ADA University
School of Information Technologies and Engineering
CSCI 3510: Principles of Operating Systems
Spring Semester, 2024

Operating Systems Project

Project team: <Team Name>
Instructor: Mr. Umid Suleymanov

Submitted in partial fulfillment of the requirements of
the CSCI 3510: Principles of Operating Systems course
Deadline: May 1, 2024, 23:59

Version date	Version information
<Date>	Initial draft of the System Proposal
<Date>	<Version description>

Team member	Contribution to the homework	Estimated %
<Student Name 1>	<Description of the work contributed>	<X>%
<Student Name 2>	<Description of the work contributed>	<X>%
<Student Name 3>	<Description of the work contributed>	<X>%
<Student Name 4>	<Description of the work contributed>	<X>%

ADA University
School of Information Technologies and Engineering
CSCI 3510: Principles of Operating Systems
Spring Semester, 2024

CSCI 3510: Principles of Operating Systems

For this project, you are expected to design and implement a simulated virtual memory manager. You can compensate for your missed scores by completing extra tasks.
The parameters of the simulated manager are:

- Program address space: 16-bit
- Page size: 2^8 bytes
- TLB capacity: 16 entries
- Number of frames: 256
- Physical memory size: 64 KB

Grading Components

This assignment has multiple components:

1. Address translation (10%)

You are expected to translate logical addresses given one per line in an input file named "addresses.txt" into physical addresses using the TLB and the page table. Given a logical address, your program takes the following three steps to perform address translation.

1. Extract the page number from the logical address.
2. Access the TLB using the extracted page number.
    a. If the page entry is in the TLB, obtain the frame number from the TLB.
    b. Otherwise, follow step 3.
3. Access the page table.
    a. If the page is in the main memory, obtain the frame number from the page table.
    b. Otherwise, a page fault occurs.

2. Page fault (40%)

You must implement paging in your simulated virtual memory system. Your virtual memory simulator will perform the following three steps when a page fault occurs:

1. Read the relevant page from the disk file into the physical memory.
2. Update the page table.
3. Update the TLB.

The hard disk drive is simulated by the file named "disk_sim", which is provided and is a binary file of 64 KB. The `disk` file is accessed in a random-access fashion, where your virtual memory system can randomly seek to a certain position of the file and read a page. In other words, you are not supposed to read in the whole file. You will need to implement simulated physical memory which initially will be empty.
Your system is expected to handle page faults using the page table and the TLB which you will need to implement. Initially, both the TLB and the page table will be empty. You

can choose any suitable data structure of your choice. You must implement FIFO replacement strategy for cases when the TLB is full and needs to be updated.
You can also experiment with different TLB capacities.

3. **Output (30%)**

Your solution is expected to produce an output file named "log.txt" with the following information:

- System information
- Virtual address, translated physical address, value at the translated address on one line per given address
- Page fault rate (percentage of page faults)
- TLB hit rate (percentage of TLB hits)

4. **Style (10%)**

Your code should meet the following criteria:

- It should be function oriented. You are expected to build several decoupled reusable functions each with a single responsibility. You should avoid scripting (if coding in Python) and throwing all functionalities in the main().
- It should be readable.
- It should be well-commented.
- It should have low modifiability cost (good use of constants, arguments, and macros).

5. **Presentation (10%)**

You will be expected to present your implementation during the final session of the term. Your presentation should answer the following questions:

1. How have the simulated page table, physical memory, and the TLB been implemented?
2. When a TLB miss occurs, how does the system decide which entry to replace?
3. Can the parameters such as the size of the TLB be changed? How easily?
4. What experiments have you run? What different techniques have you tried?
5. Does your program only load pages from the backing store when they are needed?
6. Does your solution allow the physical address space to be smaller than the virtual address space?
7. Does your code follow good style practices?

6. **Extra 1 (10%)**

Your simulation should handle the case where the physical memory is less than the virtual memory. You are expected to implement the second chance page replacement algorithm for this purpose. As the size of the physical memory, you can have 16 KB with all the other parameters staying the same.

7. Extra 2 (5%)

You are expected to implement the LRU replacement strategy to optimize TLB hit rate. Once implemented, a comparative analysis of the two strategies must be conducted and the results must be interpreted.

Submission

Please follow the instructions below when you make your submissions to the Blackboard System:

- The language of choice for the project must be one of C/C++/Java/Python.
- The team is expected to submit a single compressed archive file with all relevant files. It must include source code.
- Name the file according to the template (all capital):

CSCI3510_2024S_PROJECT_TEAMNUMBER

- Any team whose solution arises questions, will be asked for explanations.