# CSE31: Lab #3a – Cache

## Overview

In this lab, we will investigate how cache is organized in a CPU.

## (Exercise) Direct-map cache

**TPS (Think-Pair-Share) activity 1** Paired with the classmate(s) sitting next to you and answer the following questions (You are allowed to form a group of three):

1. What is cache? Why do we need cache?
2. There are generally 2 practical ways to organize a cache: Direct-mapped cache and N-way set associative cache. In both types of cache, data at a specific address of the main memory (RAM) are mapped to a pre-defined location in the cache. A **"Block"** is the basic unit of data being mapped between the main memory and cache. The size of a block depends on the specification of a cache. Every time data are transferred between cache and the main memory, it is **a block of data** being transferred. In this exercise, we will explore the Direct-mapped cache.
3. In a Direct-mapped cache, the cache is organized as a hash table. Addresses of the main memory are mapped to the indices of the cache (block numbers) using a *modulo* operator (%) as the hash function. As a result, we can divide a memory address into 3 fields: tag, index, offset.
4. Offset bits tell us how many bytes of data are in a block. These bits are the ***right-most bits of the memory address***. You can consider this as the **number of columns of data** in a cache. With a specific value of the offset bits from an address, we know which column of a block we are trying to access. Given the block size of a cache is 16B (bytes), how many bits do we need for offset? What is the number of bits in offset as a function of block size? Is it practical to have a cache of block size = 1 byte?
5. Index bits tell us how many blocks there are in a cache. These bits are ***the next right-most bits of the memory address after offset bits***. You can consider this as the **number of blocks (rows) of data** in a cache. With a specific value of the index bits from an address, we know which block (row) we are trying to access. Given there are 64 blocks in a cache, how many index bits

do we need? What is the number of bits in index as a function of number of blocks?

6. Once you know the number of blocks and the block size of a cache, do you know the total size of the cache? How?
7. Since the size of cache is always smaller than the size of the main memory, the sum of bits of the offset and index of a cache will be less than the number of bits in an address of the main memory. What do we do to the left over bits from the address? Why are they important?
8. Given a memory address of 20 bits (during Intel 8086 era), 128B of cache, and 8B block size, answer the following questions:
    a. How big is this main memory?
    b. How many offset bits?
    c. How many blocks are there in the cache?
    d. How many index bits?
    e. How many tag bits?
    f. Draw the layout of the cache: including tags, valid bits, dirty bits, and data blocks.
    g. What is the number of bits per row of the cache (number of bits being used in a row: tag, valid bit, dirty bits, and data block)?

## (Exercise) N-way set associative cache

**TPS (Think-Pair-Share) activity 2** Paired with the same classmate(s) sitting next to you and answer the following questions:

1. What is the disadvantage of a Direct-mapped cache? What kind of cache miss will it introduce?
2. To overcome this problem, we can allow multiple blocks of data to occupy the same ***set*** of a cache. Note that we use ***"set"*** here instead of index of cache. In this organization, we group N blocks (rows) of cache into a set and allow more than one block of data to stay within a set. The layout of the cache remains the same as its direct-mapped version, but the difference is that every N blocks are now being grouped into a set.

3. The memory address is still partitioned into the same 3 fields, but the index bits now refer to the set number. Given a cache with 1024 blocks and the associativity is 4 (4 blocks per set), how many index bits do we need? What is the number of bits in index as a function of number of blocks and associativity?
4. Given a memory address of 20 bits (during Intel 8086 era), **128B** of **2-way** cache, and **8B block size**, answer the following questions:
    a. How big is this main memory?
    b. How many offset bits?
    c. How many blocks are there in the cache?
    d. How many sets are there in the cache?
    e. How many index bits?
    f. How many tag bits?
    g. Draw the layout of the cache: including tags, valid bits, dirty bits, and data blocks. Indicate the sets with a different color (or a thicker) boarder.
    h. What is the number of bits per row of the cache (number of bits being used in a row: tag, valid bit, dirty bits, and data block)?

---

## (Assignment, individual) Cache in your computer

Download and install CPUID and find out details about the cache(s) in your computer.

For Windows: CPU-Z: https://www.cpuid.com/softwares/cpu-z.html

For Mac: MacCPUID: https://software.intel.com/en-us/download/download-maccpuid

For Linux: https://www.tecmint.com/check-linux-cpu-information/

Answer the following questions:

1. How many levels of caches does your CPU have (L1, L2, L3, etc.)? Is there separate L1 cache for data and instructions?
2. How big is each level of cache?

3. What is the block size (sometimes it is called line size)?
4. Are the caches direct-mapped or set associative? If set associative, how many ways?
5. With L1 data cache, how many tag bits, index bits, and offset bits?

## What to submit

When you are done with this lab assignments, you are ready to submit your work. Make sure you have included the following ***before*** you press Submit:

- Your answers to ***assignment***, ***TPS activities*** in a *text file*, and a list of Collaborators.