April 12

- 6 classes to go!
- Assignment was supposed to be in single precision but if you've already done it in double that is OK.
- Cache

Cache

- **cache** (ksh)
  n.
  1. A hiding place used especially for storing provisions.
  2. A place for concealment and safekeeping, as of valuables.
  3. The store of goods or valuables concealed in a hiding place.
  4. *Computer Science.* A fast storage buffer in the central processing unit of a computer. In this sense, also called **cache memory.**
- *v. tr.* **cached, cach·ing, cach·es.**
  1. To hide or store in a cache.

Cache Analogy

- You are writing a term paper at a table in the library
- As you work you realize you need a book
- You stop writing, fetch the reference, continue writing
- You don't immediately return the book, maybe you'll need it again
- Soon you have a few books at your table and no longer have to fetch more books
- The table is a CACHE for the rest of the library

Cache

- Sits between CPU and main memory
- Very fast table that stores a *TAG* and *DATA*
  - *TAG* is the memory address
  - *DATA* is a copy of memory at the address given by TAG

Tag	Data
1000	17
1040	1
1032	97
1008	11

Memory

1000	17
1004	23
1008	11
1012	5
1016	29
1020	38
1024	44
1028	99
1032	97
1036	25
1040	1
1044	4

Cache Access

- On load we look in the TAG entries for the address we're loading
  - Found ✍ a *HIT*, return the DATA
  - Not Found ✍ a *MISS*, go to memory for the data and put it and the address (TAG) in the cache

Tag	Data
1000	17
1040	1
1032	97
1008	11

Memory

1000	17
1004	23
1008	11
1012	5
1016	29
1020	38
1024	44
1028	99
1032	97
1036	25
1040	1
1044	4

Cache Lines

- Usually get more data than requested
  - a *LINE* is the unit of memory stored in the cache
  - usually much bigger than 1 word, 32 bytes per line is common
  - bigger LINE means fewer misses because of locality
  - but bigger LINE means longer time on miss

Tag	Data	
1000	17	23
1040	1	4
1032	97	25
1008	11	5

Memory

1000	17
1004	23
1008	11
1012	5
1016	29
1020	38
1024	44
1028	99
1032	97
1036	25
1040	1
1044	4

Finding the TAG in the Cache

- A 1MByte cache may have 32k different lines each of 32 bytes
- We can't afford to sequentially search the 32k different TAGs
- *ASSOCIATIVE* memory uses hardware to compare the address to the TAGs in parallel but it is expensive and 1MByte is thus unlikely
- *DIRECT MAPPED CACHE* computes the cache entry from the address
  - multiple addresses map to the same cache line
  - use TAG to determine if right
- Choose some bits from the address to determine the Cache line
  - low 5 bits determine which byte within the line
  - we need 15 bits to determine which of the 32k different lines has the data
  - which of the 32 – 5 = 27 remaining bits should we use?

Direct-Mapping Example

- With 8 byte lines, the bottom 3 bits determine the byte within the line
- With 4 cache lines, the next 2 bits determine which line to use

1024d = 10000**00**000b ✍ line = 00b = 0d

1000d = 01111**01**000b ✍ line = 01b = 1d

1040d = 10000**10**000b ✍ line = 10b = 2d

Tag	Data	
1024	44	99
1000	17	23
1040	1	4
1016	29	38

Memory

1000	17
1004	23
1008	11
1012	5
1016	29
1020	38
1024	44
1028	99
1032	97
1036	25
1040	1
1044	4

Direct Mapping Miss

- What happens when we now ask for address 1008?

1008d = 0111111**10**000b ✍ line = 10b = 2d

but earlier we put 1040d there...

1040d = 1000001**10**000b ✍ line = 10b = 2d

Tag	Data	
1024	44	99
1000	17	23
1008	11	5
1016	29	38

Memory

1000	17
1004	23
1008	11
1012	5
1016	29
1020	38
1024	44
1028	99
1032	97
1036	25
1040	1
1044	4

Miss Penalty and Rate

- The *MISS PENALTY* is the time it takes to read the memory if it isn't in the cache
  - 50 to 100 cycles is common.
- The MISS RATE is the fraction of accesses which MISS
- The HIT RATE is the fraction of accesses which HIT
- MISS RATE + HIT RATE = 1

Suppose a particular cache has a MISS PENALTY of 100 cycles and a HIT RATE of 95%. The CPI for load is normally 5 but on a miss it is 105. What is the average CPI for load?

Average CPI = 10

Suppose MISS PENALTY = 120 cycles?

then CPI = 11 (slower memory doesn't hurt much)

Some Associativity can help

- Direct-Mapped caches are very common but can cause problems...
- SET ASSOCIATIVITY can help.
- Multiple Direct-mapped caches, then compare multiple TAGS
  - 2-way set associative = 2 direct mapped + 2 TAG comparisons
  - 4-way set associative = 4 direct mapped + 4 TAG comparisons
- Now array size == power of 2 doesn't get us in trouble
- But
  - slower
  - less memory in same area
  - maybe direct mapped wins...

What about store?

- What happens in the cache on a store?
  - WRITE BACK CACHE ✍ put it in the cache, write on replacement
  - WRITE THROUGH CACHE ✍ put in cache and in memory
- What happens on store and a MISS?
  - WRITE BACK will fetch the line into cache
  - WRITE THROUGH might just put it in memory

5