SUNY at New Paltz

Project: Virtual Memory Manager

Operating System

Low Level Design Document

Author:
Xu Zhang

Supervisor:
Professor Atul Kumar

October 28, 2014

Low Level Design Document
Programming Project: Virtual Memory Manager

Xu Zhang
zhangx5@hawkmail.newpaltz.edu

October 28, 2014

1 Overview

In high level design document, I designed the solution by decomposing the problem. In this document I will make much more detailed design.
In this document, I shall make a low level design for programming project. This document is based on the third version of high level design. And there will be some changes I need to make as I refine the details of the design.
There are three sections in this documents, an overview of low level design, project program outline, and a project declaration code file listing.
By the end of the low level design, there will be a very detailed declaration of the structure of code.

2 Code File Outline

2.1 Language

This project will be implemented in Java.

2.2 Source Files

- Use BACKING_STORE.bin to represent the backing store.
- Use addresses.txt for testing.

2.3 Classes and Methods

The diagram Figure 1 is the Class Diagram.

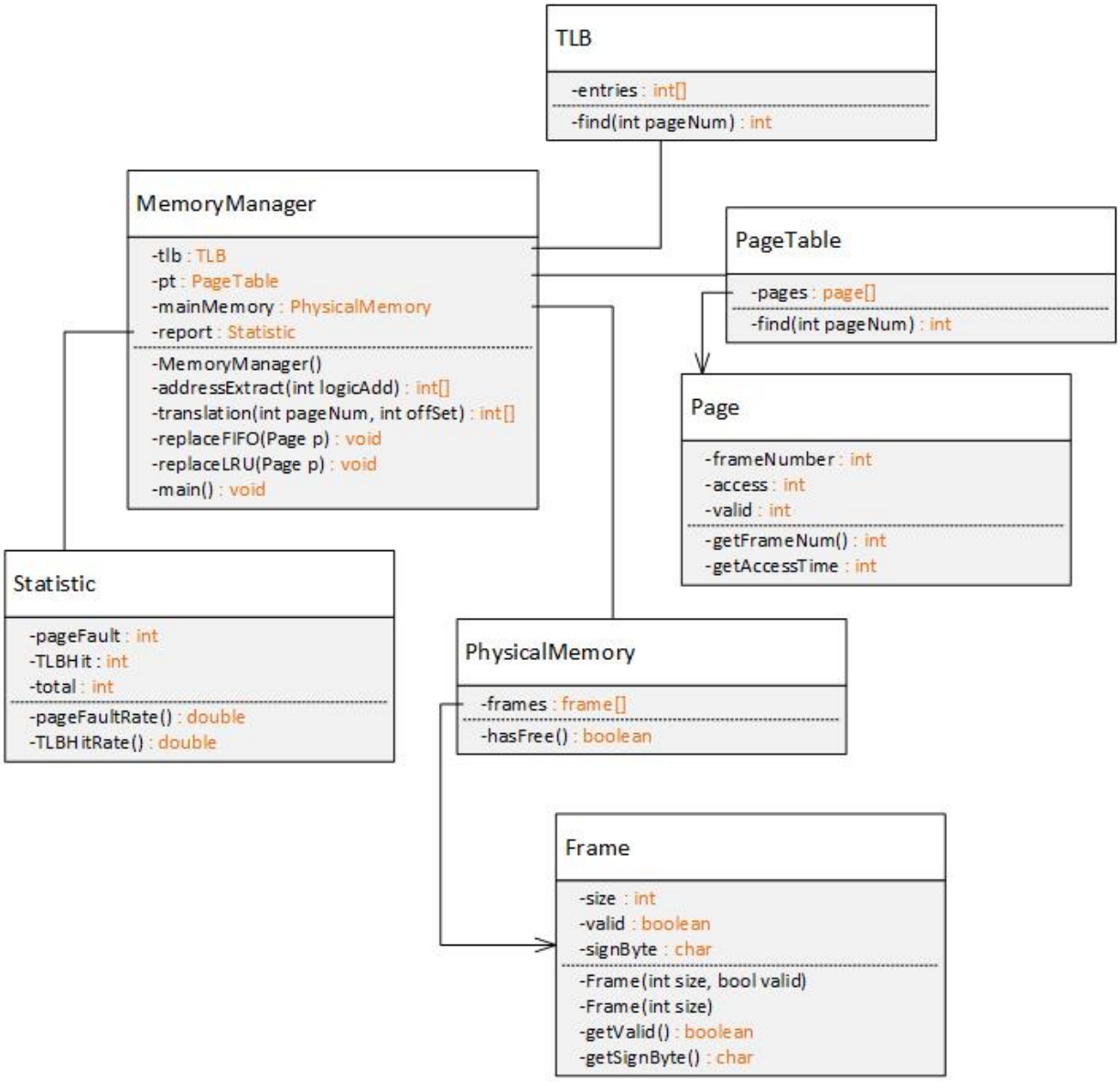

Figure 1: class diagram

3 Code File Listing

3.1 MemoryManager

- Description: Simulate the memory management unit, including the basic modules designed in the high level document.
- Data:
  - tlb : a referenc to TLB object.
  - pt : a reference to PageTable object.
  - mainMemory : a reference to PhysicalMemory object.
  - report : a reference to Statistic object.
- Method:
  - addressExtract(int logicAdd) :
    Extract the page number and offset from the logic address.
    * parameters : Integer Logic Address
    * return : An integer array answer[],answer[0]is the page number, answer[1] is the offset.
  - translation(int pageNum, int offSet) :
    Call method find(int pageNum) from pageTable to get the Frame Number. Read from BACKING_STORE.bin when meet a page fault and update the pageTable and TLB. Call a page replacement method when there is no free frame in main memory. At same time, update the report data.
    * parameters : Integer pageNum, Integer offSet
    * return : An integer array answer[],answer[0]is the physical address, answer[1] is the signed byte value.
  - replaceFIFO(Page p) :(optional)
    Using FIFO algorithm to do the page replacement.
    * parameters : none
    * return : void
  - replaceLRU(Page p) :(optional)
    Using LRU algorithm to do the page replacement.
    * parameters : none
    * return : void

3.2 Page

- Description: Build a data structure for implementation of page table(and logic memory if nessessary).
- Data:
  - frameNumber: Frame number
  - access : how many times allocate in main memory
  - valid: whether allocate in the main memory
- Method:
  - getFrameNum():
    - parameter : none
    - return: frame number
  - getAccessTime():
    - parameter : none
    - return: access time

3.3 PageTable

- Description: Implemented by array of page objects, the index of array represent the page number, and call getFrameNum() method to get the frame number.
- Data:
  - pages: array of page objects.
- Method:
  - find(int pageNum):
    - parameter : page number
    - return: frame number

3.4 TLB

- Description: Implemented by integer array, the index of array represent the page number, and the value of each item represent the frame number. The length of the TLB is smaller than page table.

- Data:
  - entries: integer array of frame numbers.
- Method:
  - find(int pageNum):
    - parameter : page number
    - return: frame number

3.5 Frame

- Description: Build a data structure for implementation of physical memory.
- Data:
  - size: the size of the frame(not nessessary in this program).
  - vaild: boolean value to check whether this frame is free.
- Method:
  - getValid():
    - parameter : none.
    - return: boolean value.

3.6 PhysicalMemory

- Description: Implement the physical memory.
- Data:
  - frames: an array of frame objects.
- Method:
  - hasFree():
    - parameter : none.
    - return: boolean value.

3.7 Statistic

- Description: Implement the physical memory.
- Data:
  - pageFault: how many times the page fault happens
  - TLBHit: how many times the TLB hit happens
  - total: how many times the translation method excute
- Method:
  - pageFaultRate():
    - parameter : none.
    - return: page fault rate: pageFault/total.
  - TLBHitRate()
    - parameter : none.
    - return: TLB hit rate : TLBHit/total.

4 Page Replacement Algorithm

4.1 FIFO Algorithm

Create a FIFO queue to hold all pages in memory.
When a page is brought into memory, we insert it at the tail of the queue.
When page fault happen and there is no free frame in the physical memory we replace the page at the head of the queue and insert the new page at the tail of the queue.

4.2 LRU Algorithm

Modify the PageTable class. Associate with each page-table entry a time-of-use eld and add to the CPU a logical clock or counter. The clock is incremented for every memory reference. Whenever a reference to a page is made, the contents of the clock register are copied to the time-of-use eld in the page-table entry for that page. Replace the page with the smallest value of time-of-use.

Algorithm 1 Algorithm LRU Page Replacement

for all pages q in the page table **do**

if q.lastTime()< min **then**

$victim = q$

$min = q.lastTime()$

end if

end for

$p.setLastTime() = currentTime$