Ramiro Gonzalez **Homework/Lab 03a** CSE 140 March 10, 2020

## Using Ubuntu Linux

- Collaborators : Jamal Bills, Nikko Solon

1. What is cache? Why do we need cache?
   - Cache stores data, cache is used for faster access to data.

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
   - In direct-mapping indexing is used to find a block.
   - Direct-mapped cache: Each main memory address maps to exactly one cache block.
   - N-way set associative cache: Cache is divided into groups of blocks

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.
   - tag | index | offset
   - Cache size is $2^n$ so n bits are used for the index
   - Block size is $2^m$ words $(2^{m+2}) bytes$ so m bits are used for the word withing the block, and two bits are used for the byte part of the address

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?
   - We need 4 amount of bits for the offset. $2^4$
   - offset bits = $log_2(blocksize)$
   - No, because a 1 byte cache block size does not take advantage of spatial locality. You will need to load more than one block.

5. Index bits tell us how many blocks there are in a cache. These bits are the next rightmost 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?

   - # index bits = $log_2(64) = 6$
   - # index bits = $log_2(\#blocks)$

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?

   - From the number of blocks and the block size of a cache we can calculate the total size of the cache.
   - AREA(cache size, B) = HEIGHT(# of blocks)*WIDTH(size of one block)
   - Total Size = (# of blocks)*(block size)

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?

   - In direct mapping memory locations may map to the same cache block. Left over bits may be used as tags bits to to distinguish which memory locations maps to which cache block.

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?
   - Main memory is $2^{20}B = 1048576$

   (b) How many offset bits?
   - Offset bits = $log_2(8) = 3$

   (c) How many blocks are there in the cache?
   - Blocks in cache = 128B / 8B = 16 blocks

   (d) How many index bits?
   - Index bits: 16 blocks => $log_2 16 = 4$ bits

   (e) How many tag bits?

- lengthAddress - offset - index = 20 - 3 - 4 = 13 bits

(f) Draw the layout of the cache: including tags, valid bits, dirty bits, and data blocks.

```
Address
[tag][index][offset]
tag —— dirty ——Index 0————————————————————————1023
[valid][dirty][tag][data]
|          .              |
|          .              |
|          .              |
|          .              |
|          .              |
|                         |
__________________________
```

(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)?

- Bits per row in cache: tag bits + bytes/block + bits/bytes + valid bit = 13 + 8*8 + 1 = 78

## Activity 2

1. What is the disadvantage of a Direct-mapped cache? What kind of cache miss will it introduce?
   - A disadvantage is that Main memory maps to a fixed block of cache
   - The kind of cache miss will be a instruction read miss because if more than one memory maps to a fixed block of cache they may continually swap in and out.

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.
   - N blocks (row) of

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?

- index bits: $1024/4 = 256 = 2^8$
- numBits $= \log_2(256) = 8$

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:

   - How big is this main memory?
     - Main memory $= 2^{20}$
   - How many offset bits?
     - Blocks in the cache $= 128/8 = 16$block
   - How many blocks are there in the cache?
     - Number of blocks/associative $= 16/2 = 8$
   - How many sets are there in the cache?
     - 8 sets $= 2^3$ -> 8 bits
   - How many index bits?
   - How many tag bits?
     - addr length - offset - index = 20 - 3 - 8 = 9 bits
   - 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.

```
    [tag]   [valid] [dirty] [data  ]
    |14 bits| 1bit  | 1bit| 64 bits|
```

   - 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)?
     - tag bits + bytes/block + bits/byte + valid bit + dirty bit = 9 + 8*8 + 1 + 1 = 75 bits

## (Assignment, individual) Cache in your computer

1. How many levels of cache does your CPU have (L1, L2, L3, etc)? Is there separate L1 cache for data instructions.

   - It has L1, L2, and L3 L1d, L1i cache

2. How big is each level of cache?

- L1d cache: 32k , L1i cache: 32k, L2 cache: 256k, L3 cache: 6144k

3. What is the block size (sometimes it is called line size)?
   - Block Size: 4096

4. Are the caches direct-mapped or set associative? If set associative, how many ways?
   - L1: Direct, L2: Direct, L2: Direc, L3: Complex

5. With L1 data cache, how many tag bits, index bits, and offset bits?
   - Tag bits 54 bits
   - 64 - 6 - 6 = 52
   - Index bits: 6 bits