Aida Bumbu

Elie Kheirallah

Programming Project 4: Virtual Memory Manager

Operating Systems

To Dr. Trabelsi

Due on 13/4/2018

“We certify that this submission is our original work and meets the faculty’s expectations of originality”.

Objectives:

- Create a virtual memory manager.
- Make use of Page Tables, TLBs to map logical addresses to physical addresses.
- Re-create the program with the physical address capacity being half the size of the logical addresses capacity.

Flow:

```
//Green is used for elements only used in Part 1
//Red is used for elements only used in Part 2
//Black is used for elements used in both Part 1 and Part 2

Take last element of bin file through fseek function
Using ftell, get the index of that element.
Read from addresses file and store addresses in a vector addressEntries
Allocate memory to buffer which points to elements of bin file
Store every element of the bin file into an array of characters
Go through each address and do the following:
    Find position of current address of  addressEntries.
    Mask number with 65535 and shift it by 8 bits to the right to find pageNum
    Mask number with 255 to find offset
    Search in TLB if pageNum is stored.
    If TLB-miss
        Search in pageTable if value of pageNum has a frame
        If page fault
            Increment pageFault variable that keeps count of number of page faults
            Check if the vector reached 128 arrays (Vector is full)
            If full, overwrite elements and change the value of frame in page table to -1, to show that it was modified
            Point to physical memory at the next free frame
            Fill the physical memory with the values from the bin file
            Update the TLB to contain the last added element using LRU algorithm
            Update the pageTable: go to index pageNum and add the frame
            Calculate physical address by multiplying frame with 256/128 and adding the offset
            Print the virtual address, the physical address, and the value
            Increment the next free frame as the previous last free frame is now filled
```

```
                        Check if the vector is now full. If full, set the next frame to the beginning (FIFO)
                If page was found in the page table
                        Retrieve the frame from the page table
                        Update TLB as was done previously
                        Print the output as was done previously
        If TLB-Hit
                Retrieve frame from the TLB
                Point to location of that frame in the physical memory
                Update the TLB as was done previously
                Print output as was done previously
Output the total page faults, the page fault rate, the TLB hits and the TLB hit rates to a file
```

LRU Implementation (update TLB):
Works like FIFO at beginning
If an element exists in the table and the same element gets passed again, change its position to the first place.
That way the pages that are used more will remain close to the top but will not stay there indefinitely just because they got called multiple times. Their time before they get kicked out is prolonged since these pages will have a longer way down to the bottom of the page.

Conclusion

The program runs with user-defined test cases. A virtual memory manager that translates logical to physical addresses for a virtual address space of size 65 536 bytes was implemented. The program reads from a file containing 1000 addresses. The page number and the offset are retrieved by masking the address number. The logical address is translated to its physical address using a TLB and a page table. If there is a page fault and a table miss, a 256-byte page from the file BACKING_STORE is stored in a page frame into the physical memory. The TLB is then updated using LRU page-replacement strategy. The page table is also updated. For the first part of the program, the size of physical memory is the same as the size of virtual address space. A page-replacement strategy is not required. However, for the second part, the physical address space uses 128 page frames instead of 256. A FIFO page replacement policy is used in the physical memory. Also, the program is modified to keep track of free page frames. The value stored in the physical address will then be output. The page-fault rate and TLB hit rate will be shown in the output file generated by the program. Since the logical addresses were generated randomly, the TLB hit rate is low (0.054%). The second part of the program has a higher fault rate, because it uses less page-frames in the physical memory.