Virtual Memory Manager

This project is a simulation of a Virtual Memory Manager implemented in Java. It includes components like a page table, TLB (Translation Lookaside Buffer), and backing store to simulate memory management in an operating system. The program translates logical addresses to physical addresses and handles page faults by loading pages from a backing store when necessary.

Features

- Page Replacement: Supports FIFO and LRU replacement policies for managing limited physical frames.
- TLB (Translation Lookaside Buffer): Uses FIFO or LRU to manage a small cache of page-to-frame mappings.
- Backing Store: Simulates secondary storage to load pages into physical memory upon page faults.
- Statistics: Tracks and reports page fault rate and TLB hit rate.

Directory Structure:

```
virtual-memory-manager/
├── src/                          # Source files
│   ├── virtualmemory/            # Main package
│   │   ├── Main.java             # Entry point
│   │   ├── core/                 # Core components
│   │   │   ├── AddressTranslator.java
│   │   │   ├── MemoryConfiguration.java
│   │   │   └── MemoryStatistics.java
│   │   ├── components/           # Subpackage for components
│   │   │   ├── PageTable.java
│   │   │   ├── TLBCache.java
│   │   │   └── PhysicalMemory.java
│   │   └── utils/                # Utilities
│   │       ├── AddressResult.java
│   │       └── BackingStore.java
├── bin/                          # Compiled .class files
├── input/                        # Input files
│   ├── InputFile.txt             # Input file containing logical addresses
│   └── BACKING_STORE.bin         # Binary file simulating the backing store
└── README.md                     # Project documentation
```

Setup Instructions

Prerequisites

Java Development Kit (JDK) version 8 or higher

Compilation

Navigate to the project root directory:

```
cd path\to\virtual-memory-manager
```

Compile all source files:

```
javac -d bin src\virtualmemory\core\*.java src\virtualmemory\components\*.java src\virtualmemory\utils\*.java
```

Running the Program

Once the program is compiled, you can execute it by running the following command from the project root:

```
java -cp bin virtualmemory.Main input/InputFile.txt
```

The output includes:

The logical address, physical address, and value at the physical address for each address in InputFile.txt.

Statistics, including page fault rate and TLB hit rate.

The final state of the page table after processing all addresses.

Configuration Options:
In Main.java, you can configure the following options: (EXTRA CREDIT)

- useLimitedFrames: Set to true to use 128 frames instead of 256.
- useLRUForPages: Set to true to use LRU for page replacement; otherwise, FIFO will be used.
- useLRUForTLB: Set to true to use LRU for TLB replacement; otherwise, FIFO will be used.

Contributions:

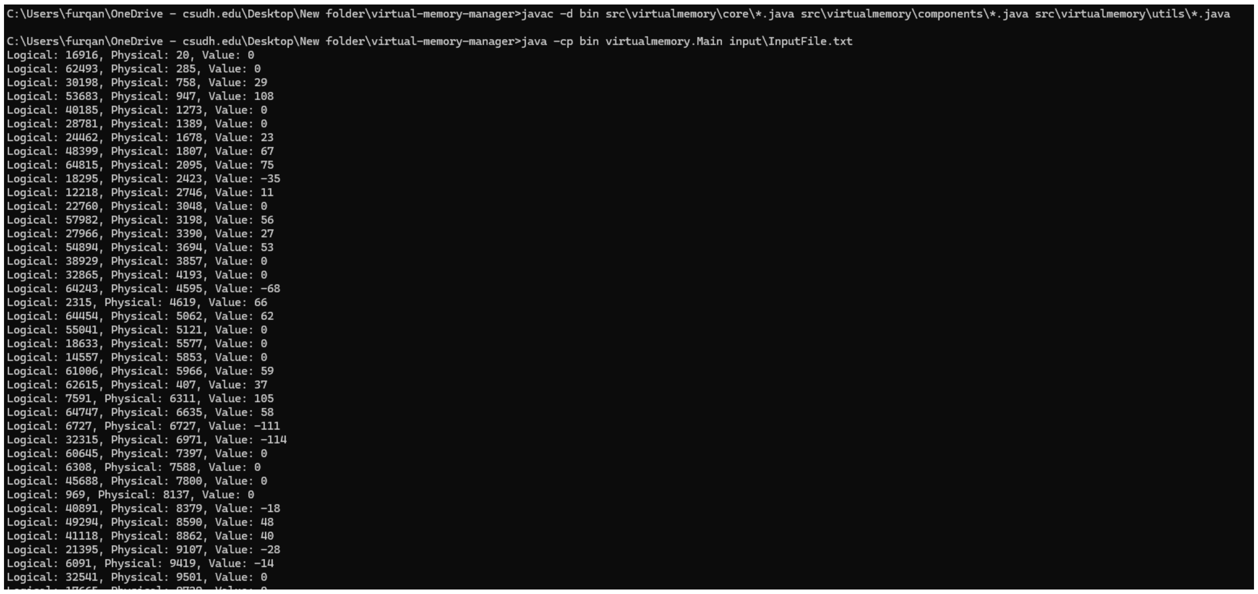
Mohammed Furqan: Memory Management Components & TLB

- Implemented Memory Configuration:Created the MemoryConfiguration class to store settings related to memory size, frame size, and page replacement policies.
- Designed and Implemented Page Table:Developed the PageTable class to manage mappings between page numbers and frame numbers.
- Implemented both FIFO and LRU page replacement policies within the PageTable class for managing limited physical frames.
- Built Physical Memory Simulator:Created the PhysicalMemory class to simulate physical memory, including methods for allocating and writing to frames.
- Developed TLB Cache:Implemented the TLBCache class with support for FIFO and LRU replacement policies.
- Optimized TLB lookups to improve address translation speed.
- Debugging and Testing:
- Conducted unit tests on the PageTable, PhysicalMemory, and TLBCache classes to ensure correct functionality and adherence to FIFO/LRU policies.

Kaifuddin Ahmed: Address Translation, Backing Store, and Project Integration

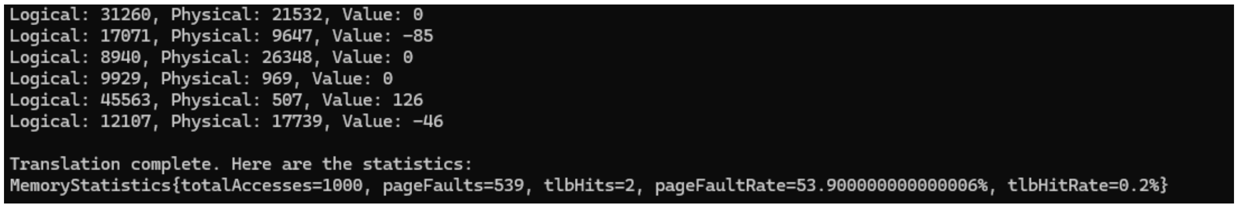
- Created Address Translator:
    - Developed the AddressTranslator class to handle the entire address translation process, coordinating TLB lookups, page table checks, and page faults.
    - Integrated TLB and page table with AddressTranslator to provide seamless translation of logical to physical addresses.
- Implemented Backing Store:
    - Built the BackingStore class to manage reading pages from BACKING_STORE.bin using RandomAccessFile.
    - Ensured efficient page loading and handling of page faults by simulating access to secondary storage.
- Memory Statistics Tracking:
    - Designed the MemoryStatistics class to track key statistics, including total accesses, page faults, and TLB hits.
    - Calculated and reported the page fault rate and TLB hit rate.
- Developed Main Program and Integration:
    - Created Main.java as the entry point for the program, integrating all components and managing user input.
    - Handled input file reading, address translation output, and final statistics reporting.
- Documentation and README:
    - Wrote the README.md file, including setup instructions, usage examples, and troubleshooting information.
    - Documented configuration options and provided a sample output for users.

Output:



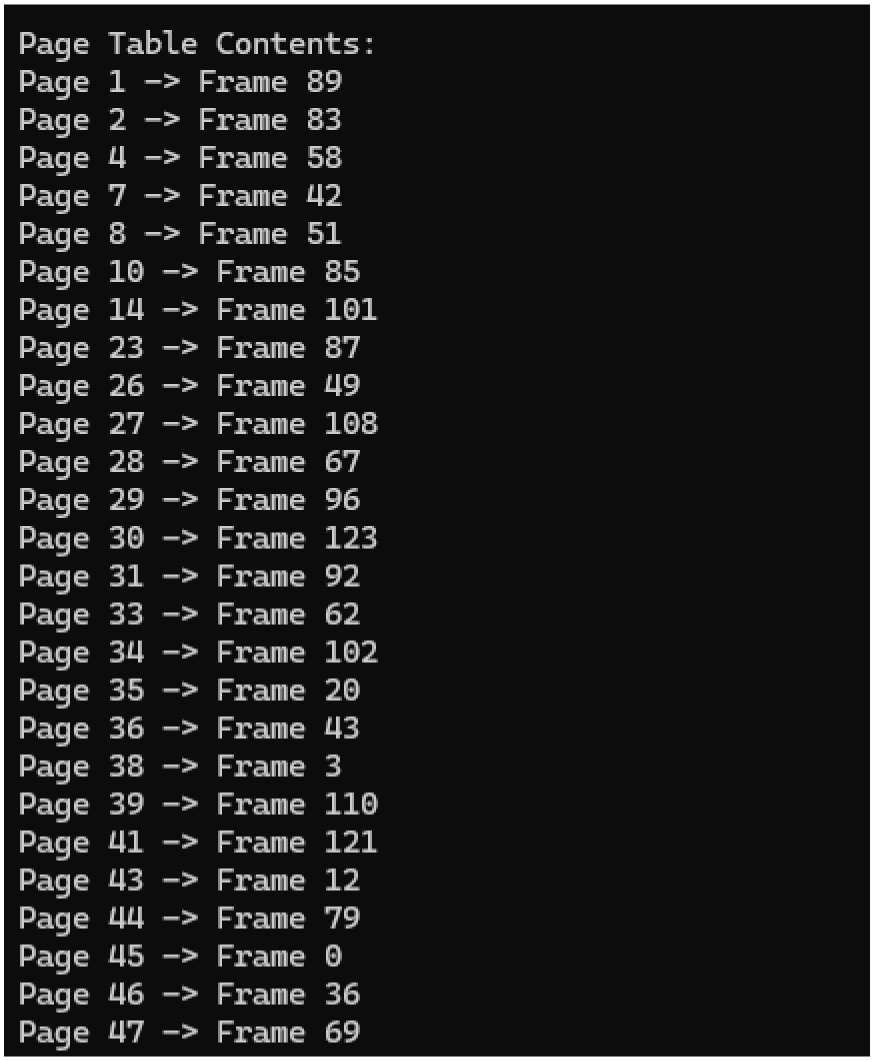
```
C:\Users\furqan\OneDrive - csudh.edu\Desktop\New folder\virtual-memory-manager>javac -d bin src\virtualmemory\core\*.java src\virtualmemory\components\*.java src\virtualmemory\utils\*.java

C:\Users\furqan\OneDrive - csudh.edu\Desktop\New folder\virtual-memory-manager>java -cp bin virtualmemory.Main input\InputFile.txt
Logical: 16916, Physical: 20, Value: 0
Logical: 62493, Physical: 285, Value: 0
Logical: 30198, Physical: 758, Value: 29
Logical: 53683, Physical: 947, Value: 108
Logical: 40185, Physical: 1273, Value: 0
Logical: 28781, Physical: 1389, Value: 0
Logical: 24462, Physical: 1678, Value: 23
Logical: 48399, Physical: 1807, Value: 67
Logical: 64815, Physical: 2095, Value: 75
Logical: 18295, Physical: 2423, Value: -35
Logical: 12218, Physical: 2746, Value: 11
Logical: 22760, Physical: 3048, Value: 0
Logical: 57982, Physical: 3198, Value: 56
Logical: 27966, Physical: 3390, Value: 27
Logical: 54894, Physical: 3694, Value: 53
Logical: 38929, Physical: 3857, Value: 0
Logical: 32865, Physical: 4193, Value: 0
Logical: 64243, Physical: 4595, Value: -68
Logical: 2315, Physical: 4619, Value: 66
Logical: 64454, Physical: 5062, Value: 62
Logical: 55041, Physical: 5121, Value: 0
Logical: 18633, Physical: 5577, Value: 0
Logical: 14557, Physical: 5853, Value: 0
Logical: 61006, Physical: 5966, Value: 59
Logical: 62615, Physical: 407, Value: 37
Logical: 7591, Physical: 6311, Value: 105
Logical: 64747, Physical: 6635, Value: 58
Logical: 6727, Physical: 6727, Value: -111
Logical: 32315, Physical: 6971, Value: -114
Logical: 60645, Physical: 7397, Value: 0
Logical: 6308, Physical: 7588, Value: 0
Logical: 45688, Physical: 7800, Value: 0
Logical: 969, Physical: 8137, Value: 0
Logical: 40891, Physical: 8379, Value: -18
Logical: 49294, Physical: 8590, Value: 48
Logical: 41118, Physical: 8862, Value: 40
Logical: 21395, Physical: 9107, Value: -28
Logical: 6091, Physical: 9419, Value: -14
Logical: 32541, Physical: 9501, Value: 0
```



```
Logical: 31260, Physical: 21532, Value: 0
Logical: 17071, Physical: 9647, Value: -85
Logical: 8940, Physical: 26348, Value: 0
Logical: 9929, Physical: 969, Value: 0
Logical: 45563, Physical: 507, Value: 126
Logical: 12107, Physical: 17739, Value: -46

Translation complete. Here are the statistics:
MemoryStatistics{totalAccesses=1000, pageFaults=539, tlbHits=2, pageFaultRate=53.900000000000006%, tlbHitRate=0.2%}
```



```
Page Table Contents:
Page 1 -> Frame 89
Page 2 -> Frame 83
Page 4 -> Frame 58
Page 7 -> Frame 42
Page 8 -> Frame 51
Page 10 -> Frame 85
Page 14 -> Frame 101
Page 23 -> Frame 87
Page 26 -> Frame 49
Page 27 -> Frame 108
Page 28 -> Frame 67
Page 29 -> Frame 96
Page 30 -> Frame 123
Page 31 -> Frame 92
Page 33 -> Frame 62
Page 34 -> Frame 102
Page 35 -> Frame 20
Page 36 -> Frame 43
Page 38 -> Frame 3
Page 39 -> Frame 110
Page 41 -> Frame 121
Page 43 -> Frame 12
Page 44 -> Frame 79
Page 45 -> Frame 0
Page 46 -> Frame 36
Page 47 -> Frame 69
```