**COEN 210 Project #1, Spring 2018**

In this project, you are going to model an instruction cache in any programming language of your choice, You are given a file, **inst\_addr\_trace\_hex\_project\_1.txt**, which has instruction addresses as it is executed by a microprocessor. You will need to read the instruction address, one at a time, to decide whether the address is in the cache (hit/miss):

* Each instruction address accesses the instruction cache for hit/miss:
  + Hit: the clock cycle is increment by 1
  + Miss: the clock cycle is increment by 15. The address is now stored in the in the cache, next access on the same cache line (block) should be a hit
  + Note: if the instruction address straddles the cache lines, then 2 accesses are needed. The combination cycles of hit and miss cycles are used
  + The number of bytes read for each instruction address is kept in the file **inst\_data\_size\_project\_1.txt.** The read data for each instruction address are compared to the data in the file **inst\_data\_trace\_hex\_project\_1.txt** which should match
* Cache configuration:
  + Cache size: 8K-Byte
  + Way associative: 1 direct map
  + Cache line size (block size): 16-Byte

Additional file is given:

* **inst\_mem\_hex\_16byte\_wide.txt**: this file is the main memory with 128-bit (16-byte) wide data
  + 0x00000000 is the first line address
  + 0x00000010 is the second line address
  + 0x00000020 is the third line address
  + 0x00000030 is the fourth line address

Upon a cache miss, you will need to access this memory to get the data using the instruction address to read appropriate line for 16-byte of data. The fetched data from the main memory is put into the cache and extract data for the fetched instruction

Note for **inst\_data\_size\_project\_1.txt,** this is the size of data to read from instruction cache. The number of bytes is calculated by dividing the size of data by half:

* 4: 4/2 = 2-byte instruction
* 12: 12/2 = 6-byte instruction
* 16: 16/2 = 8-byte instruction

**Track information in your program**:

* Total number of cache accesses (straddle instruction address counts as 2 accesses)
* Total number of hits, calculate hit ratio
* Total instruction addresses
* Total number of clock cycles, calculate instruction-per-cycle (IPC) = total instruction fetched/total cycles

**COEN 210 Project #1, Fall 2021**

**Extra Credit (+10 points)**

Configuration parameters:

* Cache size: 16K-Byte
* Way associative: 4-way
* Cache line size (block size): 32-byte