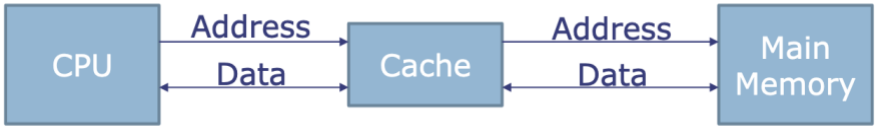
6.004 Recitation Problems
L15 – Memory Hierarchy

Keep the most often-used data in a small, fast SRAM (often local to CPU chip). The reason this strategy works: LOCALITY.

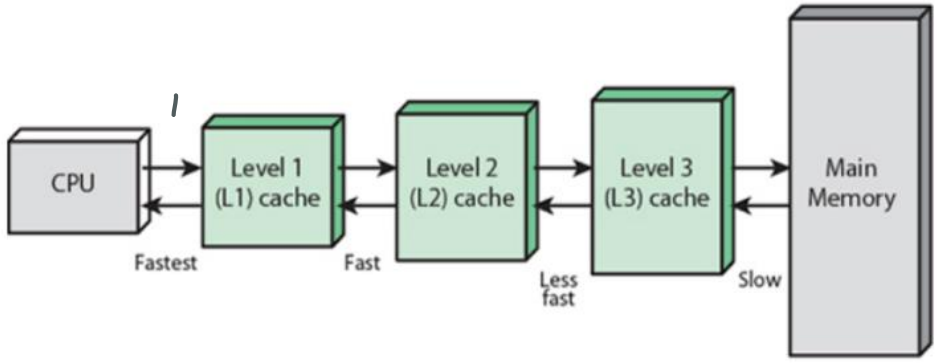
- *Temporal locality: If a location has been accessed recently, it is likely to be accessed (reused) soon*
- *Spatial locality: If a location has been accessed recently, it is likely that nearby locations will be accessed soon*

AMAT(Average Memory Access Time) = HitTime + MissRatio * MissPenalty

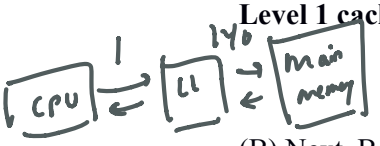


Problem 1. ★

Belly Eyelash is designing a processor and is analyzing the performance of different numbers of cache levels. Without any caches, Belly's main memory has an access time of 140 cycles.



(A) As a test, Belly adds a 32 KB Level 1 (L1) cache that has single-cycle reads/writes, and runs a long computation, during which she observes a new AMAT of 10. **What is the hit ratio for the Level 1 cache during this test?**

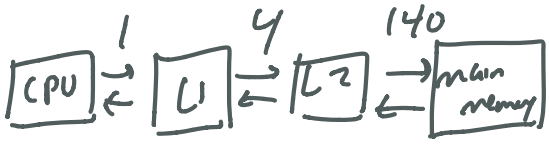


$AMAT = 10 = 1 + (\text{missRatio}) \times 140$

$\text{HitRatio1} = 131/140$

$\text{MissRatio1} = 9/140$

(B) Next, Belly adds a 256 KB Level 2 (L2) cache between the L1 cache and main memory. The L2 cache takes 4 cycles to decide if the a memory access is a hit or a miss. After running the same computation with this new memory hierarchy, Belly observes an improved AMAT of 1.45. Assume that the hit rate for the L1 cache is the same as in (A). **What is the hit ratio for the Level 2 cache during this test?**



$AMAT = 1.45 = 1 + (\text{missRatio1})(4 + (\text{missRatio2}) 140)$

$0.45 \times (\frac{140}{9}) - 4 = 7 - 4 = (\text{missRatio2}) 140$

$\text{missratio2} = \frac{3}{140}$

$\text{hitratio2} = 137/140$

(C) Finally, Belly adds a 10 MB Level 3 (L3) cache between the L2 cache and main memory. After running the same test as before, she observes that the main memory was accessed once for every 140 accesses to the L3 cache. If Belly wants to achieve an AMAT of 1.3 for this computation, **what is the maximum number of clock cycles that the Level 3 cache can take to decide if a memory access is a hit or a miss?** Assume that the L1 and L2 caches can not be changed and they have the same hit rates observed in (A) and (B).

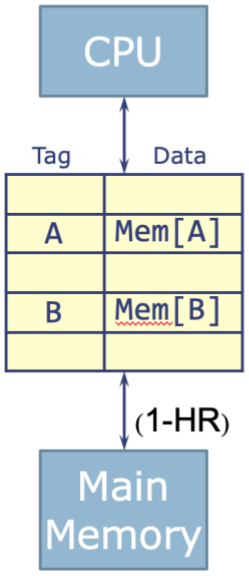
$AMAT = 1.3 = 1 + (9/140)(4 + (3/140)(\text{Hit time 3} + (1/140)140))$

$0.3 \times 140^2 = 9 \cdot 140 \cdot (4 + 3(\text{hit time 3} + 1))$

$\Rightarrow \text{hit time 3} = (0.3 \times 140^2 - 36 \times 140)/27 - 1$

$\text{Hit Time} <= 30$

Basic Cache Algorithm (Reads)



On reference to Mem[X], look for X among cache tags

HIT: X = Tag(i) for some cache line i

Return Data(i)

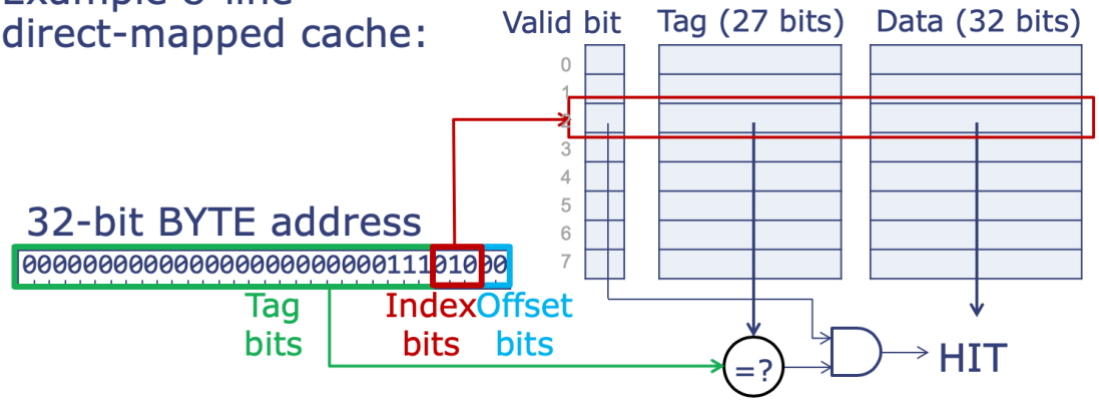
MISS: X not found in Tag of any cache line

Read Mem[X]
Return Mem[X]
Select a line k to hold Mem[X]
Write Tag(k)=X, Data(k) = Mem[X]

Q: How do we "search" the cache?

Direct-Mapped Caches

- Each word in memory maps into a single cache line
- Access (for cache with 2^W lines):
  - Index into cache with W address bits (the index bits)
  - Read out valid bit, tag, and data
  - If valid bit == 1 and tag matches upper address bits, HIT
- Example 8-line direct-mapped cache:



every cache line holds 1 block of memory

if supports byte addressing, then extra muxes

(e.g. microcontrollers where byte density is more important)

Example: Direct-Mapped Caches

64-line direct-mapped cache → 64 indexes → 6 index bits

Read Mem[0x400C]

0100 0000 0000 1100

TAG: 0x40
INDEX: 0x3
OFFSET: 0x0

HIT, DATA 0x42424242

Would 0x4008 hit?

INDEX: 0x2 → tag mismatch
→ MISS

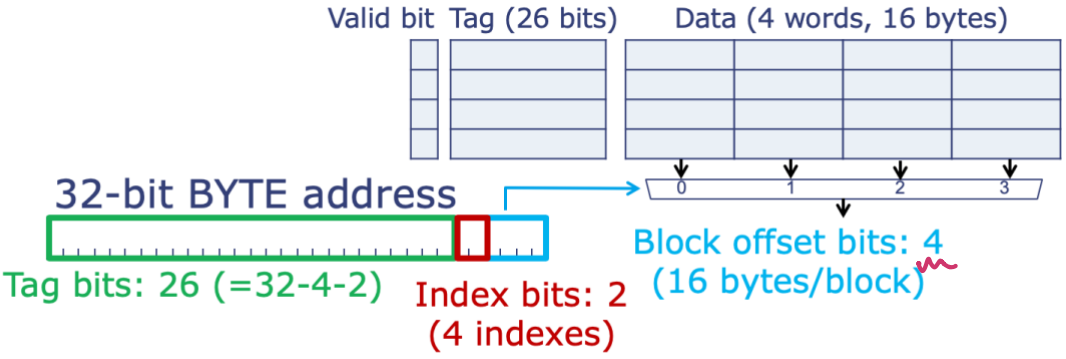
	Valid bit	Tag (24 bits)	Data (32 bits)
0	1	0x000058	0xDEADBEEF
1	1	0x000058	0x00000000
2	1	0x000058	0x00000007
3	1	0x000040	0x42424242
4	0	0x000007	0x6FBA2381
⋮	⋮	⋮	⋮
63	1	0x000058	0xF7324A32

Part of the address (index bits) is encoded in the location
Tag + Index bits unambiguously identify the data's address

Block Size

blocks : 2^{tag}

- Take advantage of spatial locality: Store multiple words per data block
  - Another advantage: Reduces size of tag memory!
  - Potential disadvantage: Fewer blocks in the cache
- Example: 4-block, 16-word direct-mapped cache



1024

1024 words/cache / 4 words/block = [cache lines] 256 = 2^8 blocks/cache (index)

Problem 2.

The RISC-V Engineering Team is working on the design of a cache. They've decided that the cache will have **a total of 2^{10} = 1024 data words**, but are still thinking about the other aspects of the cache architecture.

First assume the team chooses to build a direct-mapped cache with a block size of 4 words.

= 16 bytes

(A) Please answer the following questions:

32 bits total

Number of lines in the cache: 256

Tag: 20 bit
Index: 8 bit
Offset: 4 bit

Number of bits in the tag field for each cache entry: 20

(B) This cache takes *2 clock cycles* to determine if a memory access is a hit or a miss and, if it's a hit, return data to the processor. If the access is a miss, the cache takes *20 additional clock cycles* to fill the cache line and return the requested word to the processor. If the hit rate is 90%, what is the processor's average memory access time in clock cycles?

Average memory access time assuming 90% hit rate (clock cycles): 4

AMAT = 2 + missRate × 20

⇒ miss rate = 10%

2 + 0.1(20) = 2 + 2 = 4

8 bit = byte

4 byte = word

? words in block (depends on cache)

bytes in block
4 words in block ⇒ 16 bytes block

Problem 3. ★

(A) The timing for a particular cache is as follows: checking the cache takes 1 cycle. If there's a hit the data is returned to the CPU at the end of the first cycle. If there's a miss, it takes 10 *additional* cycles to retrieve the word from main memory, store it in the cache, and return it to the CPU. If we want an average memory access time of 1.4 cycles, what is the minimum possible value for the cache's hit ratio?

$1.4 = 1 + \text{missratio} \times 10 = 1.4$ **Minimum possible value of hit ratio:** 96%

$0.4/10 = 0.04 = \text{miss ratio}$

(B) If the cache block size, i.e., words/cache line, is doubled but the total number of data words in the cache is unchanged, how will the following cache parameters change? Please circle the best answer.

index bit −1

# of offset bits: UNCHANGED … **+1** … -1 … 2x … 0.5x … CAN'T TELL

# of tag bits: UNCHANGED … +1 … -1 … 2x … 0.5x … CAN'T TELL

# of cache lines: UNCHANGED … +1 … -1 … 2x … 0.5x … CAN'T TELL

2 index bits = # cache lines

Consider a direct-mapped cache with 64 total data words with 1 word/cache line. This cache architecture is used for parts (C) through (F).

$64 = 2^6$ indx bits = 6

(C) If cache line number 5 is valid and its tag field has the value 0x1234, what is the address in main memory of the data word currently residing in cache line 5?

offset = 2

Main memory address of data word in cache line 5: 0x 12344

0x 1234 tag

0b 0001 01 00 offset (indx bits)

The program shown on the right repeatedly executes an inner loop that sums the 16 elements of an array that is stored starting in location 0x310.

The program is executed for many iterations, then a measurement of the cache statistics is made during one iteration through all the code, i.e., starting with the execution of the instruction labeled `outer_loop:` until just before the next time that instruction is executed.

```
  . = 0                 // tell assembler to start at
                        // address 0
outer_loop:
  addi x4, x0, 16       // initialize loop index J
  mv x1, x0             // x1 holds sum, initially 0

loop:                   // add up elements in array
  subi x4, x4, 1        // decrement index
  slli x2, x4, 2        // convert to byte offset
  lw x3, 0x310(x2)      // load value from A[J]
  add x1, x1, x3        // add to sum
  bne x4, x0, loop      // loop until all words are summed

  j outer_loop          // perform test again!
```

bottom 2 hex digits are indx bit of offset

(D) In total, how many instruction fetches occur during one complete iteration of the outer loop? How many data reads? 8

instruction fetch

= 2 + 5X16 + 1

Number of instruction fetches: 83

Number of data reads: 16

cold miss → nothing in cache live, load directly from memory

(E) How many instruction fetch misses occur during one complete iteration of the outer loop? How many data read misses? Hint: remember that the array starts at address 0x310.

2.

assumption: they were not contig

cold misses (nothing at all)

Number of instruction fetch misses: 4

Number of data read misses: 4

Data: 0x310, 0x314, 0x318 0x31C 0x320

(F) What is the hit ratio measured after one complete iteration of the outer loop?

99 = 83 + 16

Hit ratio: 91/99

```
   outer_loop:
0x0.  addi x4, x0, 16   // initialize loop index J
0x4.  mv x1, x0         // x1 holds sum, initially 0

   loop:                // add up elements in array
0x8   .subi x4, x4, 1   // decrement index
0xC   .slli x2, x4, 2   // convert to byte offset
0x10  .lw x3, 0x310(x2) // load value from A[J]
0x14  .add x1, x1, x3   // add to sum
0x18  .bne x4, x0, loop // loop until all words are summe

0x1C. j outer_loop      // perform test again!
```

0x310 J

0x10

have same index in cache ⇒

CLASH