Programming Assignment 1 - Dil ki baat; Cache ke saath
CS 773: Computer Architecture for Performance and Security, Spring 2025
Computer Science and Engineering
Indian Institute of Technology, Bombay
CASPER group: https://casper-iitb.github.io/

** Disclaimer: The names and dialogues included in this document are intended solely for fun learning. They are not meant to offend, mislead, or be taken as factual information. Please enjoy them in the spirit of fun learning.*

Welcome to the city of computer architects, where houses are called cores, and innovation bridges the gap between them. This is the tale of Raj, a very shy boy living in one core, and Simran, the girl he loves, residing in another core. Your mission is to help Raj win the heart of Simran. However, it won't be an easy task, as the town operates in peculiar ways. You will uncover these mysteries and guide Raj through his journey.

Part 1: Tujhe dekha toh ye jana sanam …		
1A	Understanding Side Channel Attacks	0 points
Part 2: Choti Choti Baatein …		
2A	Pigeon: Flush-Reload covert channel attack	5 points
2B	Postman: Occupancy-based covert channel attack	5 points
Part 3: Dil ki Baatein …		
3A	Share a heart	2.5 points
3B	Sing along	2.5 points

***Just a friendly reminder**: if you think copying code is a clever shortcut, think again. It's not only easily spotted but also a great way to miss out on the chance to actually learn something. Why not impress us with your own work?*

Part 1: Tujhe dekha toh ye jana sanam …

In this town, people living in one core do not talk directly with those living in another core. However, there exists a unique method to understand what's happening in another core without direct communication—this is known as a **side-channel attack**.

1A. Understanding Side Channel Attacks

Raj needs to master the art of understanding how a side-channel attack works. So you are encouraged to understand the concepts and workings of side-channel attacks to assist Raj in his journey.

Here is a helpful resource for this:
Reference Link: https://github.com/cs773-spring-2024-25/pa1

This resource includes an implementation of the side-channel attack using the **Flush+Reload** technique. Students should carefully study and understand this implementation, as it will be crucial for completing the next part of the assignment and helping Raj progress.

Raj has a few questions. He needs your help in answering them. (These questions are for you to consider when referring to the code. You will NOT be graded in this part.)

1. On which cache is the flush-reload attack being done and why?
2. How do we detect a bit using a flush-reload attack?
3. How do you find the hit/miss threshold of the cache?
4. How are the two processes synchronized?
5. What memory is being shared among the processes?

Part 2: Choti Choti Baatein …

*"Bade bade sheheron mein esi **choti choti baatein** hoti rehti hai senorita"*

Now that Raj has learned to establish a covert channel between different cores, it's time to do *choti choti baatein* by sending a letter (text file) to Simran. Make use of the covert channel to send this letter. You only have two modes of communication - Pigeon (Flush-Reload covert channel) and Postman (Occupancy covert channel).

2A: Pigeon: Flush-Reload covert channel attack

Here, you must help Raj mount a Flush-Reload covert channel attack and transmit a small text file to Simran. However, we have specific requirements for successful reception:

Bandwidth Threshold: A minimum data transfer rate to ensure timely reception.
Accuracy Tolerance: The transmitted message must maintain a low error rate. For example, "Love" cannot be incorrectly received as "Hate".

Raj must successfully transmit the text file to Simran through the Flush-Reload covert channel while meeting both bandwidth and error rate requirements.

Note: The data transmission must be only through the cache covert channel and NOT anything else (pipes, shared memory, sockets, etc). You are allowed to use shared memory, but you cannot directly write and read from it to transmit the data.

2B: Postman: Occupancy-based covert channel attack

Now, you must help Raj send another letter to Simran through postman (i.e., an occupancy-based attack). Again, there are bandwidth and error rate requirements for successful reception.
Raj must successfully transmit this text file to Simran through the Occupancy-based covert channel while meeting bandwidth and error rate requirements.

Note: No memory should be shared between sender and receiver processes in the occupancy-based attack.

Deliverables for Task 2A and Task 2B:

- The file sender.c represents Raj living in one core. This file will be responsible for initiating the covert channel and transmitting the letter.
- The file receiver..c represents Simran living in another core. This file will be responsible for receiving the letter.
- Send the letter msg.txt written by Raj to Simran using the covert channel (FLUSH + RELOAD attack) and Occupancy-based attack.
- The template code provided at the GitHub repository is the same for Task 2A and Task 2B. Your submission should have two directories, **task2a**, and **task2b**, containing the solutions for both the sub-tasks.

Part 3: Dil ki Baatein …

3A: Share a heart ❤️

Enough letters now; it's time for Raj to do a heart-to-heart communication with Simran. In this task, Raj needs to send a picture of a heart to Simran using the covert channel (FLUSH + RELOAD attack) and Occupancy attack.

Unfortunately, Raj accidentally deleted Simran's number (in our case it means there is no agreement on which memory addresses to do the FLUSH + RELOAD attack on). So he cannot send messages directly. Simran thought of a clever way to send Raj her contact number - using an occupancy covert channel. But they need your help in figuring out the details.

Technical details: In Part 2 most of you would have used a file-backed **mmap** to create a shared memory and then used specific addresses in that memory to create a covert channel. In this part, you will have to share the name of the file first using an occupancy attack and then use it to mount a FLUSH + RELOAD attack. If you have used any other method to create a shared memory, that is fine too. But you cannot assume that both processes know the location of shared memory to create a covert channel. You should assume both processes have zero idea about shared memory locations and then share this info using the occupancy covert channel. Later you have to use the FLUSH + RELOAD attack to share the actual file (in this case a heart image). [Something to think about: Why not use the occupancy channel to share the heart image?]

Here's a link to the image:
https://drive.google.com/file/d/18yimZ24cFlDRslyk6RIi2rkUPM6S-f6S/view?usp=sharing

---

3B: Sing along

After successfully sending the heart image to Simran using the FLUSH + RELOAD attack, Raj realizes he also wants to share his favorite song with Simran. Since they have already established a shared memory file using the Occupancy Based Covert Channel, they can reuse the same shared memory to transmit the audio file using the FLUSH + RELOAD attack.

Here's a link to the audio file:
https://drive.google.com/file/d/19A5Mxml7R6xXP4gn-uZjLwLGizIziNZM/view?usp=sharing

Note: Use the same template code from part 2 for this task.

Deliverables for Task 3A and Task 3B:

- Both FLUSH + RELOAD and occupancy attacks need to be implemented in this task, as directed in the problem statement above.
- Put your code inside a directory task3
- The image and the audio transmitted should have a minimum error rate. The heart in the image should be visible and the audio should be as clear as possible!

Deadline: Feb 15, 2025 4:45 PM GMT+5:30

All the best!

Submission Guidelines

We have provided the submission instructions in the README file over here. Please fill the README file in your submission as directed.

Directory Structure

Folders - **task2a**, **task2b**, **task3a** and optionally **task3b**. Each of these directories should contain:

1. sender.c
2. receiver.c
3. utils.c
4. utils.h
5. MakeFile
6. Additional relevant files

If the implementation for task3a and task3b are the same, put the code in the folder task3a. Apart from the above folders, there should be a completely filled **README** file.

Your submission should be **.zip** file with **<roll-number>.zi**p. Only one team member should submit, or else there will be penalties.
