.** Note that if no command line parameters are set, the default configuration should be applied 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 Format

   It is the same as that of Project 1. Please refer to the specification of Project 1 and *Appendix B,* "*Sample Input*."

4. Output Requirement

   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 Icache, Dcache, Ipagetable, Dpagetable, ITLB, DTLB.

5. Test Case Design

   Design a test case for the default configuration or other legal configuration to verify if your simulator handles every cache event correctly. By cache events, we mean read hit, read miss, write hit and write miss.

6. Evaluation and Grading

   Same as Project 1.

   - Note that for all evaluations, verifications will be done on **nthucad workstation**. Furthermore, we will evaluate your project using scripts. Please make sure that your project can be executed by the script provided by TA's. **If it cannot run through the script, you will win zero points even if your program or result is correct.**

7. **Etiquette**

   a. **Do not copy 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.**