Assignment 3

- Group assignment – can be done in groups of 2
- You can choose to do the design in any language you wish, C, C++, python, Verilog etc
- Submit your code (well commented) and output graphs/tables on LMS.
- Rename the filename of your code to <roll_numbers>_filename.< >
- All codes will run through a plagiarism check. Files found similar with get a 0 for the assignment. Repeat offence will attract Grade penalty on the overall grade
- Submit by July 7 2021, 11:59pm
- Marks: 20 (code)

Problem statement

a. Design a direct mapped cache of size 64kilobytes. Block size: 4 bytes. Assume a 32 bit address. Figure out how many cache lines you need.

You need not implement a main memory, which means you do not need to implement data in the cache. (Implementing a 2^32 memory will be impossible!)

Output: You need to report hit/miss rates of the cache for the input memory trace files (5 traces) provided in the next slide.

[Hint: For reporting hit/miss, all you need to check is a tag match, and a valid bit, so there is no need to have data in the cache]

b. Increase the cache size to 512kB and repeat the experiment. Not the change in hit/miss rates

c. Keeping the cache size at 64kB, vary the block size from 1 byte, 4 bytes,8 bytes, 16 bytes, 32 bytes. Note that the number of cache lines will reduce, if you increase the block size keeping the cache size same. Repeat the experiment for all the trace files, and note the hit/miss rates

Input to your code

- Use the memory trace files at this location (https://cseweb.ucsd.edu/classes/fa07/cse240a/proj1-traces.tar.gz) as input.
- The trace file will specify all the memory accesses/addresses that occur in a certain program. Each line in the trace file will specify a new memory reference and has the following fields:
  - Access Type: A single character indicating whether the access is a load ('l') or a store ('s'). You can ignore this field. For reporting hit/miss, it does not matter whether it is a Load/Store
  - Address: A 32-bit integer (in unsigned hexadecimal format) specifying the memory address that is being accessed. This is the only field you need.
  - Instructions since last memory access: Indicates the number of instructions of any type that executed between since the last memory access (i.e. the one on the previous line in the trace). For example, if the 5th and 10th instructions in the program's execution are loads, and there are no memory operations between them, then the trace line for with the second load has "4" for this field. You can safely ignore this field

Output of the code

- Hit rates or miss rates for all the 5 traces for the 3 different experiments can be reported in an excel or in the form of a table/graph.
- Pls submit the graphs/table/excel with your observations on LMS
- We will have you to explain the observations during the demo