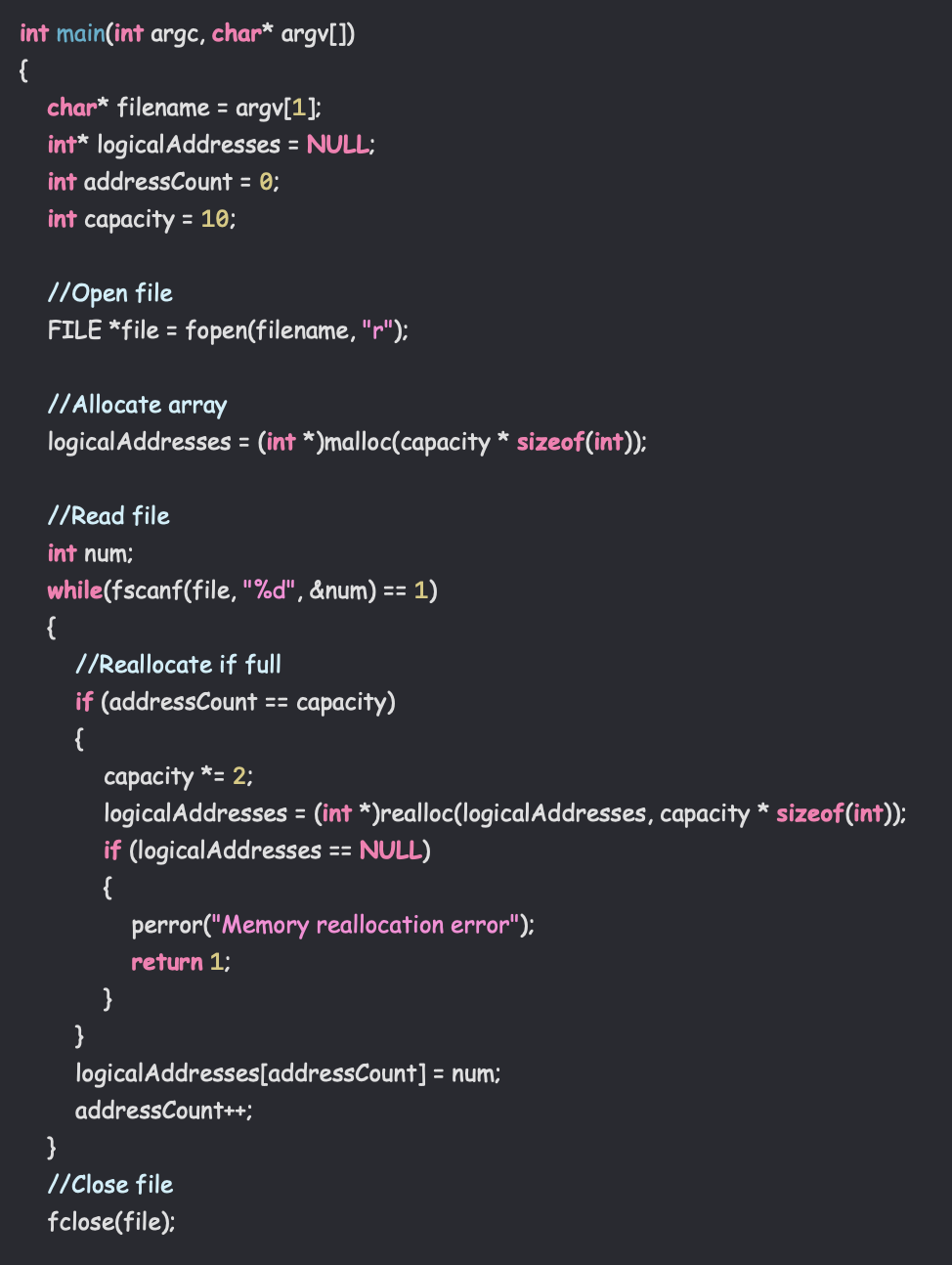
Stephanie Contino, Emmanuel Pasteur

Project 3 Report

CSC 345-01

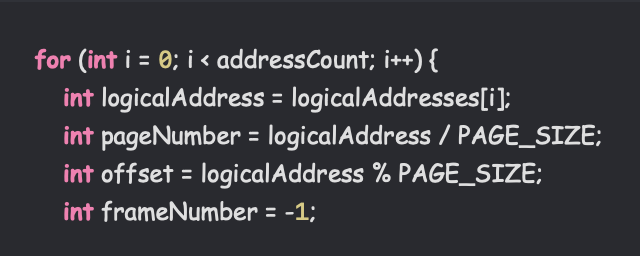
11/10/23

• Correctly read in input logical addresses:



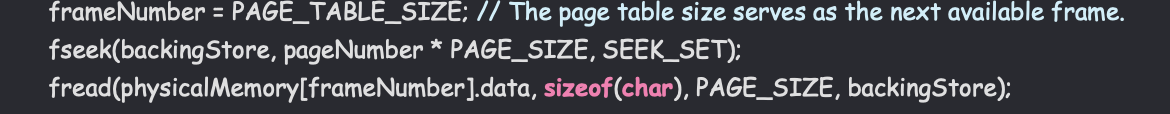
To read in our input addresses, we start off by reading only 10 addresses at a time to simulate a dynamic array. This involves initially allocating 10 spaces in the array. While there are numbers in the array, we check to see if we have hit capacity. If so, we reallocate the array and increase the capacity by another 10 addresses. Otherwise, we go directly into reading the address from the file into the array, and increment a counter so we know how many addresses we have.

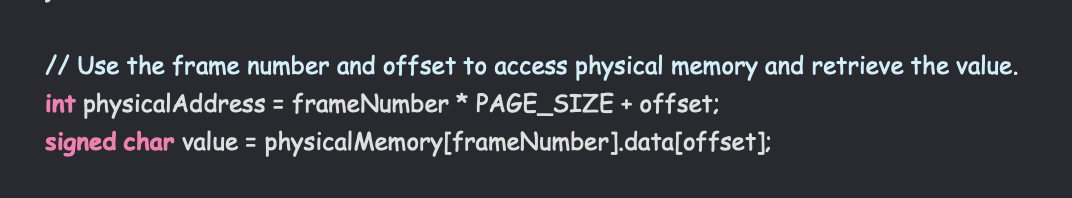
• Correctly translated the input addresses into physical addresses.



To translate the addresses, we declare the constant PAGE\_SIZE at the start of the program to be 256. When we retrieve the logical address from the array, we convert it into its page number by dividing it by the page size. To get the offset, we take the modulo with the page size. This guarantees that the pageNumber and offset variables will always be within the bounds of the page size.

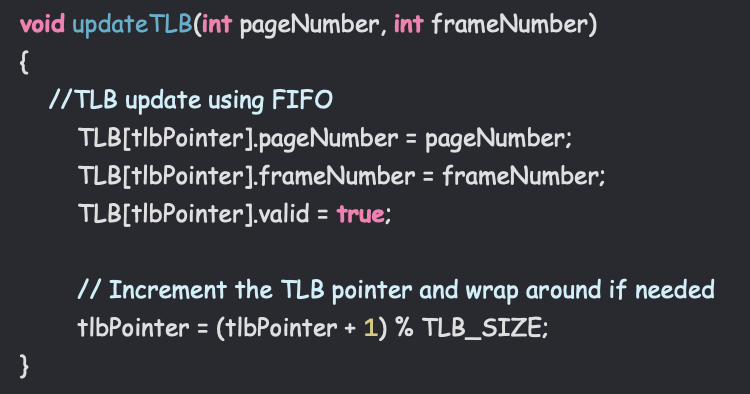
• Correctly retrieved the values stored in the physical addresses.





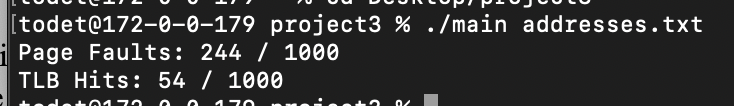
To retrieve the value stored in the physical address, we first have to save it to the physical address. We use fread to read into the physical memory and store the value. To retrieve, we create a value variable that stores the value found at the address. We then look at the location in the physicalMemory array at our current frame, and then look at the data stored in the offset.

• Implemented FIFO-based **TLB** update (main)



To implement the TLB update, we made a function called updateTLB that takes the current page and frame number as parameters. We use the TLB pointer to keep track of where we should be in the TLB. Updating the TLB involved updating the page number and frame number individually, as well as changing the valid bit to true with the update. We then increment the TLB pointer after the update. This is where the FIFO part comes in. If we are out of bounds of the TLB size, we need to start over at the beginning. To do this, we take the modulo of the pointer + 1 with TLB\_SIZE, which creates a FIFO update pattern.

• Correctly counted number of page faults and TLB hits

For main:

For main\_pr:

• Implemented FIFO-based **page replacement** algorithm (main\_pr). This should include TLB update implementation from the above.

• Try different number of frames. Start with 32, double up each time, up to 512. That is, 32, 64, 128, 256, 512, 768, 1024. Record absolute number of page faults and the page fault rate for these five cases. In your *report.pdf*, include the **plot** showing the trend. Also include your analysis of the result.