**PROJECT REPORT**

**15146312 Park, Gang Min**

This report is Created in pseudo code form for java files : AddressTranslator.java, BackStore.java, PageTable.java, PhysicalMemory.java, TLB.java. The program runs through the AddressTranslator.java file and converts the logical address into a physical address. The logical address is stored in inputFile.txt within the project folder and the physical address is recorded in the BACKING\_STORE file. For the first time, check the TLB table and if TLB miss, check the page table. If another page fault occurred, import data from BACKING\_STORE. Print Virtual address, Physical address, and value when the conversion from the logical address to the physical address is completed. Finally, trace and print TLB miss, Page Fault. A report for procedure 9 is added at the bottom of the report.

**AddressTranslator.java**  
This class is for operating address translator. Only the main function exists in the class.   
Read the first "inputfile.txt" file.   
Create and initialize objects of the TLB, PageTable, PhysicalMemory, and BackStore classes.   
**Main function:**  
While (repeat inputfile.txt one line){  
Read the logical address stored in inputfile.txt line by line.   
Because the logical address contains 32 bits of address, we divide it by 655336=2^16 to obtain the 16 bits we need and store the rest in the logical address variable.  
Because the offset is 8 bits, divide it by 2^8 to save the remainder and save the quotient as page number.  
The frame number is returned by passing the page number to the TLB object. If the page number is present in the TLB table, the frame number is received and -1 is returned if it does not exist.  
If (the frame number returns -1){  
means tlb miss. tlb\_miss+1.  
In this case, the page number is sent to the page table object for the frame number to be returned. If the frame number exists in the physical memory, the number is returned and -1 is returned if not.  
If (the frame number returns -1){  
Page fault. page\_fault+1.  
The page number must be sent to the Backing Store object to be returned to the frame on which the page exists.  
The returned frame is stored in the physical memory and the number of the frame is returned.  
Update the page table with the page number and the updated frame number.  
}  
Update the tlb table with the page number and frame number (this code is inside the if approaching the backing store). But this is where it should be.  
}  
Physical address is 8 bits frame number plus 8 bits offset.  
Gets the value corresponding to the physical address.  
After that, print virtual address, physical address, and value.  
}  
Print tlb\_Miss and page\_fault numbers at the end of all conversions.

**TLB.java**

This class has written TLB tables and functions. The TLB table is created as a hash table within the constructor, with 16 entries, initialized as -1. The key value is also initialized from 0 to -16 and a list is created and operated to keep the number of tables at 16.

**Get function(page number):**

If (the page number exists on the tlb table){

Find and return the frame number corresponding to the page number in the tlb table.

}else{ return -1}

**Put function (page number):**

Remove the oldest item from the tlb list.

If (the value of the item exists) { Remove the corresponding value from the tlb table. }

Add the page number to the list and add the page number and frame number to the tlb table.

**PageTable.java**

The page table class consists of a page table and its functions. When the table was first created, the table had 256 entries of 2^8. Each entry consists of a page table item object with a frame number and a valid bit and is initialized as -1,false for the first time. Each entry is operated in the form of a list.

**Function get (page number):**

Refer to page number of page table to return frame number. If the frame does not exist, return -1.

If (frame number -1){return -1}

Return the frame number.

**Function add(page number, frame number):**

Store the frame number in the entry corresponding to the page number and pass the effective bit to true.

**class PageTableItem:**

The PageTableItem class stores the frame number and the effective bit. When initialized, the frame number (-1) and the value bit (false) received by the parameter are initialized.

**Function getFrameNumber:**

return frame number

**PhysicalMemory.java**

The PhysicalMemory class stores frames in the form of a list and tracks the number of free frames. The number of frames is 256 and is created with a free frame variable on the constructor.

**Function addFrame(frame):**

Create a new frame as a parameter in the location corresponding to the current free frame and make the free frame variable +1. Returns the newly created free frame.

**Function getValue (Frame number, Frame offset):**

Refer to the frame through the frame number and return the data corresponding to the offset position.

**Class frame:**

The Frame class represents the frame of the physical memory. This class has variables that store data, and the constructor stores the data list in a frame of 2^8.

**BackStore.java**

This class roles BackingStore.

**Function getData (page number):**

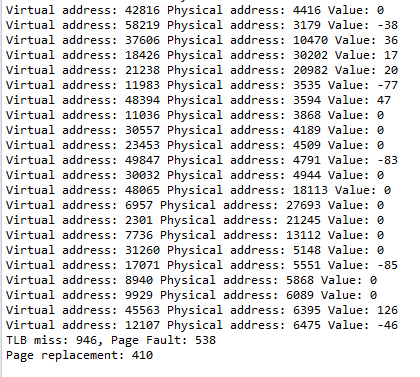
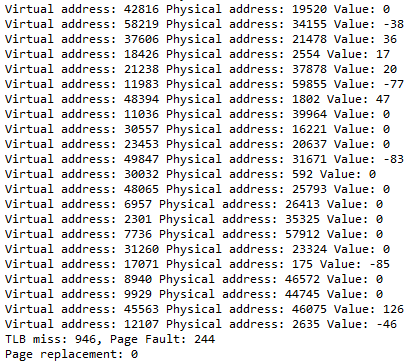
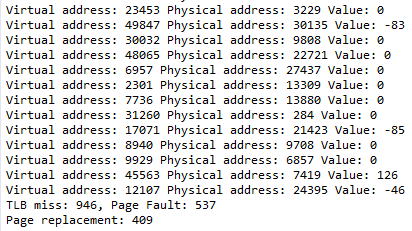
Initialize the list to store data as byte type and the list to store int form data. The size of each list is 2^8, which is the size of the page

Read the "BACKING\_STORE" file stored in the project folder using the RandomAccessFile object. RandomAccessFile allows access to any location in the file.

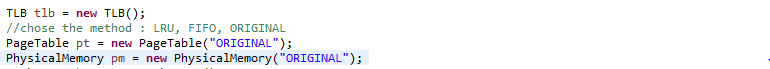
Use this object to move to location of a backing store file with page number\*2^8. Received data is stored in a byte-type list

After that, the data in the byte format is saved back to the list as the int value and then returned.

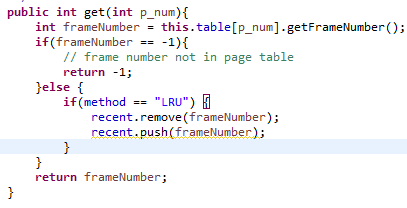
This part is for procedure 9, implement of Modifications parts.

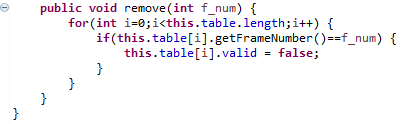
 Original FIFO LRU

Compared to the original, page faults were generally increased and more occurred in FIFO than in LRUs. Page replacement also occurred more in FIFO than in LRU. Because the page tables is maintained same, the TLB miss is the same for all three algorithms.

Through this parameter, can select method.

The addFrame function of the physical memory class is divided into three modules. For FIFO, when the entry of 128 is full, the first incoming frame is removed through the victim frame variable. At this time, the entry in the page table also changes the valid bit of the existing entry to false. Then, add 1 to the victim Frame variable to keep it. For LRUs, track the use of frame through Stack. The victim Frame is determined by the item at the first of the Stack. Changes an existing page table like FIFO, and push the used frame num to stack.





In the pageTable class, get function removes item from Stack for LRU and enters item at the top of Stack to maintain latest use.

When remove pageTable entry, change the valid bit to false.