CS 354 - Machine Organization & Programming
Tuesday Mar 21st and Thursday Mar 23rd,2023

Midterm Exam 2 - Thursday April 6th, 7:30 - 9:30 pm

- **UW ID required**
- **#2 pencils required**
- **closed book, no notes, no electronic devices (e.g., calculators, phones, watches) see "Midterm Exam 2" on course site Assignments for topics**

Homework hw4: DUE on or before Monday, Mar 27 7th

Project p3: DUE on or before Friday, Mar 24th

Project p4A: DUE on or before Friday, Mar 31st

Project p4B: DUE on or before Friday, April 7th

Last Week (before Spring Break)

Locality & Caching Bad Locality Caching: Basic Idea & Terms Designing a Cache: Blocks	Rethinking Addressing Designing a Cache: Sets and Tags TODO Basic Cache Lines TODO Basic Cache Operation Basic Cache Practice-practice quiz L14-12

This Week

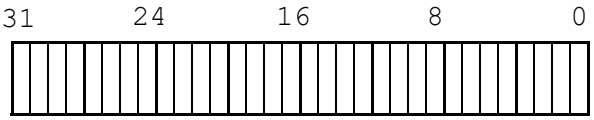
Finish L14 (bring W7 outline) Direct Mapped Caches - Restrictive Fully Associative Caches - Unrestrictive Set Associative Caches - Sweet! Replacement Policies	Writing to Caches Cache Performance Impact of Stride Memory Mountain C, Assembly, and Mach Code
Next Week: Assembly Language Instructions and Operands B&O Chapter 3 Intro 3.1 A Historical Perspective 3.2 Program Encodings 3.3 Data Formats 3.4 Accessing Information 3.5 Arithmetic and Logical Control 3.6 Control	

Direct Mapped Caches - Restrictive

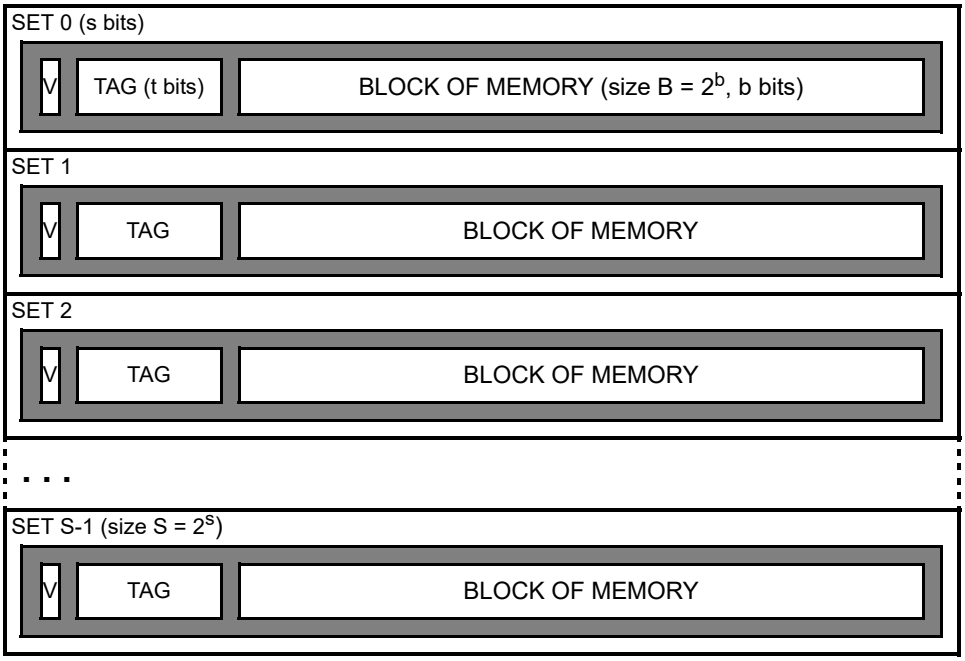
Direct Mapped Cache is a cache

→ What is the address breakdown if blocks are 32 bytes and there are 1024 sets?

32-bit Address Breakdown



→ Is the cache operation fast O(1) or slow O(S) where S is the number of sets?



→ What happens when two different memory blocks map to the same set?

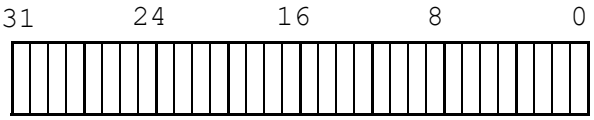
❋ *Appropriate for*

Fully Associative Caches - Unrestrictive

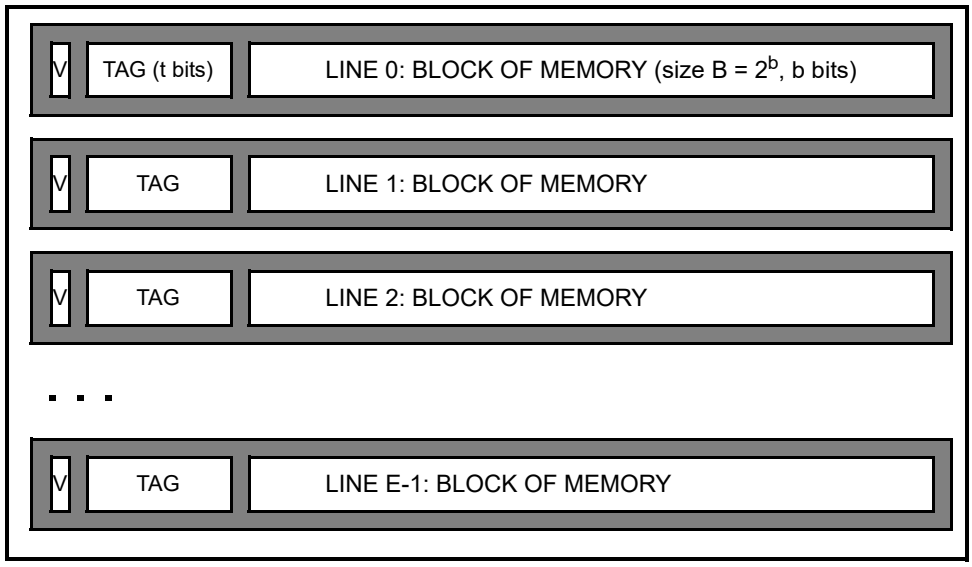
Fully Associative Cache is a cache

→ What is the address breakdown if blocks are 32 bytes and there are 1024 sets?

32-bit Address Breakdown



→ Is the cache operation fast O(1) or slow O(E) where E is the number of lines?



→ What happens when two different memory blocks map to the same set?

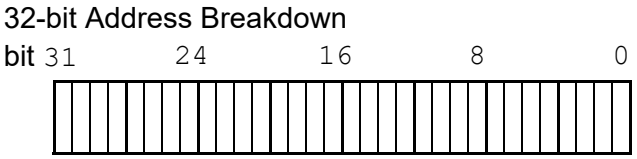
→ How many lines should a fully associative cache have?

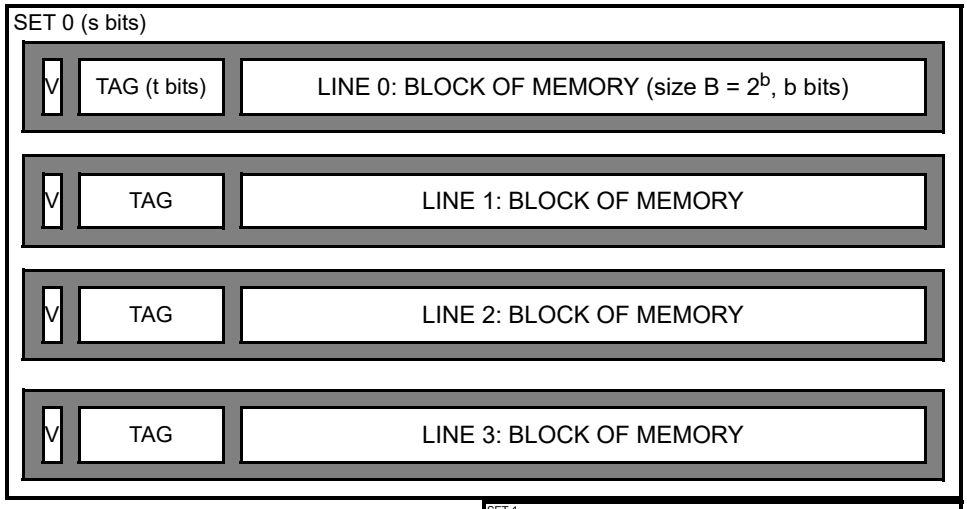
❋ *Appropriate for*

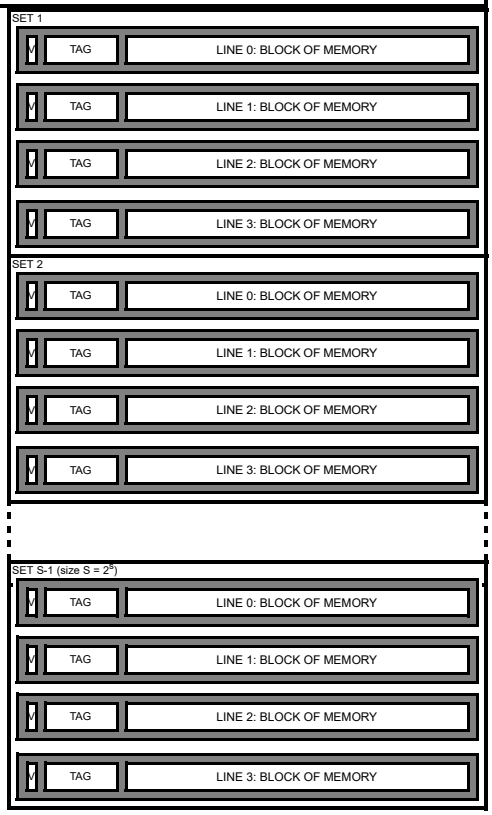
Set Associative Caches - Sweet!

Set Associative Cache is a cache commonly used today

→ What is the address breakdown if blocks are 32 bytes and there are 1024 sets?







Let E be

E = 4 is

E = 1 is

❋ ***C = (S, E, B, m)***

Let C be

→ How big is a cache given (1024, 4, 32, 32)?

→ What happens when E+1 different memory blocks map to the same set?

Replacement Policies

Assume the following sequence of memory blocks

are fetched into the same set of a 4-way associative cache that is initially empty:
`b1`, `b2`, `b3`, `b1`, `b3`, `b4`, `b4`, `b7`, `b1`, `b8`, `b4`, `b9`, `b1`, `b9`, `b9`, `b2`, `b8`, `b1`

1. *Random Replacement*

→ Which of the following four outcomes is possible after the sequence finishes?
Assume the initial placement is random.

	L0	L1	L2	L3
1.	`b9`	`b1`	`b8`	`b2`
2.	`b1`	`b2`	`--`	`b8`
3.	`b1`	`b4`	`b7`	`b3`
4.	`b1`	`b2`	`b8`	`b1`

2. *Least Recently Used* (LRU)

→ What is the outcome after the sequence finishes?
Assume the initial placement is in ascending line order (left to right below).

L0	L1	L2	L3

3. *Least Frequently Used* (LFU)

→ Which blocks will remain in the cache after the sequence finishes?

❋ *Exploiting replacement policies*

Writing to a Cache

- *Reading data copies*
- *Writing data requires that*

Write Hits

occur when writing to a block

→ When should a block be updated in lower memory levels?

1. ***Write Through***:

2. ***Write Back***:

Write Misses

occur when writing to a block

→ Should space be allocated in this cache for the block being changed?

1. ***No Write Allocate***:

2. ***Write Allocate***:

Typical Designs

1. **Write Through** paired with
2. **Write Back paired** with

→ Which best exploits locality?

Cache Performance

Metrics

hit rate

hit time

miss penalty,

Larger Blocks (S and E unchanged)

hit rate

hit time

miss penalty

THEREFORE

More Sets (B and E unchanged)

hit rate

hit time

miss penalty

THEREFORE

More Lines E per Set (B and S unchanged)

hit rate

hit time

miss penalty

THEREFORE

Intel Quad Core i7 Cache (gen 7)

all: 64 byte blocks, use pseudo LRU, write back

L1: 32KB, 4-way Instruction & 32KB 8-way Data, no write allocate
L2: 256KB, 8-way, write allocate
L3: 8MB, 16-way (2MB/Core shared), write allocate

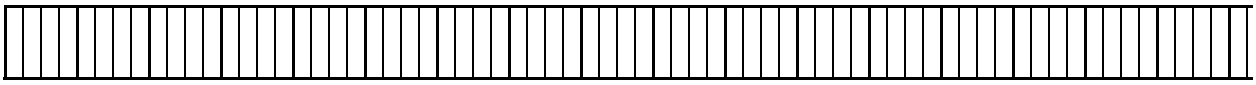
Impact of Stride

Stride Misses

Example:

```
int initArray(int a[][8], int rows) {
   for (int i = 0; i < rows; i++)
      for(int j = 0; j < 8; j++)
         a[i][j] = i * j;
}
```

→ Draw a diagram of the memory layout of the first two rows of `a`:



Assume: `a` is aligned with cache blocks
is too big to fit entirely into the cache
words are 4 bytes, block size is 16 bytes
direct-mapped cache is initially empty, write allocate used

→ Indicate the order elements are accessed in the table below and mark H for hit or M for miss:

a[i][j]	j = 0	1	2	3	4	5	6	7
i = 0								
1								
...								

→ Now exchange the `i` and `j` loops mark the table again:

a[i][j]	j = 0	1	2	3	4	5	6	7
i = 0								
1								
...								

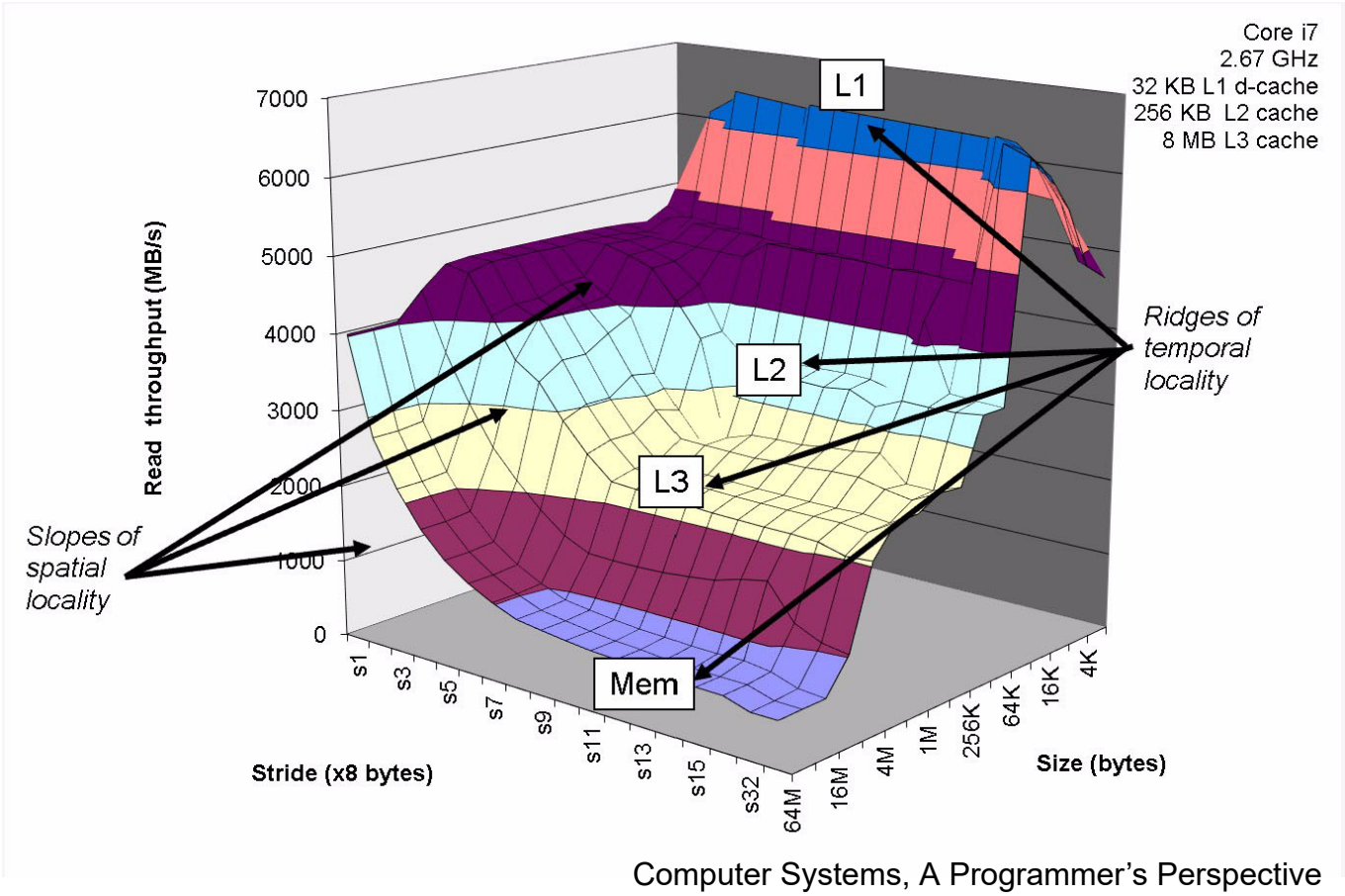
Memory Mountain

Independent Variables

stride - 1 to 16 double words step size used to scan through array
size - 2K to 64 MB arraysize

Dependent Variable

read throughput - 0 to 7000 MB/s



Computer Systems, A Programmer's Perspective
Second Edition, Bryant and O'Hallaron

Temporal Locality Impacts

Spatial Locality Impacts

- *Memory access speed is not characterized*

C, Assembly, & Machine Code

C Function	Assembly (AT&T)	Machine (hex)
`int accum = 0;`		
`int sum(int x, int y)`	`sum:`	
`{`	`  pushl %ebp`	`55`
	`  movl %esp, %ebp`	`89 e5`
	`  movl 12(%ebp), %eax`	`8b 45 0C`
`  int t = x + y;`	`  addl 8(%ebp), %eax`	`03 45 08`
`  accum += t;`	`  addl %eax, accum`	`01 05 ?? ?? ?? ??`
`  return t;`	`  popl %ebp`	`5D`
`}`	`  ret`	`C3`

C

-
-
-

→ What aspects of the machine does C hide from us?

Assembly (ASM)

-
-

→ What ISA (Instruction Set Architecture) are we studying?

→ What does assembly remove from C source?

→ Why Learn Assembly?

1.
2.
3.

Machine Code (MC) **is**

-
-

→ How many bytes long is an IA-32 instructions?