Lecture 24: Cache money & virtual memory

CMPE 120: Computer Organization & Architecture

Olivia Weng

Logistics

- Reminder: check-in with professor is 1% of your final grade
  - Sign up link

- Grade remediation plan
  - 30+ students have already signed up
  - Slots are getting booked quickly → sign up soon!

- HW5 released last Thursday. Start now!

Grade remediation plan

- HW1 Quiz Remediation: meet with professor for 15 minutes
  - Bring a newly hand-written solution. Explain what you got wrong on your quiz and your new solution.
  - Replaces HW1 quiz grade & drops lowest HW Quiz grade
  - Schedule a time on my calendar in the next ~3 weeks (10/30 - 11/20)
    - Link: https://calendar.app.google/d3D4hm3JbmpXcrVt5
    - 100 appointment slots available for 67 students
    - If none of these times work, email me to schedule a time during this window.

- Final exam grade can replace the midterm grade if higher
  - → Midterm grade replacement = max(final exam grade, midterm exam grade)

Fully associative cache

- Fully associative cache: block can be placed in any cache location
  - No rules on what data can go where

Fully associative cache

- Fully associative cache: block can be placed in any cache location
  - No rules on what data can go where

- But, some new considerations:
  - Where do you look to find an element?
  - How to decide what to remove? – replacement policy

least recently used

LRU

① compare ALL tags A & B 2-element integer arrays.

Fully associative cache – deal with conflict misses!

Item	Addr b10	Addr b2	H/M?
A[0]	16	00010 000	M
B[0]	48	00110 000	M
A[1]	20	00010 100	
B[1]	52	00110 100	
A[0]	16	00010 000	
B[0]	48	00110 000	
A[1]	20	00010 100	
B[1]	52	00110 100	

tag | block offset bits

Each block holds 2 words (8 bytes)

tag	0	1	2	3	4	5	6	7
	data							
00010	A[0]		A[1]					
00110	B[0]		B[1]					

What is the hit rate on this cache?

A. 75%
B. 50%
C. 25%
D. 12.5%
E. 0%

hit rate = # hits / # accesses

Set associativity

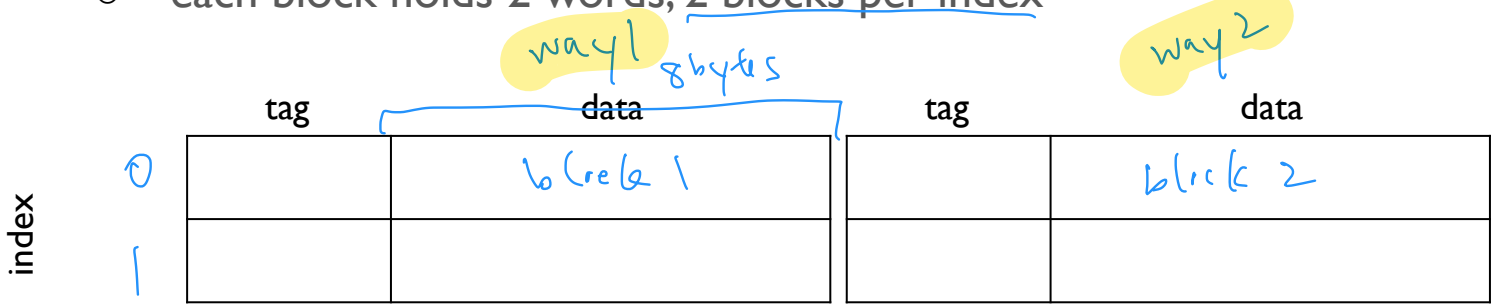
Direct mapped
index into cache — fast!
but could have many conflict misses

Fully-associative
- search through all tags
- keep track LRU

- n-way set-associative cache
  - Each index can store n blocks
  - All blocks sharing an index are in a set

2-way set associative cache

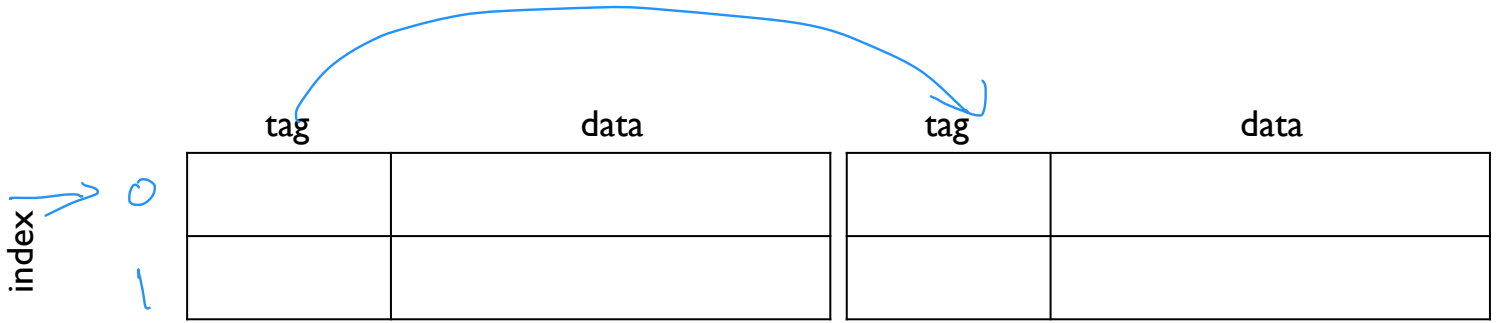
- Example: 4 blocks
  - each block holds 2 words, 2 blocks per index



Set associativity

- Consider again:
  - Where do you look to find an element?
  - How to decide what to remove?

LRU - only for your set



1. check index
2. compare w/ tags in this set

2-way set-associative cache

byte-addressable

Each block holds 2 words (8 bytes)

Item	Addr b10	Addr b2	H/M?
A[0]	16	0001 0 000	M
B[0]	48	0011 0 000	M
A[1]	20	0001 0 100	H
B[1]	52	0011 0 100	H
A[0]	16	0001 0 000	H
B[0]	48	0011 0 000	H
A[1]	20	0001 0 100	H
B[1]	52	0011 0 100	H

tag | idx | ↳ block offset bits $\log_2(8 \text{ bytes}) = 3$

idx

	tag	data	tag	data
0	0001	A[0] A[1]	0011	B[0] B[1]
1				

What is the hit rate on this cache?

A. 75%
B. 50%
C. 25%
D. 12.5%
E. 0%

↳ $\frac{\#\text{hits}}{\#\text{accesses}}$

$\frac{4}{8}$

Set associative caches

- Performance is middle-ground between directed mapped & fully associative
  - Decent hit times
  - Decent use of cache space

- Hybrid design between direct mapped & fully associative

- Hit rates generally higher than direct mapped

Least recently used (LRU)

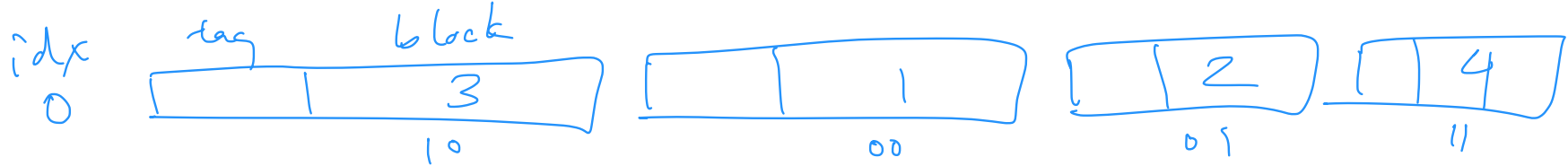
- To do LRU, must know true ordering of accesses
  - Need extra bits to keep track of order

Least recently used (LRU)

Accesses
1 ←
2
3 ←
4

- To do LRU, must know true ordering of accesses
  - Need extra bits to keep track of order
  - Ex: 4-way set-associative cache, each block has metadata on if accessed
    - 00: least recently
    - 01: second least recently
    - 10: second most recently
    - 11: most recently
  - All bits updated on every access (hit or miss)

4-way



Key caching formulas

- With n-way associative caches
  - `num_block_offset_bits = log2(block_size_in_bytes)`
  - `num_cache_entries = cache_size / (block_size * n)`
  - `num_index_bits = log2(num_cache_entries)`
  - `num_tag_bits = address_size - num_index_bits - num_block_offset_bits`

cold cache

Summary: Cache miss types

- Compulsory (or cold-start) miss: first access to data

Summary: Cache miss types

- Compulsory (or cold-start) miss: first access to data

- Capacity miss: missed only because cache is not big enough
  - Address has already been requested
  - AND fully-associative cache of same size would also experience miss

Summary: Cache miss types



- Compulsory (or cold-start) miss: first access to data

- Capacity miss: missed only because cache is not big enough
  - Address has already been requested
  - AND fully-associative cache of same size would also experience miss

- Conflict miss: missed because of indexing
  - Address has already been requested
  - AND fully-associative cache of same size would also experience hit

Virtual memory

What if ISA has address space bigger than physical memory?

- 64-bit address space with byte addressability = 16 Exabytes

2^{64} addresses

$2^4 \times 2^{60} = 2^{64}$

16 EB

$2^{10} = 1$ KB

$2^{20} = 1$ MB

$2^{30} = 1$ GB

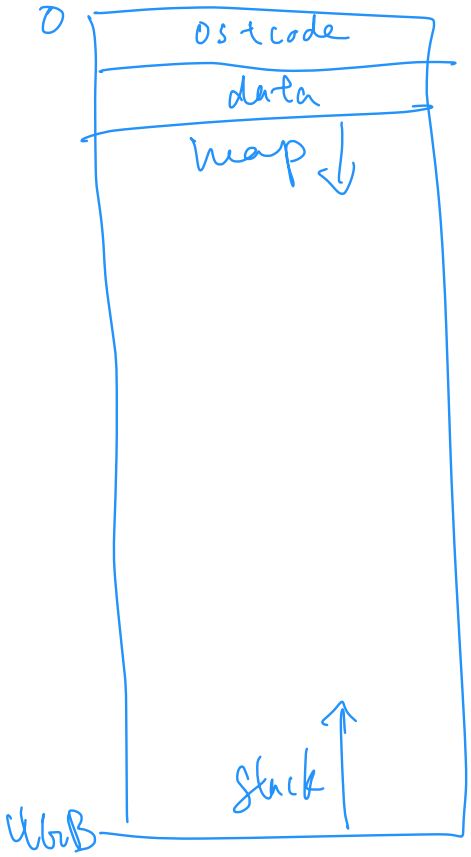
$2^{40} = 1$ TB

$2^{50} = 1$ PB

$2^{60} = 1$ EB

Remember processes?

process = running program



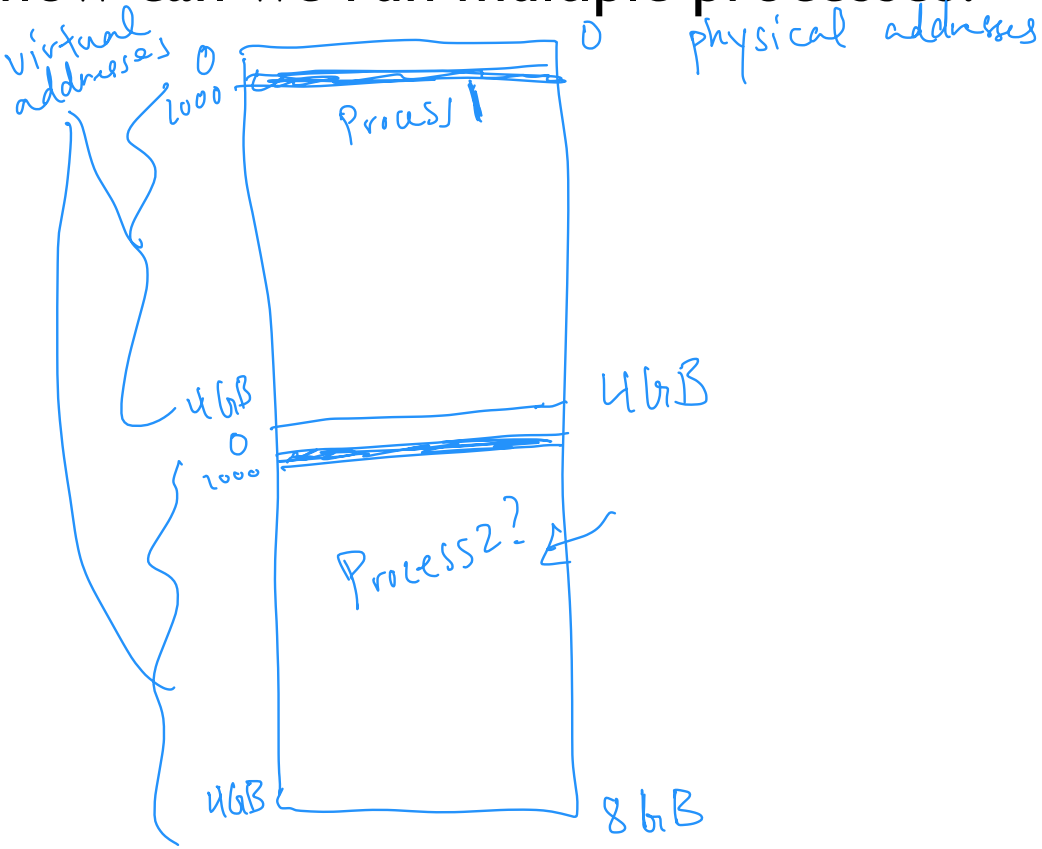
32-bit computer

2^{32} addresses

$2^{2} \times 2^{30}$

4 GB

With one RAM, how can we run multiple processes?



Memory virtualization

- Memory virtualization: divide up physical RAM into smaller "virtual" memory spaces
  - So each process thinks it has its own memory

Memory virtualization

- Memory virtualization: divide up physical RAM into smaller "virtual" memory spaces
  - So each process thinks it has its own memory
  - The physical memory can be smaller than the ISA's address space
  - Hardware and software cooperatively manage how virtual memory maps to physical memory

Memory virtualization

- Address generated by each instruction in a program is virtual address
  - Not the physical address used for main memory (DRAM)

Memory virtualization

- Address generated by each instruction in a program is virtual address
  - Not the physical address used for main memory (DRAM)

- Address translation maps virtual address to a physical address

