CS 354 - Machine Organization & Programming
Tuesday, October 29th, 2019

Midterm Exam 2 (~18%): Thursday, November 7th, 7:15 - 9:15 pm

Project p4A (~2%): DUE at 10 pm on Tuesday, November 5th
Project p4B (~4%): Assigned later this week
Homework hw4 (1.5%): DUE at 10 pm on Thursday, October 31st

Last Time

- Set Associative Cache
- Replacement Policies
- Fully Associative Cache
- Writing to Caches
- Cache Performance Metrics
- Cache Parameters and Performance

Today

- Impact of Stride
- Memory Mountain
- C, Assembly, & Machine Code
- Low-level View of Data
- Registers
- Instructions - MOV, PUSH, POP

Next Time

- More Instructions and Operands
- **Read:** B&O 3.5, 3.6

Impact of Stride

B: cache block size

Stride Misses $\%\ misses = min(1, (word\text{-}size * k)/B) * 100$.

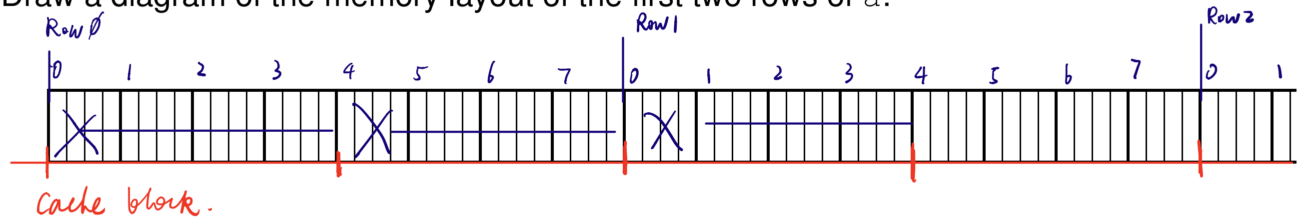
k is stride size in words.

Lower is better.

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 and <u>is too big to fit entirely into the cache</u>
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	1M	2H	3H	4H.	5M	6H	7H	8H.
1	9 M	10 H	11 H	12H.				
...								

$\%\ Misses = min(1, (4 \times 1)/16) * 100 = min(1, .25) * 100 = 25\%$.

→ 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	1M							
1	2M							
...	3M							

$\%\ Miss = min(1, 4 \times 8/16) = 100\%$

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

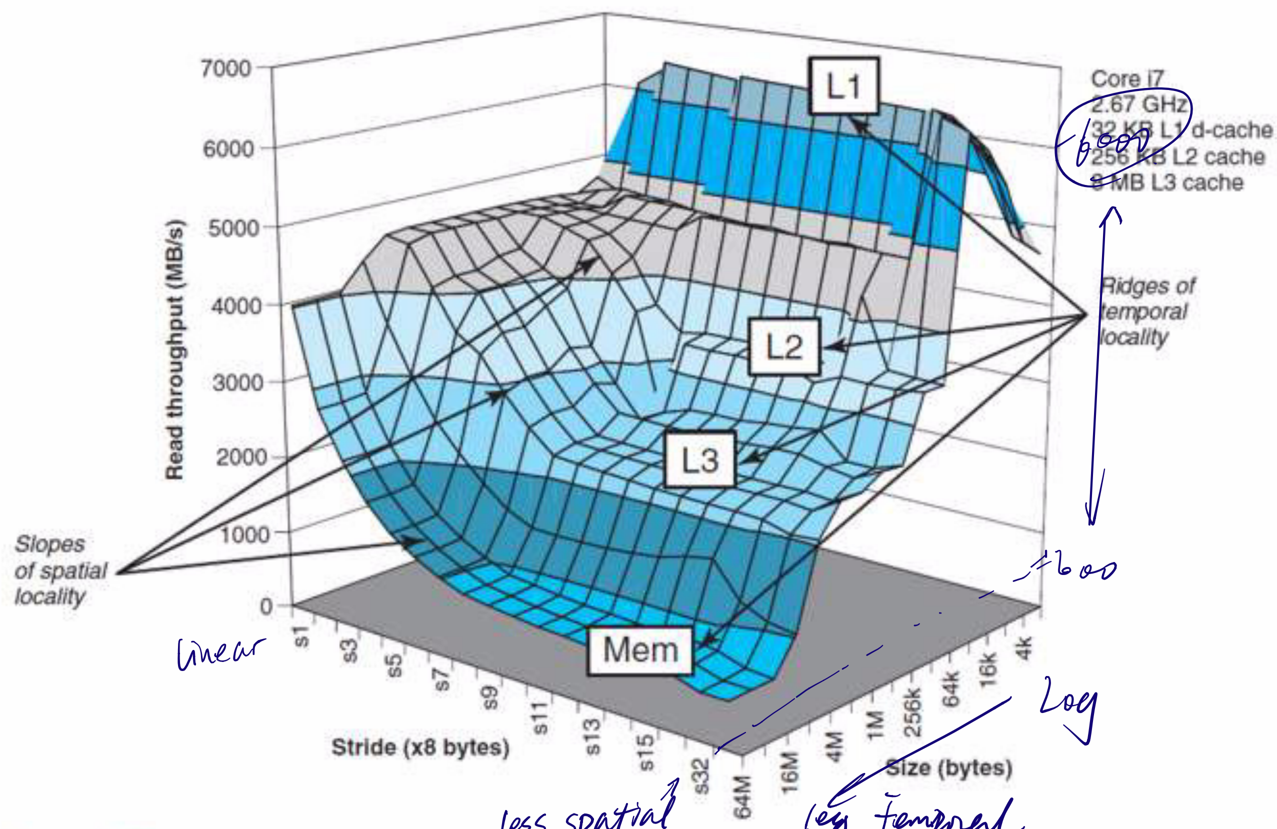


Figure 6.43 The memory mountain.

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

Temporal Locality Impacts Factor ~10 Times.

Spatial Locality Impacts

Factor ~7 times.

✵ *Memory access speed is not characterized* by a single value
It's a landscape that can be exploited through the use of spatial and temporal locality

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

- is high level language that enable us be more productive
- help us write correct code with syntax and type checking
- can be compiled and run on different platforms

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

low-level machine details. - machine instructions.
- Addressing modes.
- registers, condition codes ...

Assembly (ASM)

- is human readable representation of machine code
- is very machine depending.

→ What ISA (Instruction Set Architecture) are we studying? IA-32 x86

→ What does assembly remove from C source?

high level-language constructs, - logical control structs. IF, For,
- variance names, and data types
- composite data such as arrays and stacks.

→ Why Learn Assembly?

~~XXX~~ to better understand stack, C
· identify code inefficiencies and vulnerabilities
· understand compiler optimizations.

Machine Code (MC)

- is elementary CPU instrucs and data in binary.
- is the unique encodings that a particular machine architecture understand and can execute

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

1-15 bytes.

Low-Level View of Data

C's View

- variables are declared to be specific types
- types can be compacted composites built from arrays and structs.

Machine's View

memory is like an array of bytes indexed by addresses where each element is a byte.

⚹ *Memory contains bits that do not* distinguish instructions from different data types or pointer addresses.

→ How does a machine know what it's getting from memory?

1. by how mem is accessed : instruction fetching v.s. operand loading.
2. by the instruction itself : operation (what type) and operands (where and what size).

Assembly Data Formats

	C	IA-32	Assembly Suffix	Size in bytes	
	`char`	byte	b	1	
	`short`	word	w	2	
	`int`	double word	l	4	l : Long
	`long int`	double word	l	4	
ADDR.	$\boxed{\texttt{char*}}$	double word	l	4	
	`float`	single precision	s	4	
	`double`	double prec	l	8	
	`long double`	extended prec	t	10	

⚹ *In IA-32 a word* is actually 2 bytes.

Registers

What? Registers

- are the fastest memory directly accessed by ALU
- can store 1, 2, and 4 bytes of data.
  or 4 bytes for addresses

General Registers

prenamed locations that store up to 32 bits.

8 bit, higher
h = 8-15 bits

8 Bit
l means lower part

	bit 31	16 15		8 7 0
%eax	Accumulator	%ax	%ah	%al
%ecx	count	%cx	%ch	%cl
%edx	data	%dx	%dh	%dl
%ebx	base	%bx	%bh	%bl
%esi	source index.	%si		
%edi	destination index	%di		
%esp	stack pointer	%sp		
%ebp	base pointer	%bp		

Bits 0-7.

32 bit.
bit 0-32

16 bits registers.
bits 0-15.

Program Counter PC `%eip` extended instruction pointer
stores address of next instruction.

Condition Code Registers

1 bit registers that store status.
of most recent ALU operations.

Instructions - MOV, PUSH, POP

What? These are instructions to copy data from source s to destination d.

Why? enables info to be moved around in mem and registers.

How?

instruction class	operation	description
MOV S, D movb movw movl	D ← S	copies S to D
MOVS S, D movsbw movsbl movswl	D ← signextended(S)	copies s sign extended to d.
MOVZ S, D movzbw bl wl	D ← zeroextended(S).	copies s zero extended to d.
pushl S	R[%esp] ← R[%esp] - 4 M[R[%esp]] ← S	push S onto stack.
popl D	D ← M[R[%esp]] R[%esp] ← R[%esp] + 4	pop from stack to D

Practice with Data Formats

→ What data format suffix should replace the _ given the registers used?

```
1. movl   %eax, %esp
2. pushl  $0xFF
3. movw   (%eax), %dx
4. movb   (%esp, %edx, 4), %dh
5. movb   0x800AFFE7, %bl
6. movw   %dx, (%eax)
7. popl   %edi
```

B byte 1
w word 2
l long 4