Memory Hierarchy Inside Out: (1) Memories bring back you...

Hung-Wei Tseng


Disney+
Disney · PIXAR
INSIDE OUT
GET DISNEY+
TRAILER
PG 2015 • 1h 35m • Coming of age, Family, Animation
When 11-year-old Riley moves to a new city, her Emotions team up to help her through the transition. Joy, Fear, Anger, Disgust and Sadness work together, but when Joy and Sadness get lost, they must journey through unfamiliar places to get back home.

von Neumman Architecture

509cbd23

00c2e800

Processor

Instructions	Data
0f00bb27	00c2e800
509cbd23	00000008
00005d24	00c2f000
0000bd24	00000008
2ca422a0	00c2f800
130020e4	00000008
00003d24	00c30000
2ca4e2b3	00000008

Memory

Program

Instructions	Data
0f00bb27	00c2e800
509cbd23	00000008
00005d24	00c2f000
0000bd24	00000008
2ca422a0	00c2f800
130020e4	00000008
00003d24	00c30000
2ca4e2b3	00000008

Storage

Recap: Summary of CPU Performance Equation

$$Performance = \frac{1}{Execution\ Time}$$

$$Execution\ Time = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

$$ET = IC \times CPI \times CT$$

$$Speedup = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- IC (Instruction Count)
  - ISA, Compiler, algorithm, programming language, **programmer**
- CPI (Cycles Per Instruction)
  - Machine Implementation, microarchitecture, compiler, application, algorithm, programming language, **programmer**
- Cycle Time (Seconds Per Cycle)
  - Process Technology, microarchitecture, **programmer**

Lessons learned from Amdahl's Law

$$Speedup_{enhanced}(f, s) = \frac{1}{(1 - f) + \frac{f}{s}}$$

- Corollary #1: Maximum speedup
- Corollary #2: Make the common case fast
  - Common case changes all the time
- Corollary #3: Optimization is a moving target
- Corollary #4: Exploiting more parallelism from a program is the key to performance gain in modern architectures
- Corollary #5: Single-core performance still matters

$$Speedup_{max}(f, \infty) = \frac{1}{(1 - f)}$$

$$Speedup_{max}(f_1, \infty) = \frac{1}{(1 - f_1)}$$

$$Speedup_{max}(f_2, \infty) = \frac{1}{(1 - f_2)}$$

$$Speedup_{max}(f_3, \infty) = \frac{1}{(1 - f_3)}$$

$$Speedup_{max}(f_4, \infty) = \frac{1}{(1 - f_4)}$$

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})}$$

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})}$$

Is TFLOPS (Tera FLoating-point Operations Per Second) a good metric?

$$TFLOPS = \frac{\#\ of\ floating\ point\ instructions \times 10^{-12}}{Exection\ Time}$$

$$= \frac{IC \times \%\ of\ floating\ point\ instructions \times 10^{-12}}{IC \times CPI \times CT}$$

$$= \frac{\%\ of\ floating\ point\ instructions \times 10^{-12}}{CPI \times CT}$$

IC is gone!

A good performance metric must cover IC, CPI, CT!

- Cannot compare different ISA/compiler
  - What if the compiler can generate code with fewer instructions?
  - What if new architecture has more IC but also lower CPI?
- Does not make sense if the application is not floating point intensive

Instances per batch

Inference per second

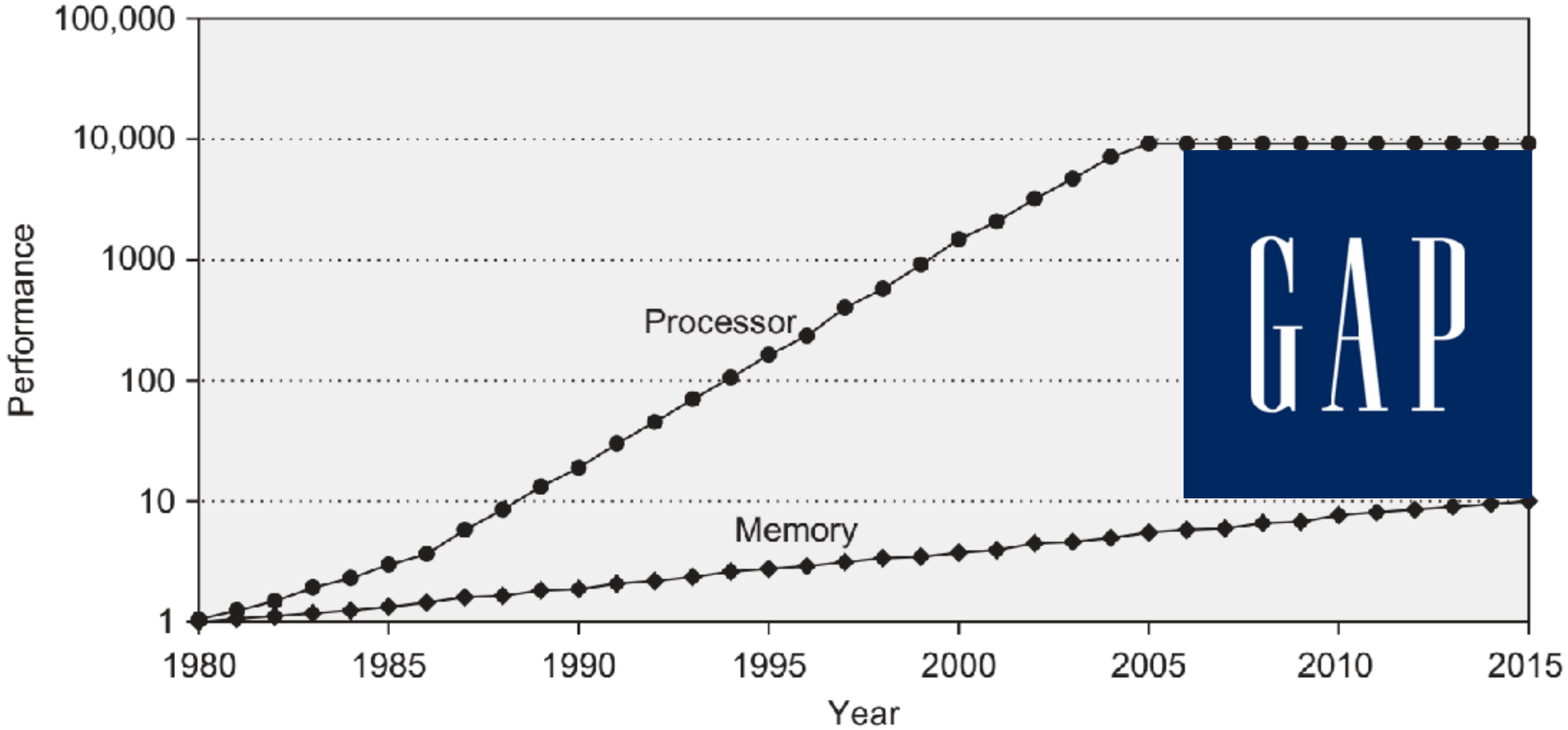
$$\frac{Inferences}{Second} = \boxed{\frac{Inferences}{Operation}} \times \frac{Operations}{Second}$$

$$= \frac{Inferences}{Operation} \times [\frac{operations}{cycle} \times \frac{cycles}{second} \times \#\_of\_PEs \times Utilization\_of\_PEs]$$

IC is gone again!

	Hardware	Model	Input Data
Operations per inference		v	
Operations per cycle	v		
Cycles per second	v		
Number of PEs	v		
Utilization of PEs	v	v	
Effectual operations out of (total) operations		v	v
Effectual operations plus unexploited ineffectual operations per cycle	v		

Recap: Performance gap between Processor/Memory



3:00

How do you usually prepare a closed book midterm?

How do you prepare closed-book exams?

- Review questions from prior years
- Review the whole chapter
- Practice similar questions
- Practice many times

Outline

- The Basic Idea behind Memory Hierarchy
- How cache works

Performance of modern DRAM

			Best case access time (no precharge)			Precharge needed
Production year	Chip size	DRAM type	RAS time (ns)	CAS time (ns)	Total (ns)	Total (ns)
2000	256M bit	DDR1	21	21	42	63
2002	512M bit	DDR1	15	15	30	45
2004	1G bit	DDR2	15	15	30	45
2006	2G bit	DDR2	10	10	20	30
2010	4G bit	DDR3	13	13	26	39
2016	8G bit	DDR4	13	13	26	39

Figure 2.4 Capacity and access times for DDR SDRAMs by year of production. Access time is for a random memory word and assumes a new row must be opened. If the row is in a different bank, we assume the bank is precharged; if the row is not open, then a precharge is required, and the access time is longer. As the number of banks has increased, the ability to hide the precharge time has also increased. DDR4 SDRAMs were initially expected in 2014, but did not begin production until early 2016.

The impact of "slow" memory

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has "perfect" memory, the CPI is just 1. Now, consider we have DDR4 and the program is well-behaved that precharge is never necessary — the access latency is simply 26 ns. What's the average CPI (pick the most close one)?
  A. 9
  B. 17
  C. 27
  D. 35
  E. 69

The impact of "slow" memory

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has "perfect" memory, the CPI is just 1. Now, consider we have DDR4 and the program is well-behaved that precharge is never necessary — the access latency is simply 26 ns. What's the average CPI (pick the most close one)?

  A. 9
  B. 17
  C. 27
  D. 35
  E. 69

$$1 + \boxed{100\% \times (52)} + 30\% \times 52 = 68.6\ cycles$$

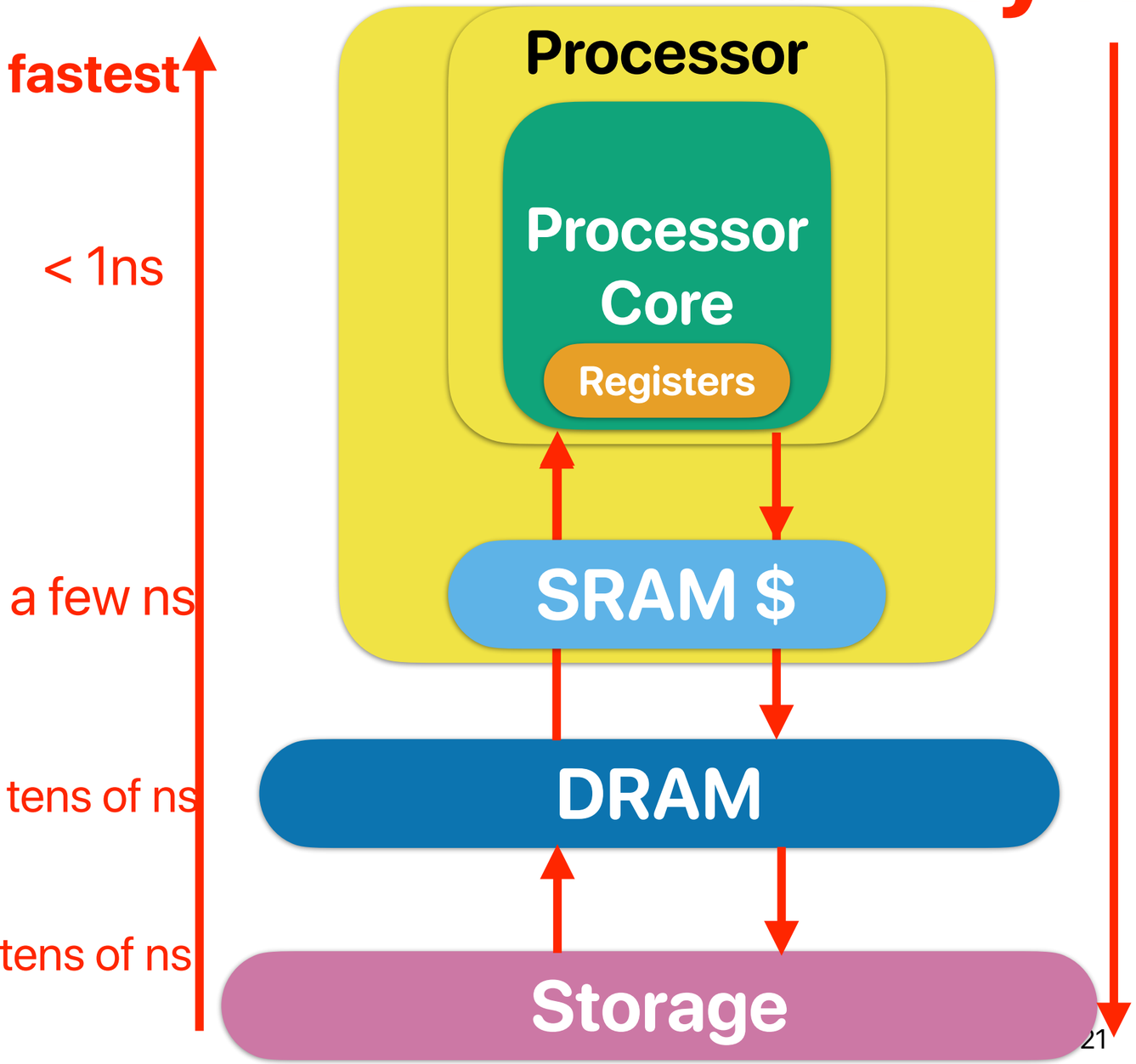
Don't forget, instructions are also from "memory"

Alternatives?

Memory technology	Typical access time	$ per GiB in 2012
SRAM semiconductor memory	0.5–2.5 ns	$500–$1000
DRAM semiconductor memory	50–70 ns	$10–$20
Flash semiconductor memory	5,000–50,000 ns	$0.75–$1.00
Magnetic disk	5,000,000–20,000,000 ns	$0.05–$0.10

Fast, but expensive $$$

Memory Hierarchy

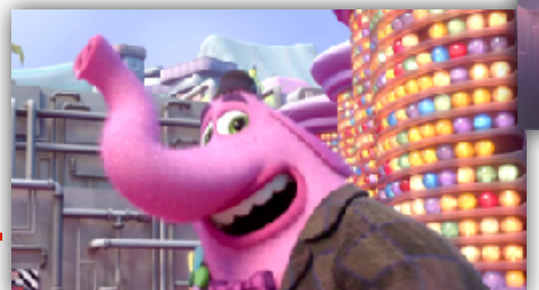


32 or 64 words

KBs ~ MBs

GBs

TBs



How can memory hierarchy help in performance?

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has "perfect" memory, the CPI is just 1. Now, in addition to DDR4, whose latency is 26 ns, we also got an SRAM cache with latency of just at 0.5 ns and can capture 90% of the desired data/instructions. what's the average CPI (pick the most close one)?
  A. 2
  B. 4
  C. 8
  D. 16
  E. 32

How can memory hierarchy help in performance?

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has "perfect" memory, the CPI is just 1. Now, in addition to DDR4, whose latency 26 ns, we also got an SRAM cache with latency of just at 0.5ns and can capture 90% of the desired data/instructions. what's the average CPI (pick the most close one)?

  A. 2

  B. 4

  C. 8

  D. 16

  E. 32

$$1 + (1 - 90\%) \times [100\% \times (52) + 30\% \times 52] = 7.76\ cycles$$

Memory Hierarchy

fastest

< 1ns

a few ns

tens of ns

us/ms

Processor

Processor Core

Registers

SRAM $

DRAM

Storage

32 or 64 words

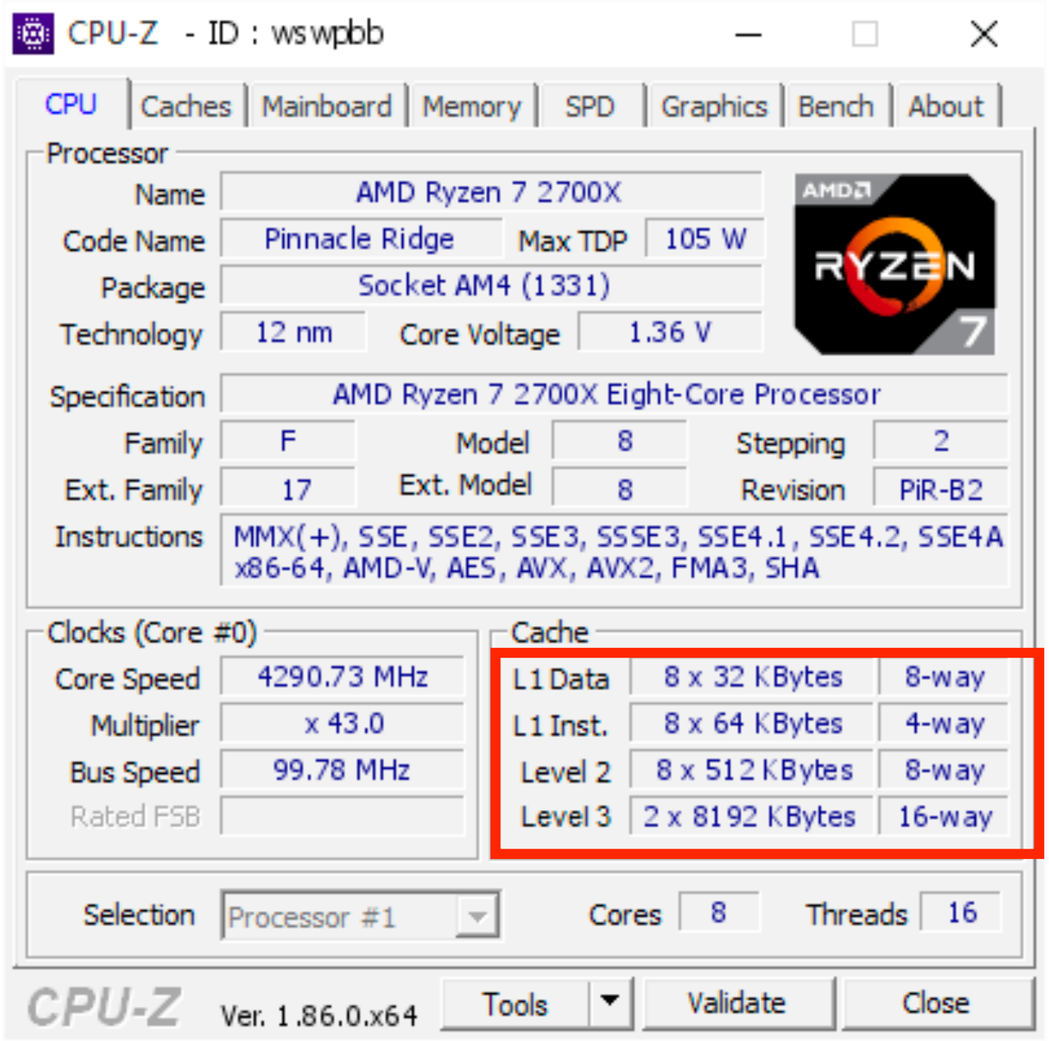
KBs ~ MBs

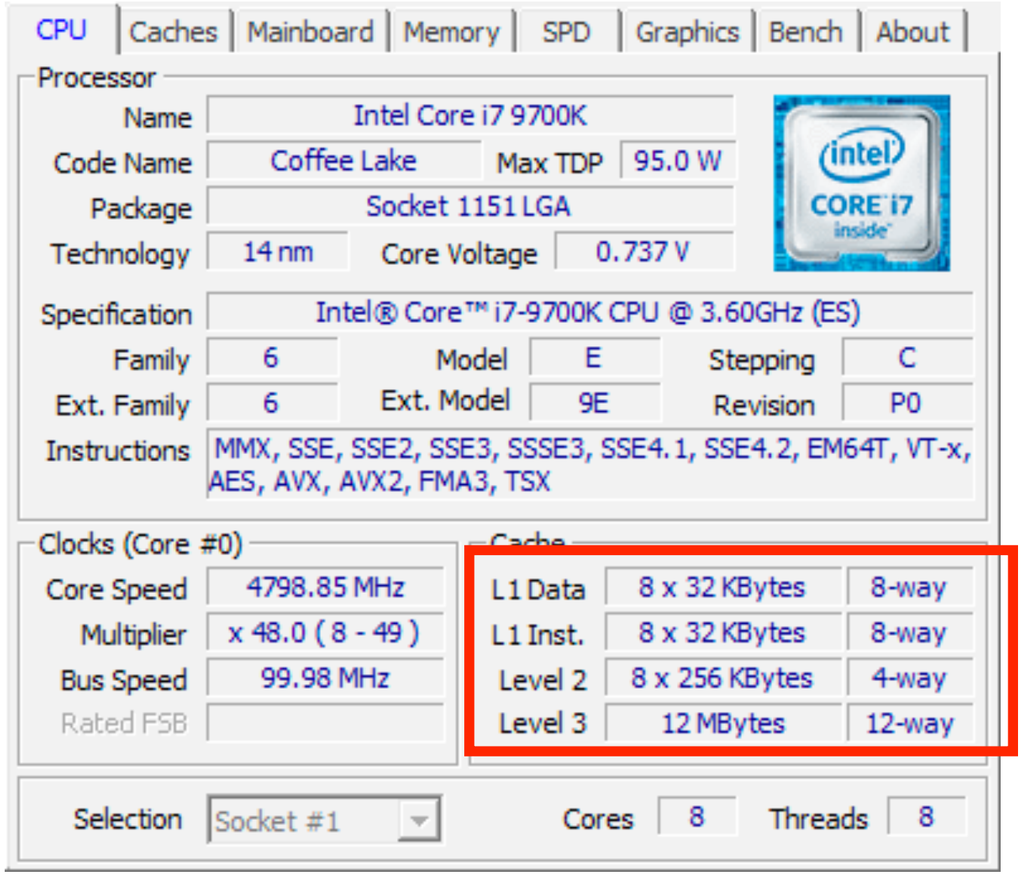
GBs

TBs

larger

L1? L2? L3?


CPU-Z - ID : wswpbb
CPU
Caches
Mainboard
Memory
SPD
Graphics
Bench
About
Processor
Name
AMD Ryzen 7 2700X
Code Name
Pinnacle Ridge
Max TDP
105 W
Package
Socket AM4 (1331)
Technology
12 nm
Core Voltage
1.36 V
Specification
AMD Ryzen 7 2700X Eight-Core Processor
Family
F
Model
8
Stepping
2
Ext. Family
17
Ext. Model
8
Revision
PiR-B2
Instructions
MMX(+), SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, SSE4A
x86-64, AMD-V, AES, AVX, AVX2, FMA3, SHA
Clocks (Core #0)
Core Speed
4290.73 MHz
Multiplier
x 43.0
Bus Speed
99.78 MHz
Rated FSB
Cache
L1 Data
8 x 32 KBytes
8-way
L1 Inst.
8 x 64 KBytes
4-way
Level 2
8 x 512 KBytes
8-way
Level 3
2 x 8192 KBytes
16-way
Selection
Processor #1
Cores
8
Threads
16
CPU-Z
Ver. 1.86.0.x64
Tools
Validate
Close


CPU
Caches
Mainboard
Memory
SPD
Graphics
Bench
About
Processor
Name
Intel Core i7 9700K
Code Name
Coffee Lake
Max TDP
95.0 W
Package
Socket 1151 LGA
Technology
14 nm
Core Voltage
0.737 V
Specification
Intel® Core™ i7-9700K CPU @ 3.60GHz (ES)
Family
6
Model
E
Stepping
C
Ext. Family
6
Ext. Model
9E
Revision
P0
Instructions
MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, VT-x,
AES, AVX, AVX2, FMA3, TSX
Clocks (Core #0)
Core Speed
4798.85 MHz
Multiplier
x 48.0 (8 - 49)
Bus Speed
99.98 MHz
Rated FSB
Cache
L1 Data
8 x 32 KBytes
8-way
L1 Inst.
8 x 32 KBytes
8-way
Level 2
8 x 256 KBytes
4-way
Level 3
12 MBytes
12-way
Selection
Socket #1
Cores
8
Threads
8

How can deeper memory hierarchy help in performance?

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has "perfect" memory, the CPI is just 1. Now, in addition to DDR4, whose latency 26 ns, we also got a 2-level SRAM caches with
  - it's 1st-level one at latency of 0.5ns and can capture 90% of the desired data/ instructions.
  - the 2nd-level at latency of 5ns and can capture 60% of the desired data/instructions

What's the average CPI (pick the most close one)?

A. 2
B. 4
C. 8
D. 16
E. 32

How can deeper memory hierarchy help in performance?

- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has "perfect" memory, the CPI is just 1. Now, in addition to DDR4, whose latency 26 ns, we also got a 2-level SRAM caches with
  - it's 1st-level one at latency of 0.5ns and can capture 90% of the desired data/ instructions.
  - the 2nd-level at latency of 5ns and can capture 60% of the desired data/instructions

  What's the average CPI (pick the most close one)?

  A. 2

  B. 4

  C. 8

  D. 16

  E. 32

$$1 + (1 - 90\%) \times [10 + (1 - 60\%) \times 52 + 30\% \times (10 + (1 - 60\%) \times 52)] = 5 \ cycles$$

Memory Hierarchy

fastest

< 1ns

a few ns

tens of ns

us/ms

Processor

Processor Core

Registers

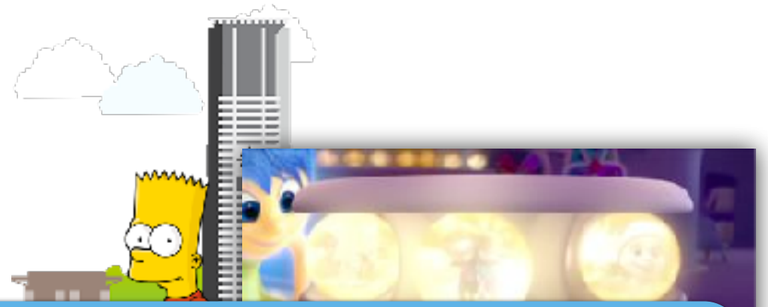
SRAM $

DRAM

Storage

TBs

larger



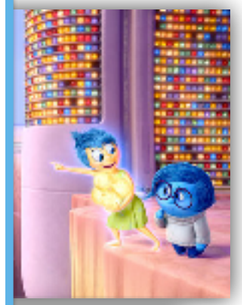
fastest

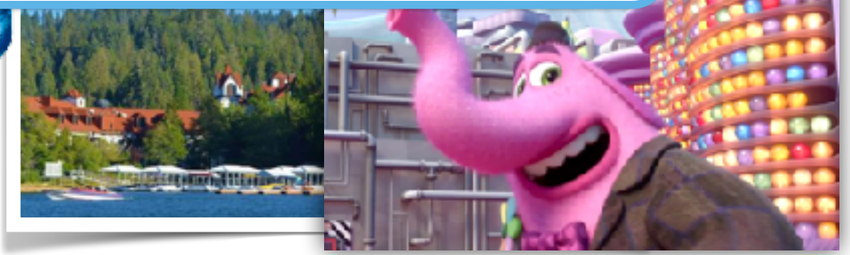
L1 $

L2 $

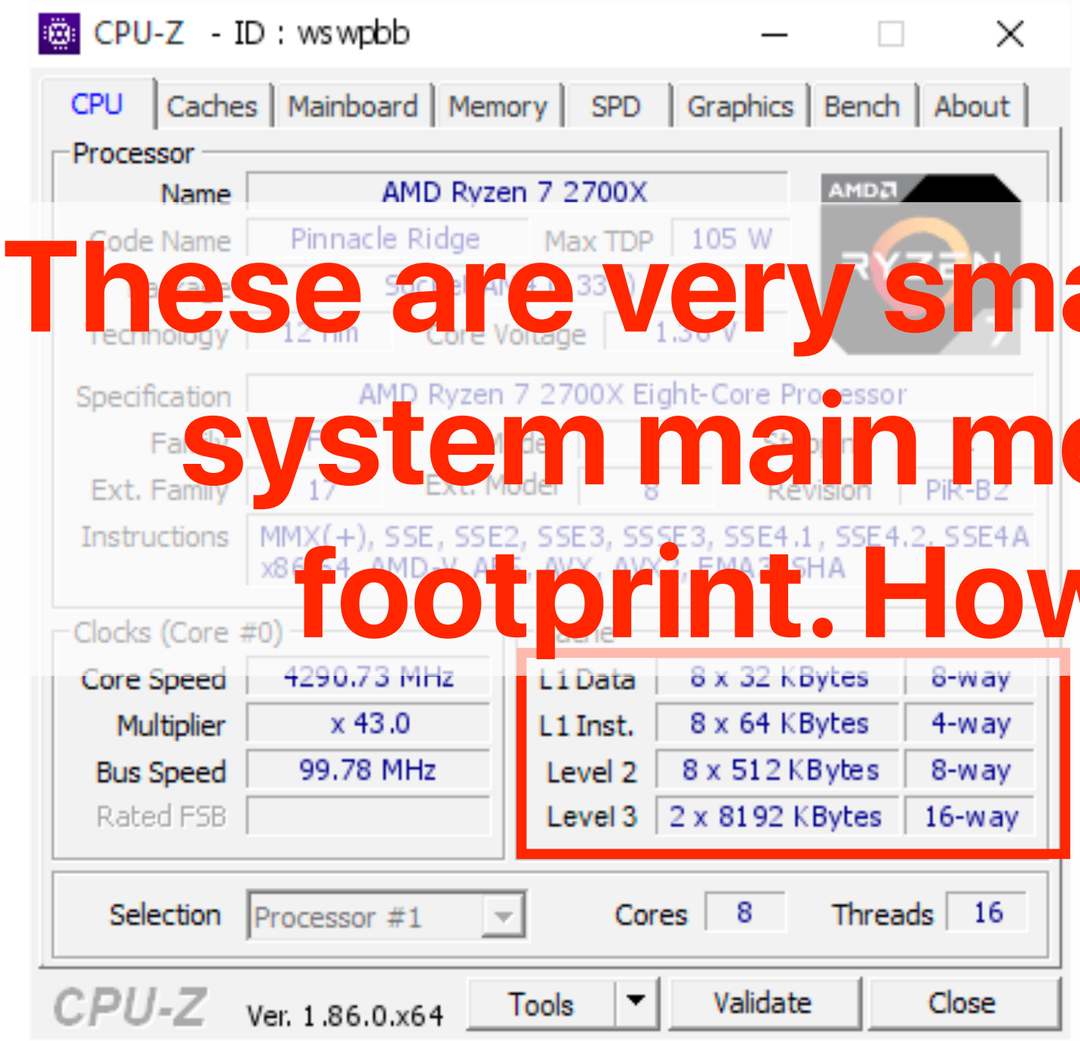
L3 $

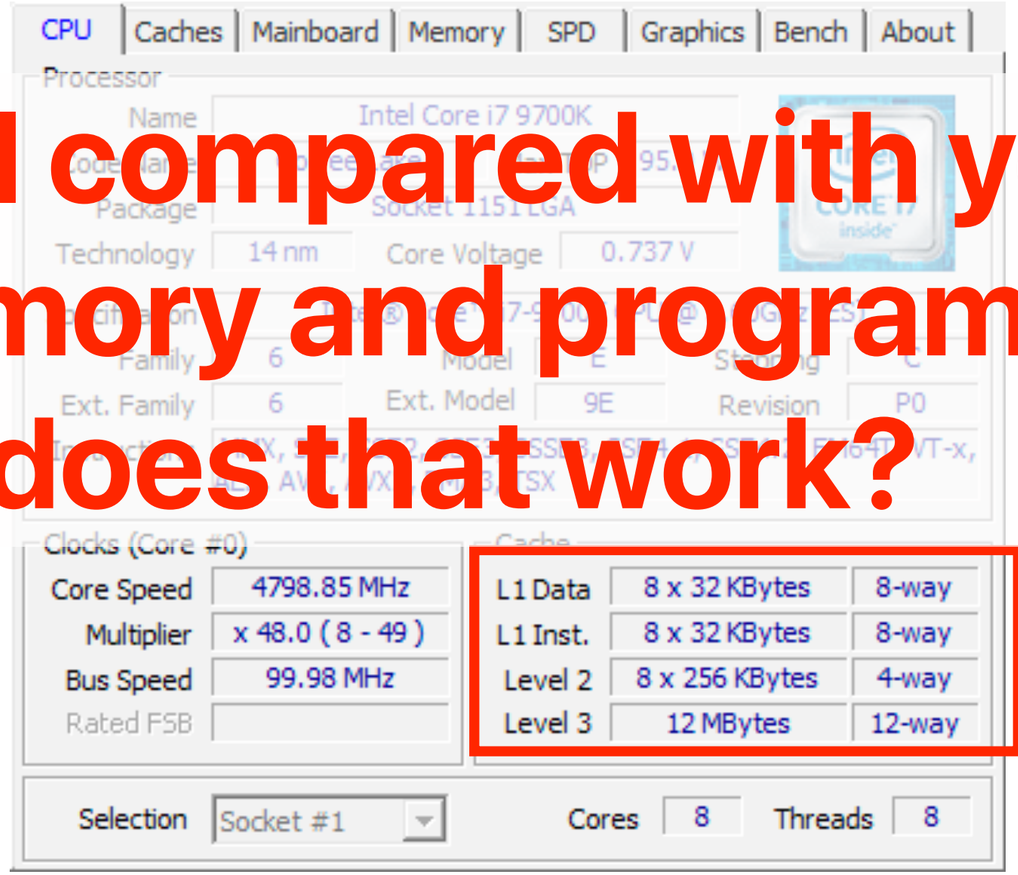
larger





L1? L2? L3?





These are very small compared with your system main memory and program footprint. How does that work?

Why adding small SRAMs would work?

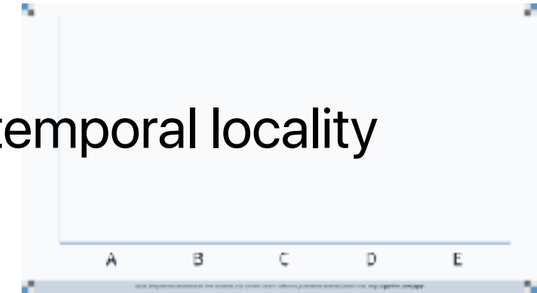
Because of localities of memory references!

Data locality

- Which description about locality of arrays `matrix` and `vector` in the following code is the **most accurate**?

```
for(uint32_t i = 0; i < m; i++) {
     result = 0;
    for(uint32_t j = 0; j < n; j++) {
          result += matrix[i][j]*vector[j];
    }
    output[i] = result;
}
```

A. Access of `matrix` has temporal locality, `vector` has spatial locality
B. Both `matrix` and `vector` have temporal locality, and `vector` also has spatial locality
C. Access of `matrix` has spatial locality, `vector` has temporal locality
D. Both `matrix` and `vector` have spatial locality and temporal locality
E. Both `matrix` and `vector` have spatial locality, and `vector` also has temporal locality



How do you prepare closed-book exams?

- Review questions from prior years — **Temporal locality**
- Review the whole chapter — **Spatial locality**
- Practice similar questions — **Spatial locality**
- Practice many times — **Temporal locality**

Data locality

- Which description about locality of arrays `matrix` and `vector` in the following code is the **most accurate**?

```
for(uint32_t i = 0; i < m; i++) {
     result = 0;
    for(uint32_t j = 0; j < n; j++) {
         result += matrix[i][j]*vector[j];
    }
    output[i] = result;
}
```

A. Access of `matrix` has temporal locality, `vector` has spatial locality
B. Both `matrix` and `vector` have temporal locality, and `vector` also has spatial locality
C. Access of `matrix` has spatial locality, `vector` has temporal locality
D. Both `matrix` and `vector` have spatial locality and temporal locality
E. Both `matrix` and `vector` have spatial locality, and `vector` also has temporal locality

Data locality

- Which description about locality of arrays `matrix` and `vector` in the following code is the **most accurate**?

```
for(uint32_t i = 0; i < m; i++) {
     result = 0;
    for(uint32_t j = 0; j < n; j++) {
          result += matrix[i][j]*vector[j];
    }
    output[i] = result;
}
```

spatial locality:
matrix[0][0], matrix[0][1], matrix[0][2], ...
vector[0], vector[1], ... , vector[n]
temporal locality:
reuse of vector[0], vector[1], ... ,

A. Access of `matrix` has temporal locality, `vector` has spatial locality
B. Both `matrix` and `vector` have temporal locality, and `vector` also has spatial locality
C. Access of `matrix` has spatial locality, `vector` has temporal locality
D. Both `matrix` and `vector` have spatial locality and temporal locality
E. Both `matrix` and `vector` have spatial locality, and `vector` also has temporal locality

Code locality

```
for(uint32_t i = 0; i < m; i++) {
    result = 0;
    for(uint32_t j = 0; j < n; j++) {
        result += matrix[i][j]*vector[j];
    }
    output[i] = result;
}
```

```
i = 0;
while(i < m) {
    result = 0;
    j = 0;
    while(j < n) {
        a = matrix[i][j];
        b = vector[j];
        temp = a*b;
        result = result + temp;
    }
    output[i] = result;
    i++;
}
```

keep going to the next instruction — spatial locality

repeat many times — temporal locality!

Locality

- Spatial locality — application tends to visit nearby stuffs in the memory
  - Code — the current instruction, and then the next
  - Data — the current element in an array, then the next
- Temporal locality — application revisit the same thing again and again
  - Code — loops, frequently invoked functions
  - Data — the same data can be read/write many times

Most of time, your program is just visiting a limited amount of data/instructions within a given timeframe

Locality and cache design

- The cache must be able to get chunks of near-by items every time to exploit spatial locality

- The cache must be able to keep a frequently used block for a while to exploit temporal locality

Architecting the Cache: capture the localities!

Locality and cache design

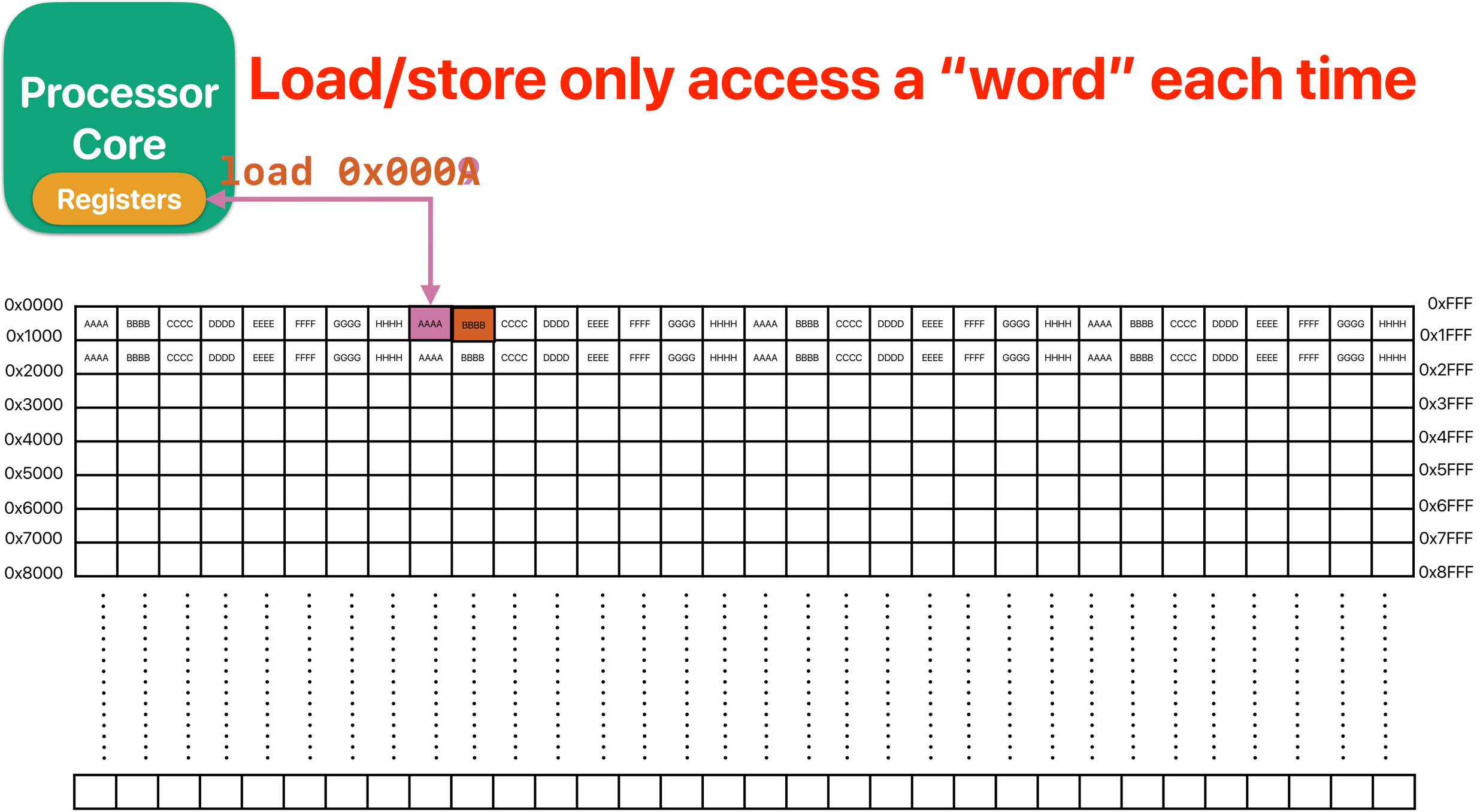
- The cache must be able to get chunks of near-by items every time to exploit spatial locality

  We need to keep a block of data every time we put things in the cache

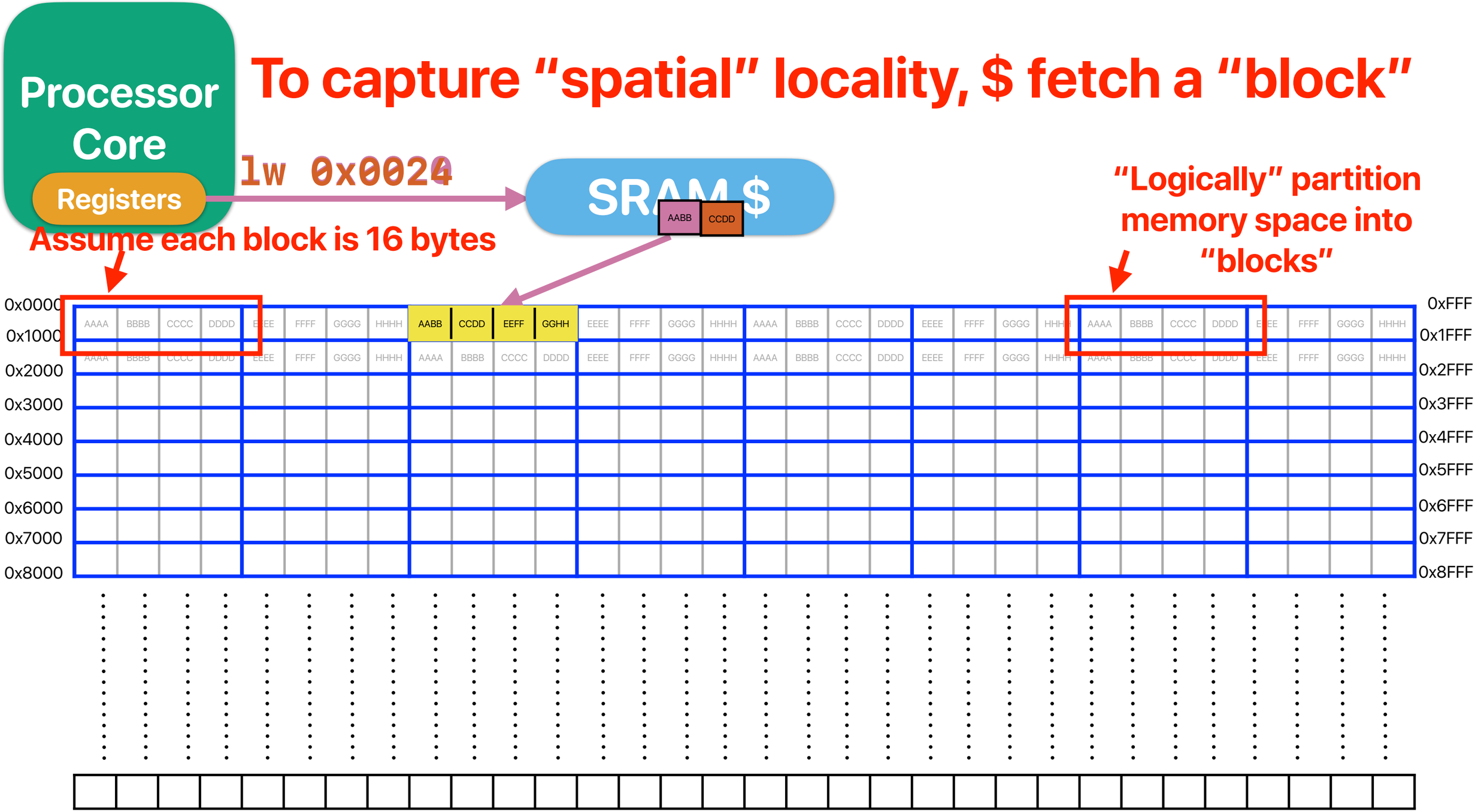
- The cache must be able to keep a frequently used block for a while to exploit temporal locality

  We need to keep multiple blocks of data in the cache

Load/store only access a “word” each time



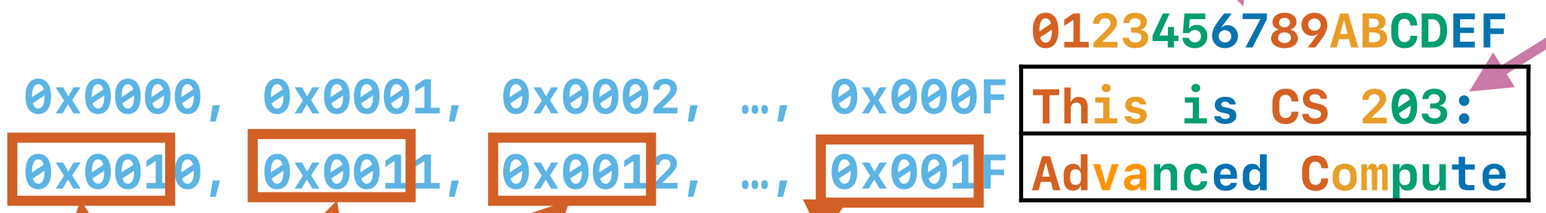
To capture "spatial" locality, $ fetch a "block"



What's a block?

the offset of the byte within a block

the data in memory



the byte addresses of each byte in the block

the address in each block starts with the same "prefix"

How to tell who is there?

the common address prefix in each block

tag array	0123456789ABCDEF
0x000	This is CS 203:
0x001	Advanced Compute
0xF07	r Architecture!
0x100	This is CS 203:
0x310	Advanced Compute
0x450	r Architecture!
0x006	This is CS 203:
0x537	Advanced Compute
0x266	r Architecture!
0x307	This is CS 203:
0x265	Advanced Compute
0x80A	r Architecture!
0x620	This is CS 203:
0x630	Advanced Compute
0x705	r Architecture!
0x216	This is CS 203:

How to tell w

Processor Core

Registers

block offset

tag

lw 0x0008

lw 0x4048

0x404 not found,
go to lower-level memory

Valid Bit

Dirty Bit

Tell if the block here can be used

Tell if the block here is modified

Valid Bit	Dirty Bit	tag	data (0123456789ABCDEF)
1	1	0x000	This is CS 203:
1	1	0x001	Advanced Compute
1	0	0xF07	r Architecture!
0	1	0x100	This is CS 203:
1	1	0x310	Advanced Compute
1	1	0x450	r Architecture!
0	1	0x006	This is CS 203:
0	1	0x537	Advanced Compute
1	1	0x266	r Architecture!
1	1	0x307	This is CS 203:
0	1	0x265	Advanced Compute
0	1	0x80A	r Architecture!
1	1	0x620	This is CS 203:
1	1	0x630	Advanced Compute
1	0	0x705	r Architecture!
0	1	0x216	This is CS 203:

The complexity of search the matching tag—

$O(n)$— will be slow if our cache size grows!

Can we search things faster?

—hash table! $O(1)$

Announcement

- Assignment #1 due this Friday
  - Assignments SHOULD BE done/submitted individually
  - We will drop your least performing assignment
- Reading quiz #4 due next Wednesday
- Office Hours
  - Walk-in, no appointment is necessary
  - Hung-Wei/Prof. Usagi: TW 11a-12p (WCH 406 or on Zoom)
  - Jinyoung Choi: M 3p-4p, Th 2p-3p (WCH 110 or on Zoom)

Computer
Science &
Engineering

203

つづく