

Performance (4) & Memory (I): (1) Memories bring back you...

Hung-Wei Tseng

Disney · PIXAR

INSIDE OUT

[GET DISNEY+](#)[▶ TRAILER](#)

PG 2015 • 1h 35m • Coming of age, Family, Animation

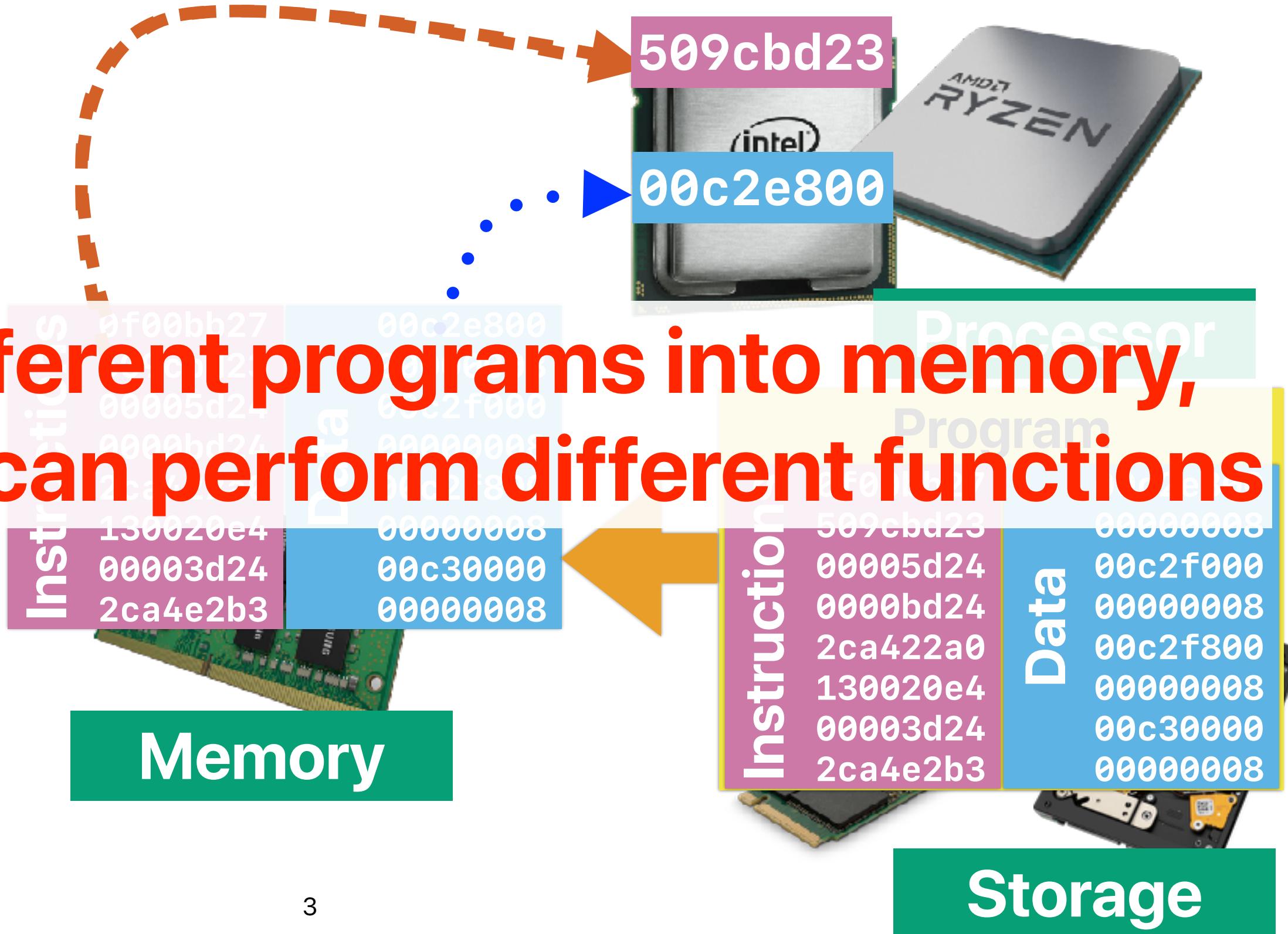
When 11-year-old Riley moves to a new city, her Emotions team up to help her through the transition. Joy, Fear, Anger, Disgust and Sadness work together, but when Joy and Sadness get lost, they must journey through unfamiliar places to get back home.



Recap: von Neumann Architecture



By loading different programs into memory,
your computer can perform different functions



Recap: Summary of CPU Performance Equation

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

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

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

$$\text{Speedup} = \frac{\text{Execution Time}_X}{\text{Execution Time}_Y}$$

- 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**

Amdahl's Law

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



$$Speedup_{enhanced} = \frac{Execution\ Time_{baseline}}{Execution\ Time_{enhanced}} = \frac{1}{(1 - f) + \frac{f}{s}}$$

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

$$\begin{aligned} Speedup_{max}(f, \infty) &= \frac{1}{(1 - f)} \\ Speedup_{max}(f_1, \infty) &= \frac{1}{(1 - f_1)} \\ Speedup_{max}(f_2, \infty) &= \frac{1}{(1 - f_2)} \\ Speedup_{max}(f_3, \infty) &= \frac{1}{(1 - f_3)} \\ Speedup_{max}(f_4, \infty) &= \frac{1}{(1 - f_4)} \end{aligned}$$

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

Outline

- Misleading performance metrics
- The basics of memory hierarchy

“Fair” Comparisons

Andrew Davison. Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers. In Humour the Computer, MITP, 1995
V. Sze, Y. -H. Chen, T. -J. Yang and J. S. Emer. How to Evaluate Deep Neural Network Processors: TOPS/W (Alone) Considered Harmful. In IEEE Solid-State Circuits Magazine, vol. 12, no. 3, pp. 28-41, Summer 2020.

Round #1

Bard

Who is Hung-Wei Tseng?

Hung-Wei Tseng is an associate professor in the Department of Electrical and Computer Engineering and a cooperating faculty of the Department of Computer Science and Engineering at University of California, Riverside. He is now leading the Extreme Storage & Computer Architecture Laboratory. He is interested in diverse research topics that allow applications or programmers to more efficiently use modern heterogeneous hardware components. Together with his students, they have demonstrated how intelligent storage devices can help improve performance, power and energy for data-intensive systems.

Their work has been recognized by IEEE Micro "Top Picks from the 2019 Computer Architecture Conferences" (IEEE MICRO Top Picks 2020) and Facebook Research Award, 2018. Prior to joining UCR, he served as an assistant professor for the Department of Computer Science and the Department of Electrical and Computer Engineering at NC State University. He was a PostDoc of the Non-volatile Systems Laboratory and a lecturer of the Department of Computer Science and Engineering at University of California, San Diego with Professor Steven Swanson.

Sources - Learn more
1. ictra.ugr.es/~htseng/
2. www.nvsl.io/portfolio/hung-wei-tseng/

Like Dislike Share Google It

How important is Computer Architecture in Computer Science?

1 answer/6 secs



I am unable to determine who Hung-Wei Tseng is. Can you please provide me with more context or details about the individual in question?

HT Who is Hung-Wei Tseng at UCR?

Hung-Wei Tseng is an Associate Professor of Computer Science and Engineering at the University of California, Riverside (UCR). His research interests are in the area of computer networks and distributed systems, with a focus on improving network performance and security. Some of his recent research work has focused on topics such as network measurement and monitoring, network optimization, and network security. Hung-Wei Tseng has been a faculty member at UCR since 2013, and he has received several awards and recognitions for his research contributions.

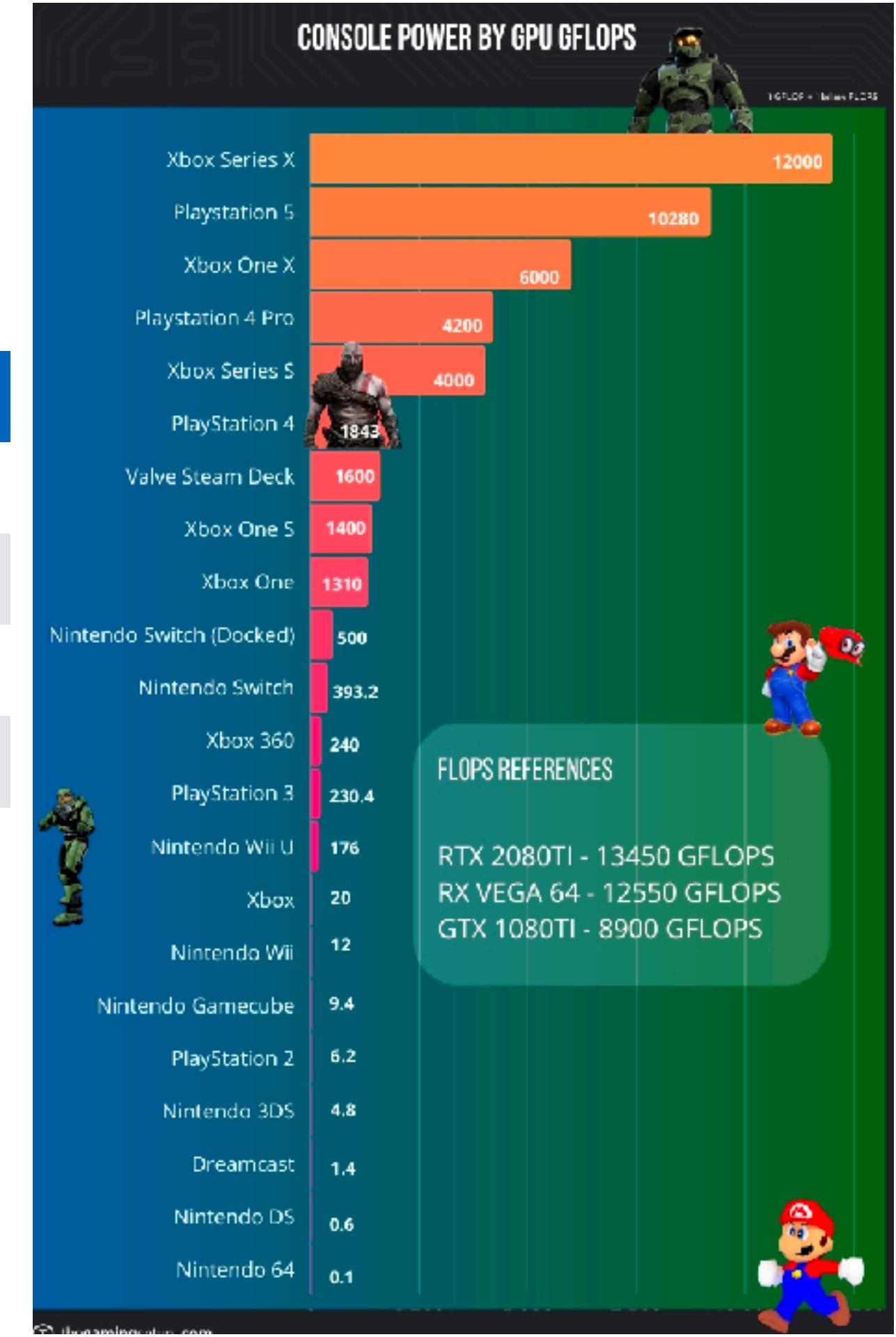
Regenerate response

How important is Computer Architecture in Computer Science?

1 answer/18 secs

TFLOPS (Tera FLoating-point Operations Per Second)

	TFLOPS	clock rate
Switch	1	921 MHz
PS5	10.28	2.23 GHz
XBox Series X	12	1.825 GHz
GeForce RTX 3090	40	1.395 GHz



Let's measure the FLOPS of matrix multiplications

```
for(i = 0; i < ARRAY_SIZE; i++) {  
    for(j = 0; j < ARRAY_SIZE; j++) {  
        for(k = 0; k < ARRAY_SIZE; k++) {  
            c[i][j] += a[i][k]*b[k][j];  
        }  
    }  
}
```

Floating point operations:

$$i \times j \times k \times 2$$

Given $i = j = k = 2048$

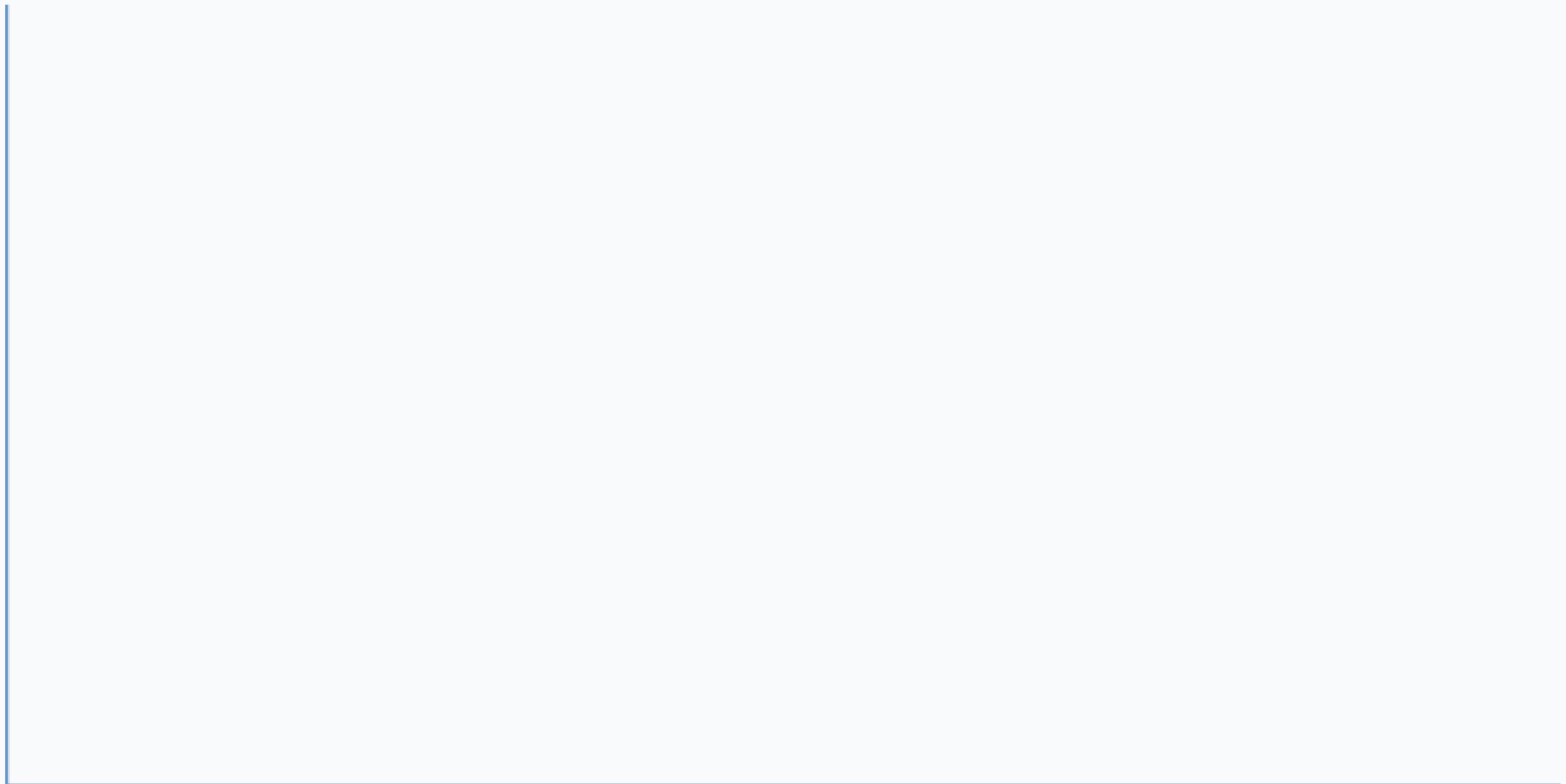
$$2^{3 \times 11} \times 2 = 2^{34} \quad \text{FLOPs in total}$$

$$FLOPS = \frac{2^{34}}{ET_{seconds}}$$

How reflective is FLOPs?

- Given the FLOPs number measured, how many of the followings are true?
 - The FLOPs number remains the same on each architecture if we change the data size
 - The FLOPs number remains the same on each architecture if we change the data type to double
 - The FLOPs number remains the same on each architecture if we change the algorithm implementation
 - The FLOPs number reflects the performance ratio of different architectures when executing floating point applications
- A. 0
B. 1
C. 2
D. 3
E. 4





A

B

C

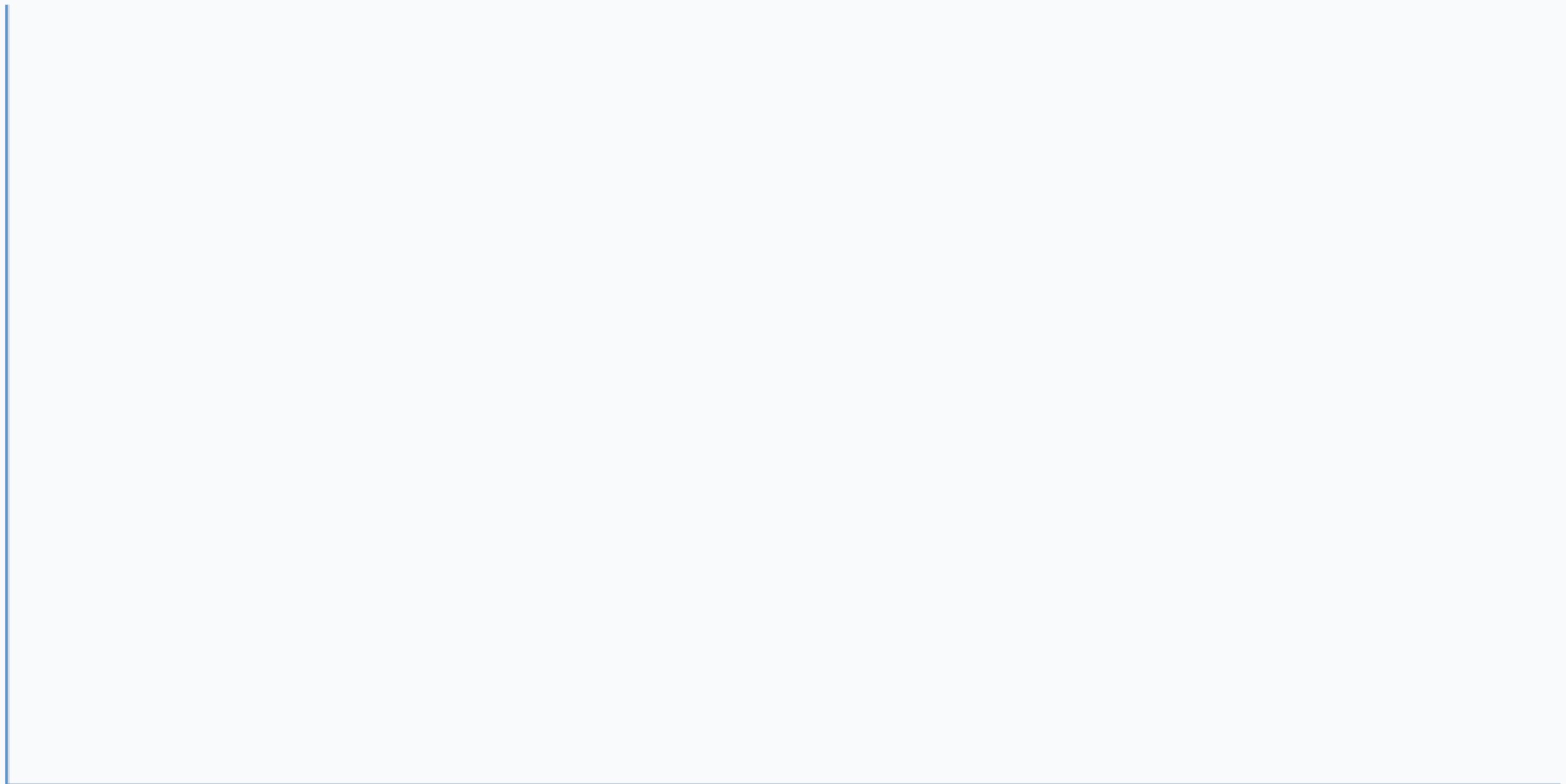
D

E

How reflective is FLOPs?

- Given the FLOPs number measured, how many of the followings are true?
 - The FLOPs number remains the same on each architecture if we change the data size
 - The FLOPs number remains the same on each architecture if we change the data type to double
 - The FLOPs number remains the same on each architecture if we change the algorithm implementation
 - The FLOPs number reflects the performance ratio of different architectures when executing floating point applications
- A. 0
B. 1
C. 2
D. 3
E. 4





A

B

C

D

E

How reflective is FLOPs?

- Given the FLOPs number measured, how many of the followings are true?
 - The FLOPs number remains the same on each architecture if we change the data size
CPI may change as data size changes
 - The FLOPs number remains the same on each architecture if we change the data type to double
CPI may change as data type changes
 - The FLOPs number remains the same on each architecture if we change the algorithm implementation
CPI and IC may change as algorithm changes
 - The FLOPs number reflects the performance ratio of different architectures when executing floating point applications
Different architecture has different ICs
- A. 0
- B. 1
- C. 2
- D. 3
- E. 4

TFLOPS (Tera FLoating-point Operations Per Second)

- 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

	TFLOPS	clock rate
Switch	1	921 MHz
PS5	10.28	2.23 GHz
XBox Series X	12	1.825 GHz
GeForce RTX 3090	40	1.395 GHz

nvidia.com

NVIDIA

Artificial Intelligence Computing Leadership from NVIDIA

CLOUD & DATA CENTER

PRODUCTS ▾

SOLUTIONS ▾

APPS ▾

FOR DEVELOPERS

TECHNOLOGIES ▾

Tesla V100

AI TRAINING AI INFERENCE HPC DATA CENTER GPUs SPECIFICATIONS

Deep Learning Training in Less Than a Workday

Configuration	Time to Solution in Hours
8X Tesla V100	5.1 Hours
8X Tesla P100	15.5 Hours

Time to Solution in Hours—Lower Is Better

Server Config: Dual Xeon E5-2699 v4 2.6 GHz | 8X NVIDIA® Tesla® P100 or V100 | ResNet-50 Training on MXNet for 90 Epochs with 1.28M ImageNet Dataset.

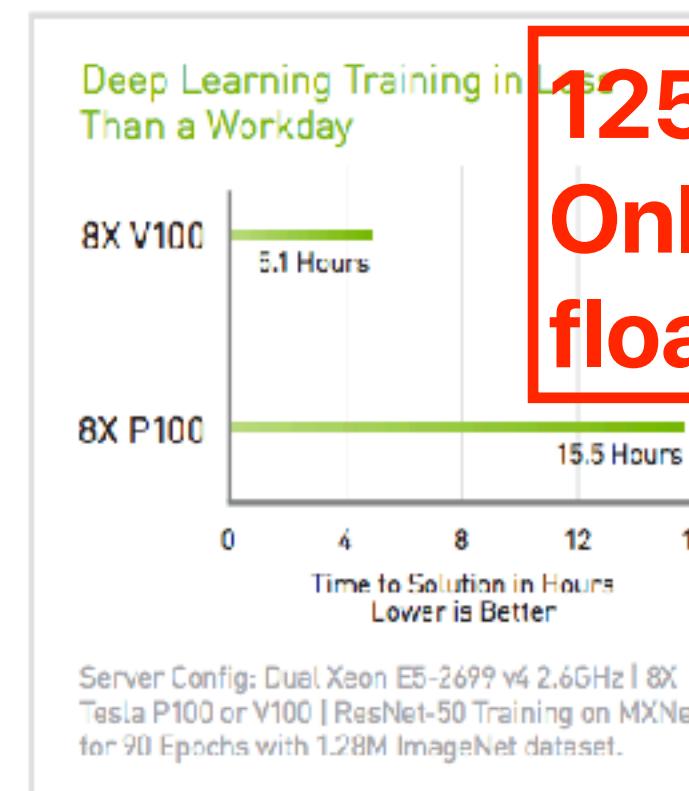
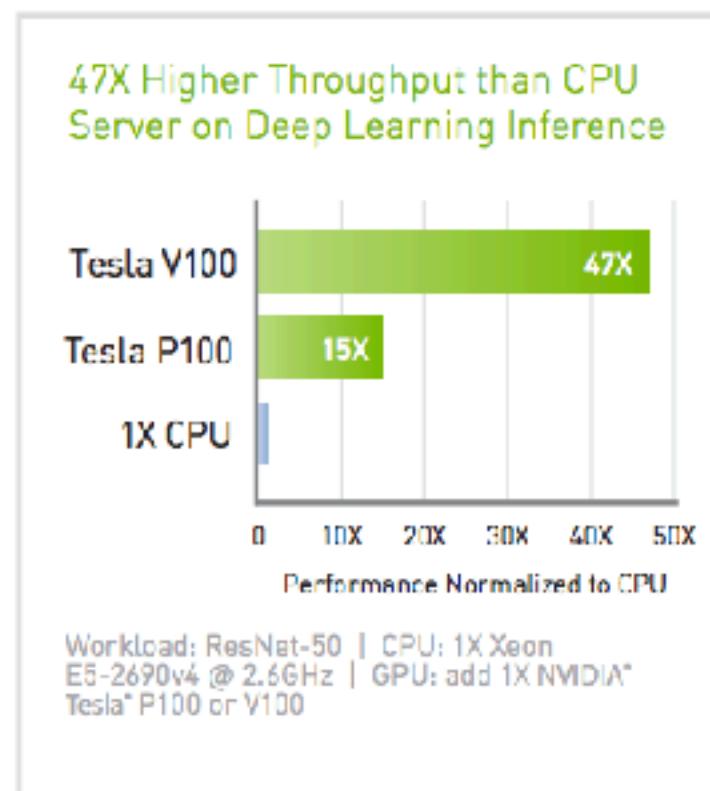
AI TRAINING

From recognizing speech to training virtual personal assistants and teaching autonomous cars to drive, data scientists are taking on increasingly complex challenges with AI. Solving these kinds of problems requires training deep learning models that are exponentially growing in complexity, in a practical amount of time.

With 640 Tensor Cores, Tesla V100 is the world's first GPU to break the 100 teraFLOPS (TFLOPS) barrier of deep learning performance. The next generation of NVIDIA NVLink™ connects multiple V100 GPUs at up to 300 GB/s to create the world's most powerful computing servers. AI models that would consume weeks of computing resources on previous systems can now be trained in a few days. With this dramatic reduction in training time, a whole new world of problems will now be solvable with AI.

The Most Advanced Data Center GPU Ever Built.

NVIDIA® Tesla® V100 is the world's most advanced data center GPU ever built to accelerate AI, HPC, and graphics. Powered by NVIDIA Volta, the latest GPU architecture, Tesla V100 offers the performance of up to 100 CPUs in a single GPU—enabling data scientists, researchers, and engineers to tackle challenges that were once thought impossible.



**125 TFLOPS
Only @ 16-bit floating point**

SPECIFICATIONS



**Tesla V100
PCIe**



**Tesla V100
SXM2**

GPU Architecture	NVIDIA Volta	
NVIDIA Tensor Cores	640	
NVIDIA CUDA® Cores	5,120	
Double-Precision Performance	7 TFLOPS	7.8 TFLOPS
Single-Precision Performance	14 TFLOPS	15.7 TFLOPS
Tensor Performance	112 TFLOPS	125 TFLOPS
GPU Memory	32GB /16GB HBM2	
Memory Bandwidth	900GB/sec	
ECC	Yes	
Interconnect Bandwidth	32GB/sec	300GB/sec
System Interface	PCIe Gen3	NVIDIA NVLink
Form Factor	PCIe Full Height/Length	SXM2
Max Power	375W	300W

1 GPU Node Replaces Up To 54 CPU Nodes

Node Replacement: HPC Mixed Workload

They try to tell it's the better AI hardware

<https://blogs.nvidia.com/blog/2017/04/10/ai-drives-rise-accelerated-computing-datacenter/>

	K80 2012	TPU 2015	P40 2016
Inferences/Sec <10ms latency	$^{1/13}X$	1X	2X
Training TOPS	6 FP32	NA	12 FP32
Inference TOPS	6 FP32	90 INT8	48 INT8
On-chip Memory	16 MB	24 MB	11 MB
Power	300W	75W	250W
Bandwidth	320 GB/S	34 GB/S	350 GB/S

Round #2

Bard

A screenshot of the Bard web interface. On the left, a sidebar lists options: 'Reset chat', 'Bard Activity', 'FAQ', and 'Help & support'. The main area shows a search result for 'Who is teaching CS203 at UCR?'. It includes a profile picture of Hung-Wei Tseng, a brief bio stating he is an assistant professor teaching CS203, and contact information like address, phone number, and email. At the bottom, there's a search bar with 'Who is Hung-Wei Tseng?' and a note about displaying accurate or sensitive information from Google's API.

Who is teaching CS203 at UCR?

Hung-Wei Tseng is teaching CS203 at UCR. He is an assistant professor in the Department of Computer Science and Engineering. He received his Ph.D. in Computer Science from the University of California, Berkeley in 2018. His research interests include machine learning, natural language processing, and artificial intelligence.

More about contact information:

- Address: Room 406, Winston-Cheng Hall, 900 University Ave., Riverside, CA 92521
- Phone: (951) 827-4122 ext 28047
- Email: htseng@ucr.edu

Who is Hung-Wei Tseng?

Don't see the display accurate or sensitive information that doesn't侵犯用户隐私。Google's API.

A screenshot of the ChatGPT web interface. On the left, a sidebar shows a list of recent chats: 'New chat', 'UCR CS203 Instructor', 'Hung-Wei Tseng', and 'HW Memory Adoption Best'. The main area shows a search result for 'Who is teaching CS203 at UCR?'. It includes a note that as an AI language model, it does not have access to real-time information about who is teaching CS203 at the University of California, Riverside (UCR). It suggests visiting UCR's official website or contacting the university's department of computer science directly for the most up-to-date information on course instructors. At the bottom, there's a search bar with 'Who is Hung-Wei Tseng?' and a note about creating a better response.

Who is teaching CS203 at UCR?

As an AI language model, I do not have access to real-time information about who is teaching CS203 at the University of California, Riverside (UCR). However, you can visit UCR's official website or contact the university's department of computer science directly for the most up-to-date information on course instructors.

Who is Hung-Wei Tseng?

Create a better response

Creating a better response... Our goal is to make it easier for everyone to learn with us. Your feedback will help us improve.

1 answer/6 secs

1 answer/6 secs

What's wrong with inferences per second?

- There is no standard on how they inference — but these affect!
 - What model?
 - What dataset?
 - Quality?
- That's why Facebook is trying to promote an AI benchmark — MLPerf

- *Pitfall: For NN hardware, Inferences Per Second (IPS) is an inaccurate summary performance metric.*

Our results show that IPS is a poor overall performance summary for NN hardware, as it's simply the inverse of the complexity of the typical inference in the application (e.g., the number, size, and type of NN layers). For example, the TPU runs the 4-layer MLP1 at 360,000 IPS but the 89-layer CNN1 at only 4,700 IPS, so TPU IPS vary by 75X! Thus, using IPS as the single-speed summary is *even more misleading* for NN accelerators than MIPS or FLOPS are for regular processors [23], so IPS should be even more disparaged. To compare NN machines better, we need a benchmark suite written at a high-level to port it to the wide variety of NN architectures. Fathom is a promising new attempt at such a benchmark suite [3].

Inference per second

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

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

	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		



Extreme Multitasking Performance

- Dual 4K external monitors
- 1080p device display
- 7 applications

What's missing in this video clip?

- The ISA of the “competitor”
- Clock rate, CPU architecture, cache size, how many cores
- How big the RAM?
- How fast the disk?

12 ways to Fool the Masses When Giving Performance Results on Parallel Computers

- Quote only 32-bit performance results, not 64-bit results.
- Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application.
- Quietly employ assembly code and other low-level language constructs.
- Scale up the problem size with the number of processors, but omit any mention of this fact.
- Quote performance results projected to a full system.
- Compare your results against scalar, unoptimized code on Crays.
- When direct run time comparisons are required, compare with an old code on an obsolete system.
- If TFLOPS rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation.
- Quote performance in terms of processor utilization, parallel speedups or TFLOPS per dollar.
- Mutilate the algorithm used in the parallel implementation to match the architecture.
- Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment.
- If all else fails, show pretty pictures and animated videos, and don't talk about performance.

How do you usually prepare a
closed book midterm?

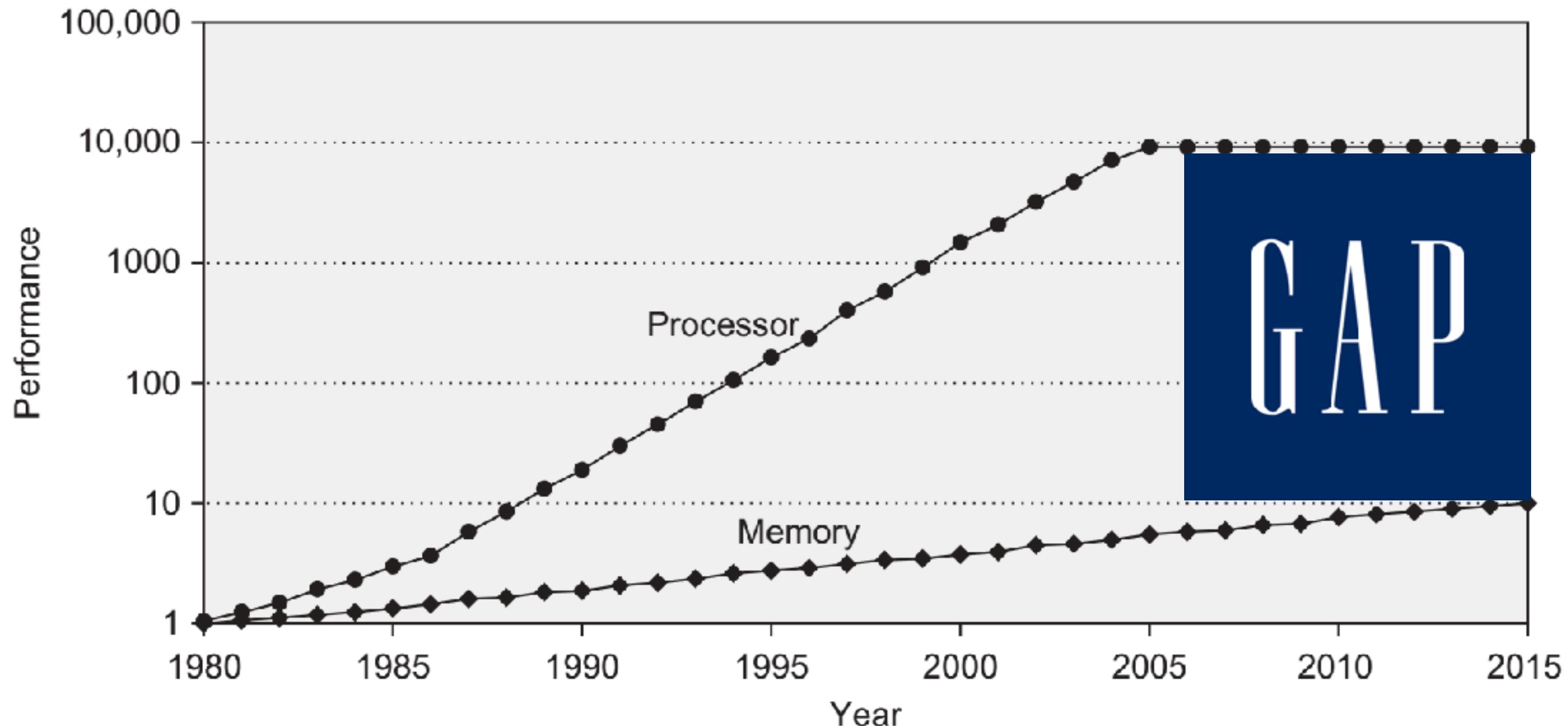


Memory Hierarchy Inside Out:

(1) Memories bring back you...

Hung-Wei Tseng

Recap: Performance gap between Processor/Memory



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

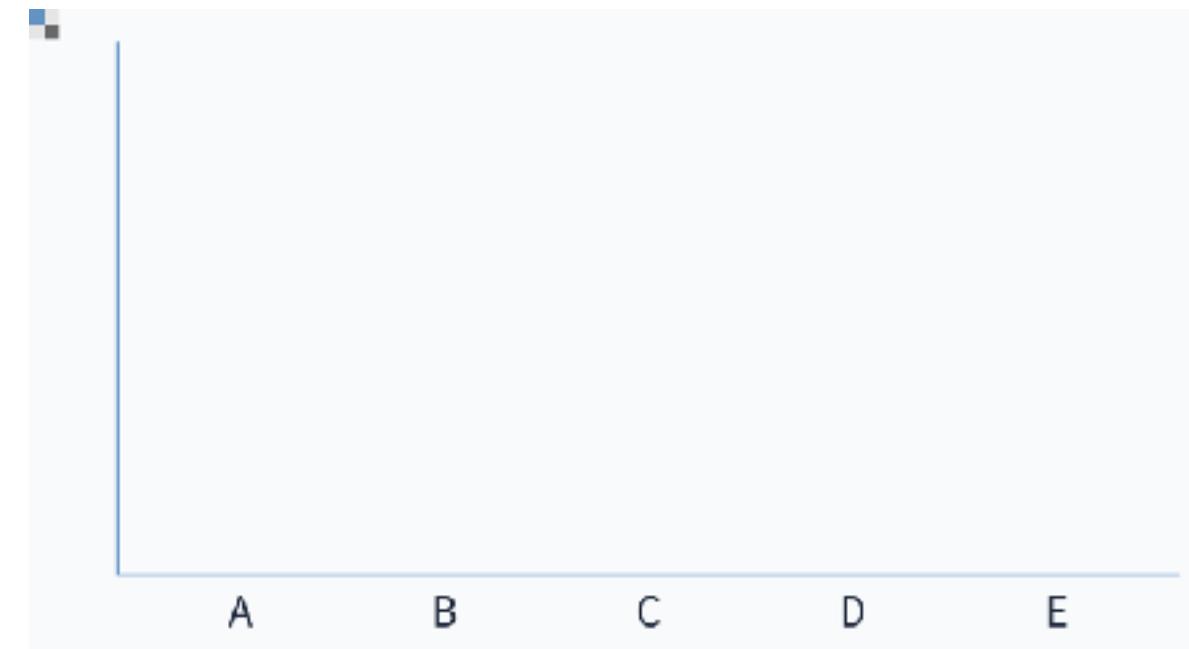
Production year	Chip size	DRAM type	Best case access time (no precharge)		Precharge needed	
			RAS time (ns)	CAS time (ns)	Total (ns)	Total (ns)
2000	256M bit	DDR1	21	21	42	63
2002	512M bit	DDR1	15	15	30	45
2004	1G bit	DDR2	15	15	30	45
2006	2G bit	DDR2	10	10	20	30
2010	4G bit	DDR3	13	13	26	39
2016	8G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

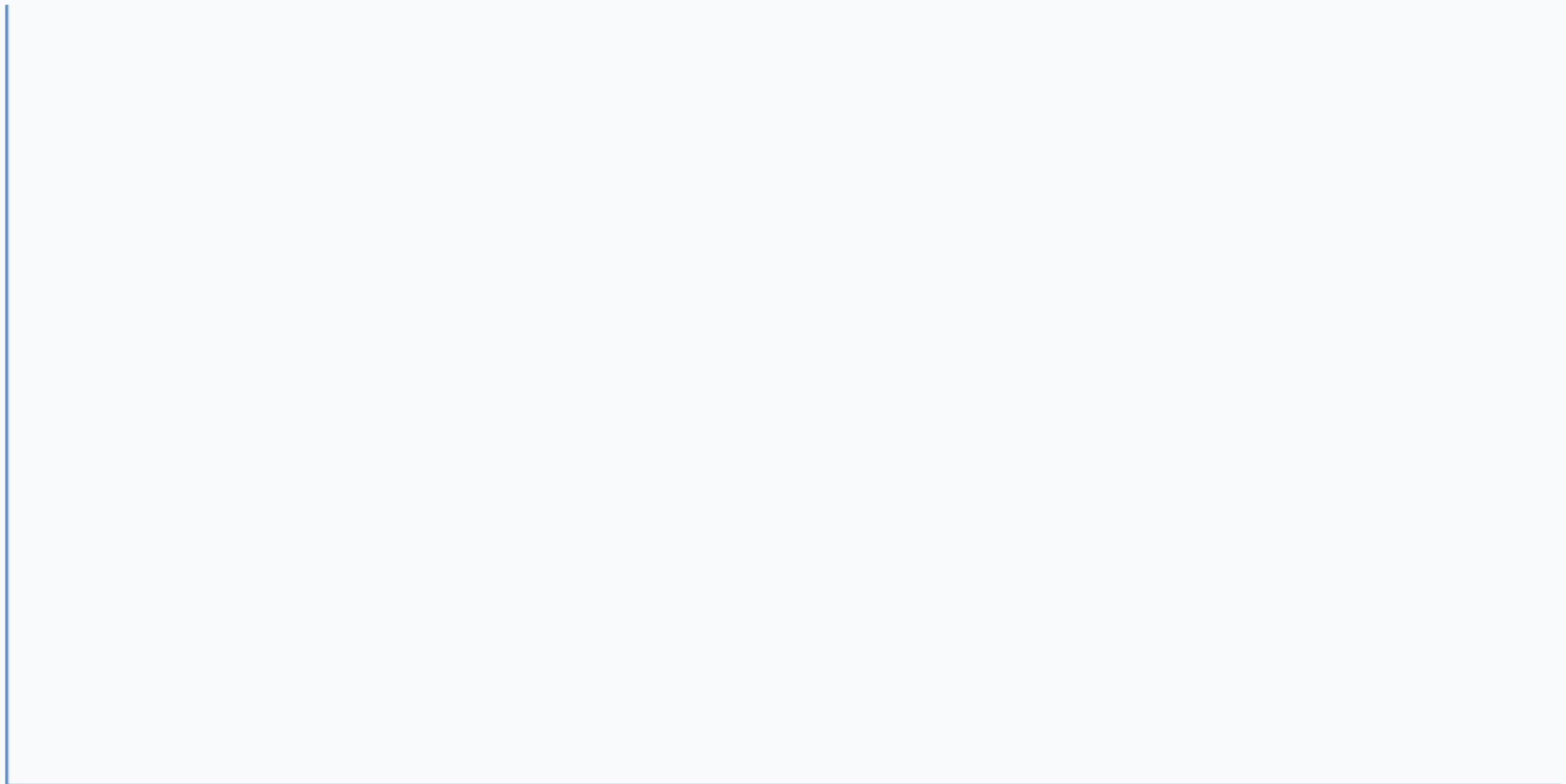
Figure 2.4 Capacity and access times for DDR SDRAMs by year of production. Access time is for a random memory word and assumes a new row must be opened. If the row is in a different bank, we assume the bank is precharged; if the row is not open, then a precharge is required, and the access time is longer. 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





A

B

C

D

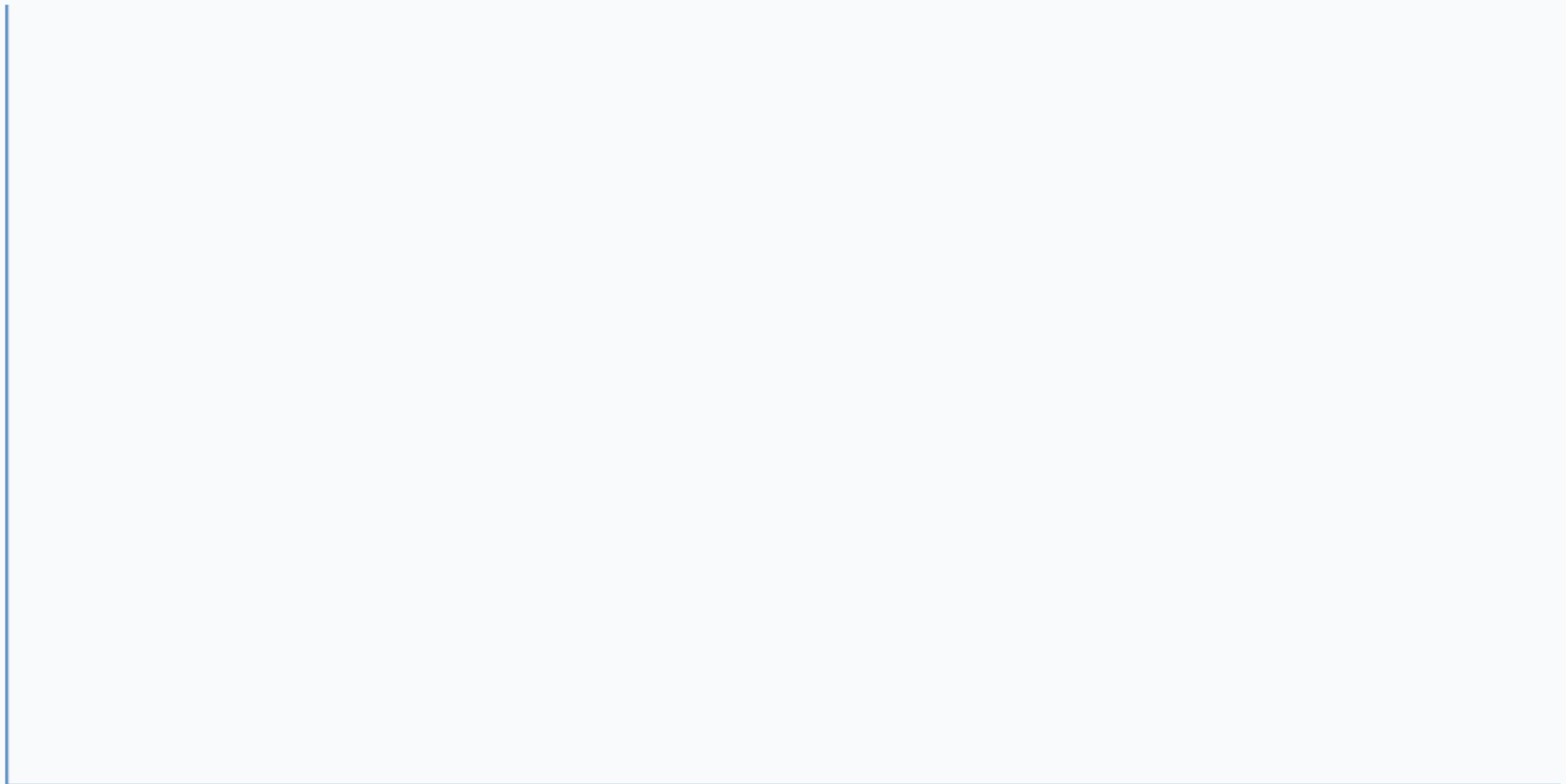
E

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





A

B

C

D

E

The impact of “slow” memory

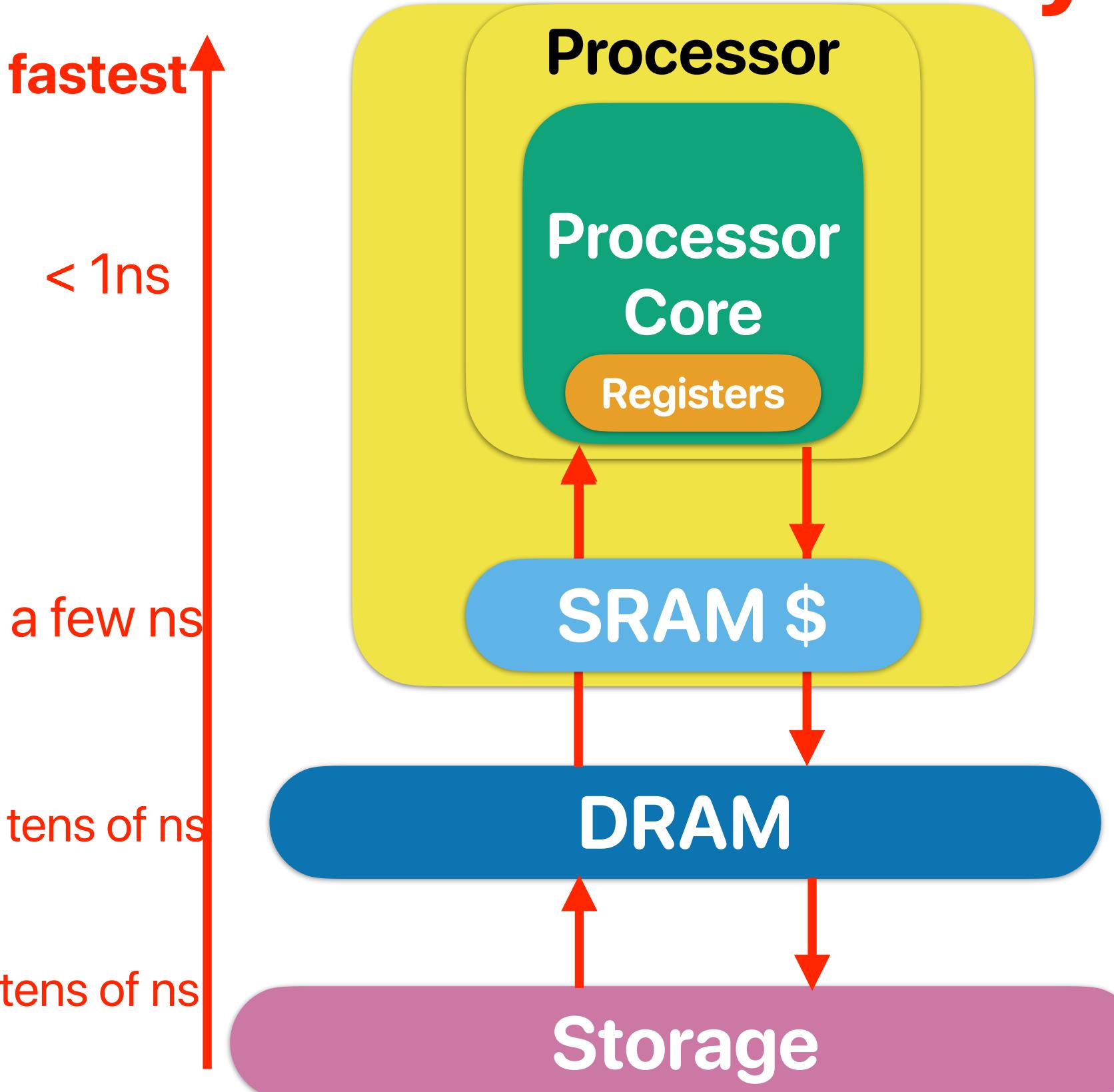
- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has “perfect” memory, the CPI is just 1. Now, consider we have DDR4 and the program is well-behaved that precharge is never necessary — the access latency is simply 26 ns. What’s the average CPI (pick the most close one)?
 - 9
 - 17
 - 27
 - 35
 - 69
- $$1 + \boxed{100\% \times (52)} + 30\% \times 52 = 68.6 \text{ cycles}$$
- 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 \$\$\$

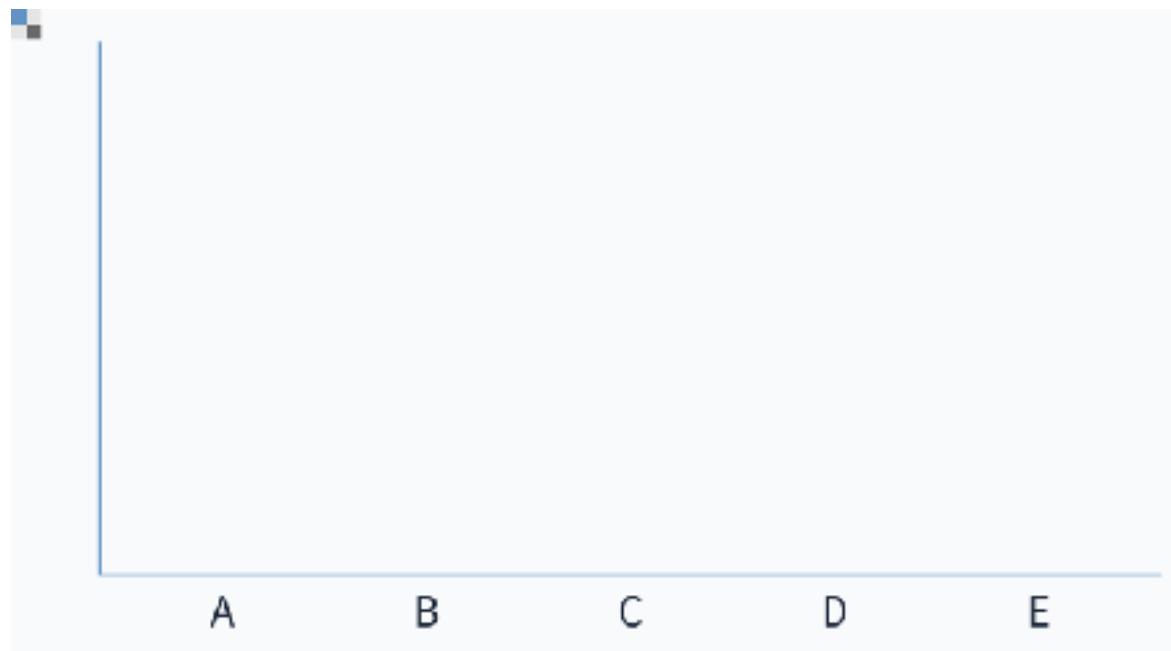
Memory Hierarchy

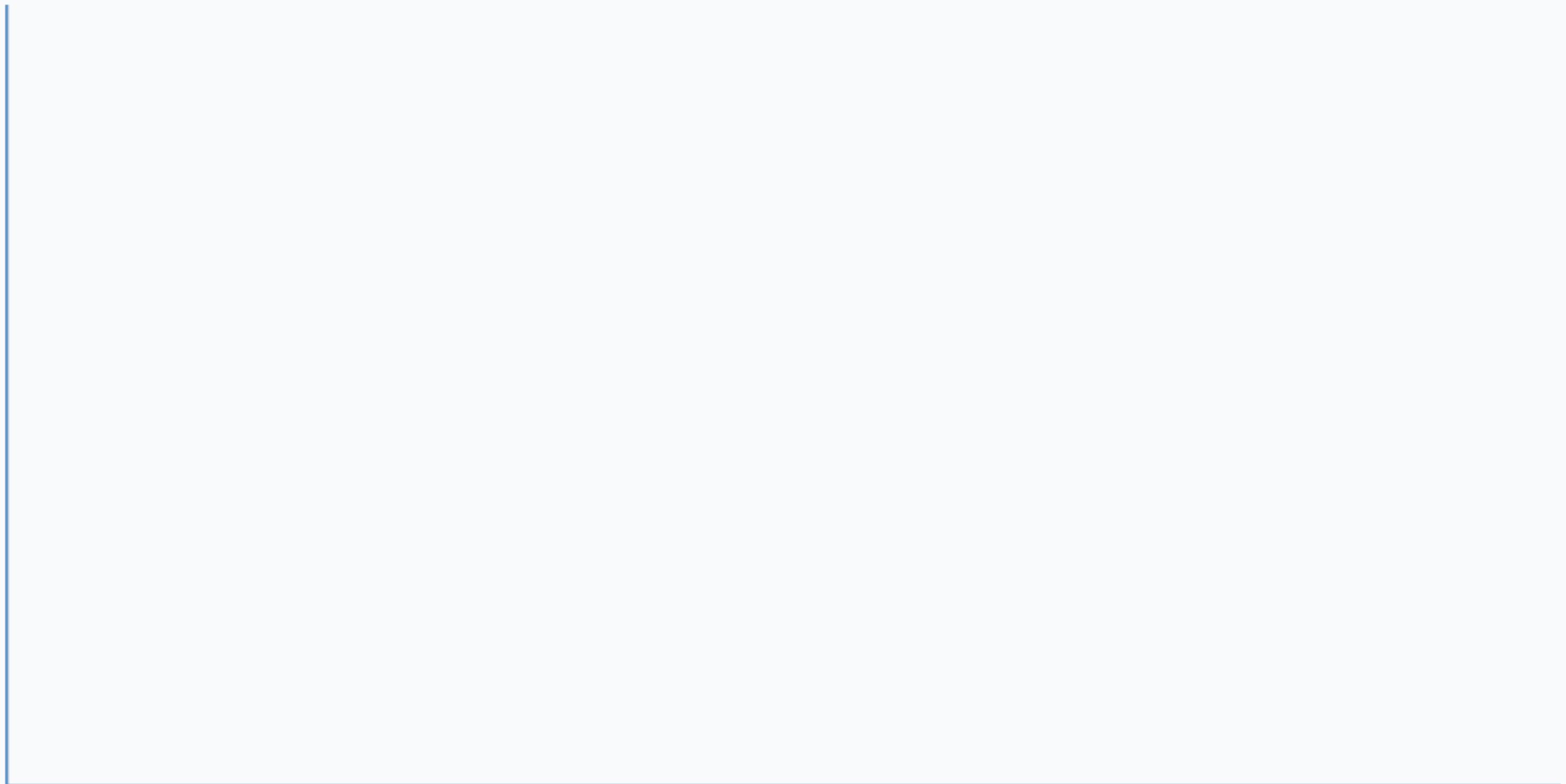


How can memory hierarchy help in performance?

- 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





A

B

C

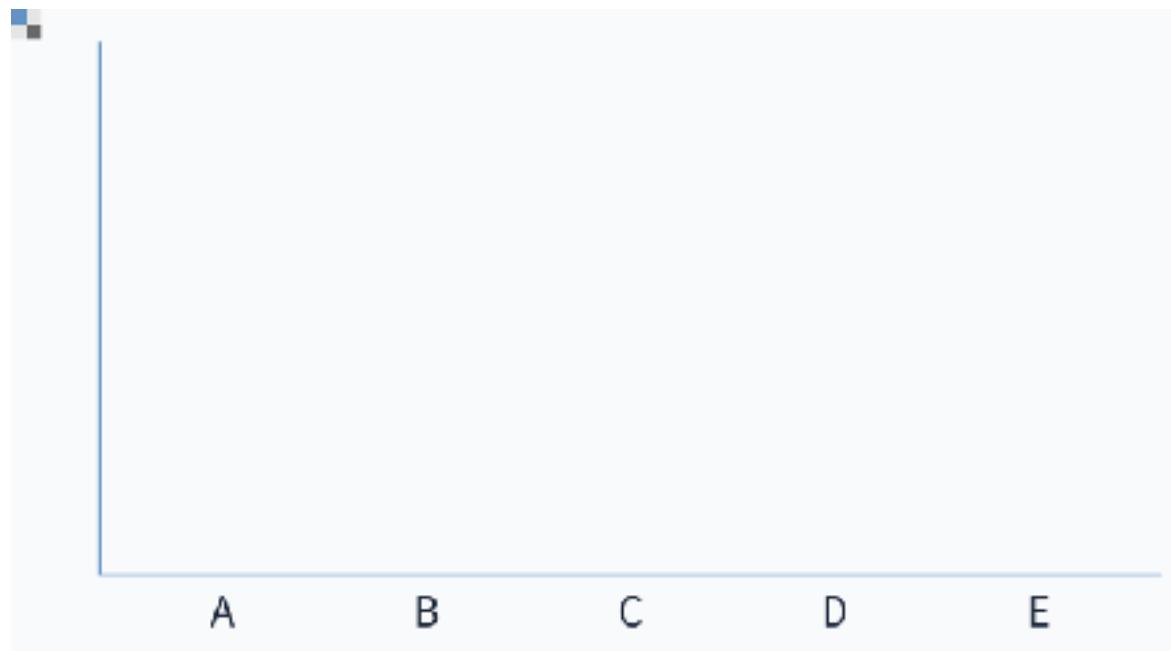
D

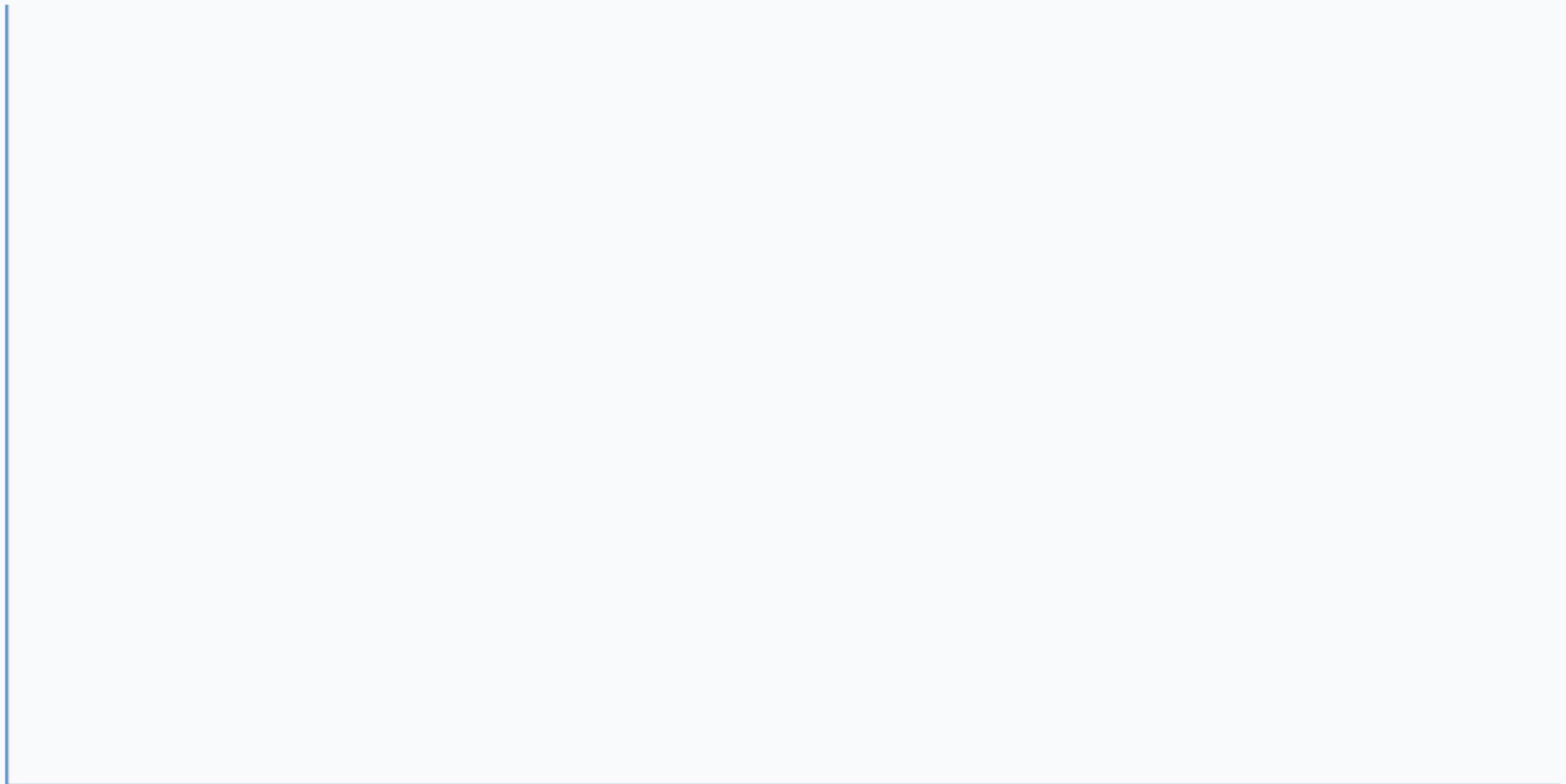
E

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





A

B

C

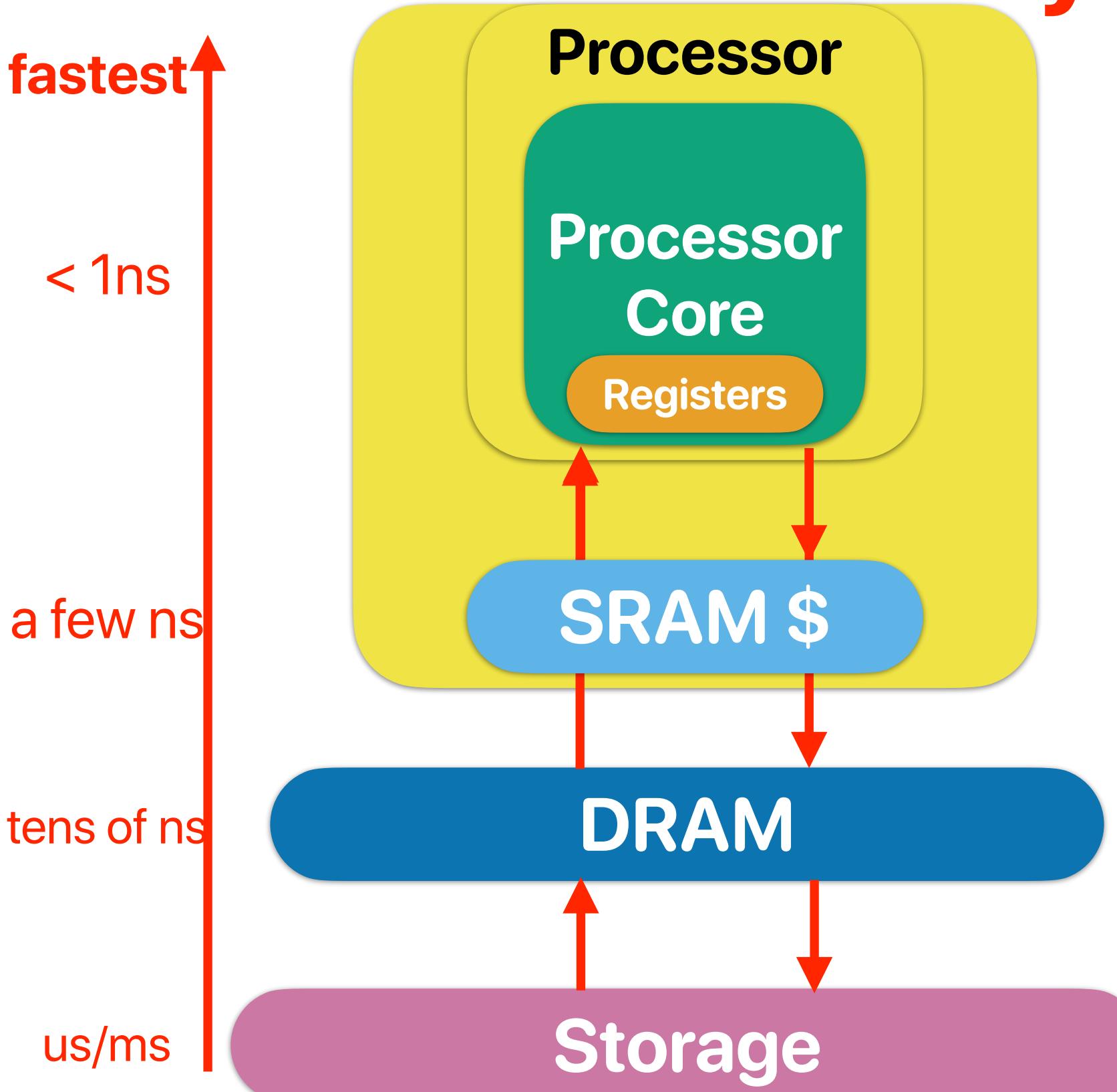
D

E

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)?
 - 2
 - 4 $1 + (1 - 90\%) \times [100\% \times (52) + 30\% \times 52] = 7.76 \text{ cycles}$
 - 8
 - 16
 - 32

Memory Hierarchy



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
Ext. Family	17	Ext. Model	8
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
Ext. Family	6	Ext. Model	9E
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

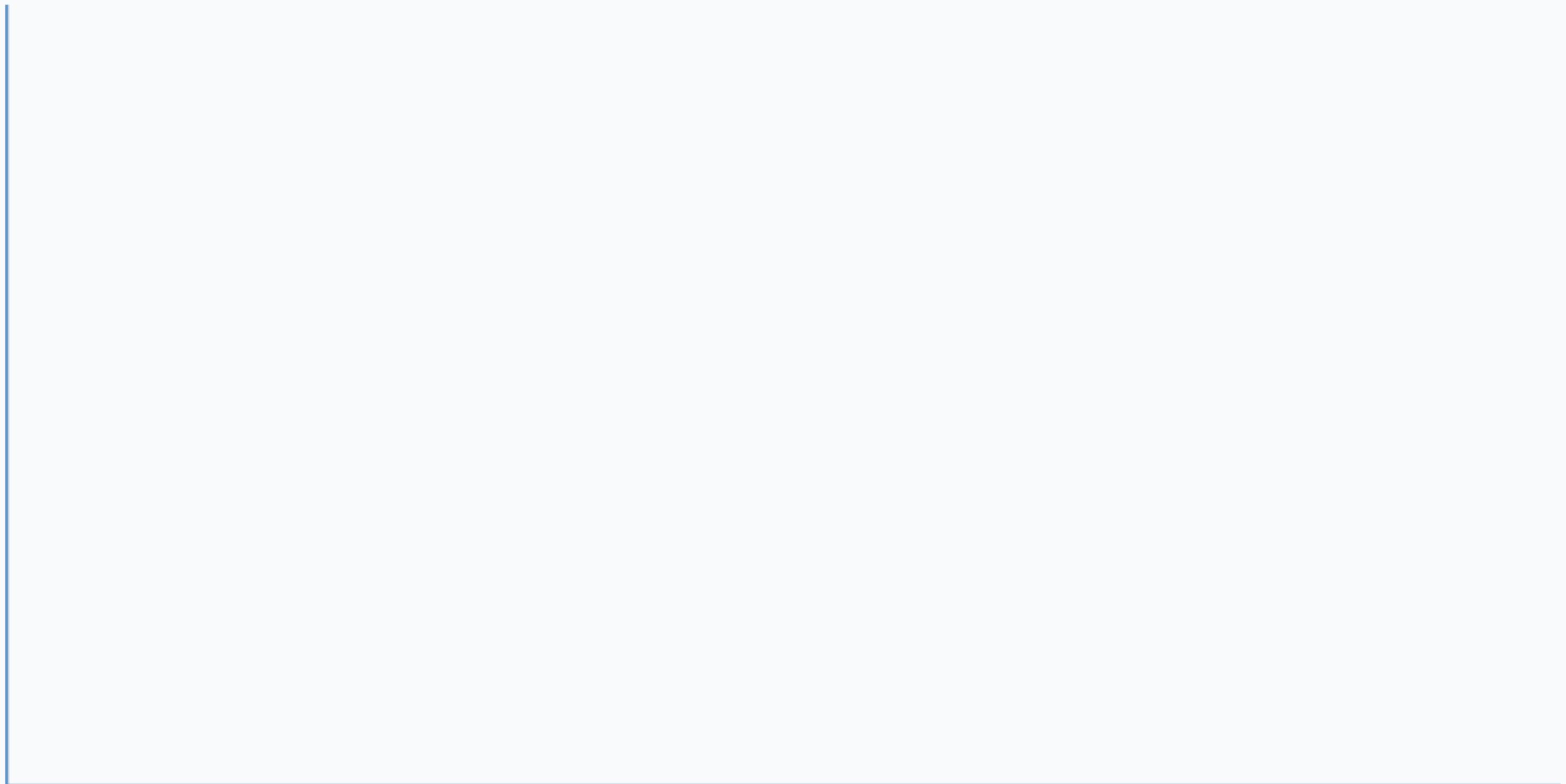
How can deeper memory hierarchy help in performance?

- 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





A

B

C

D

E

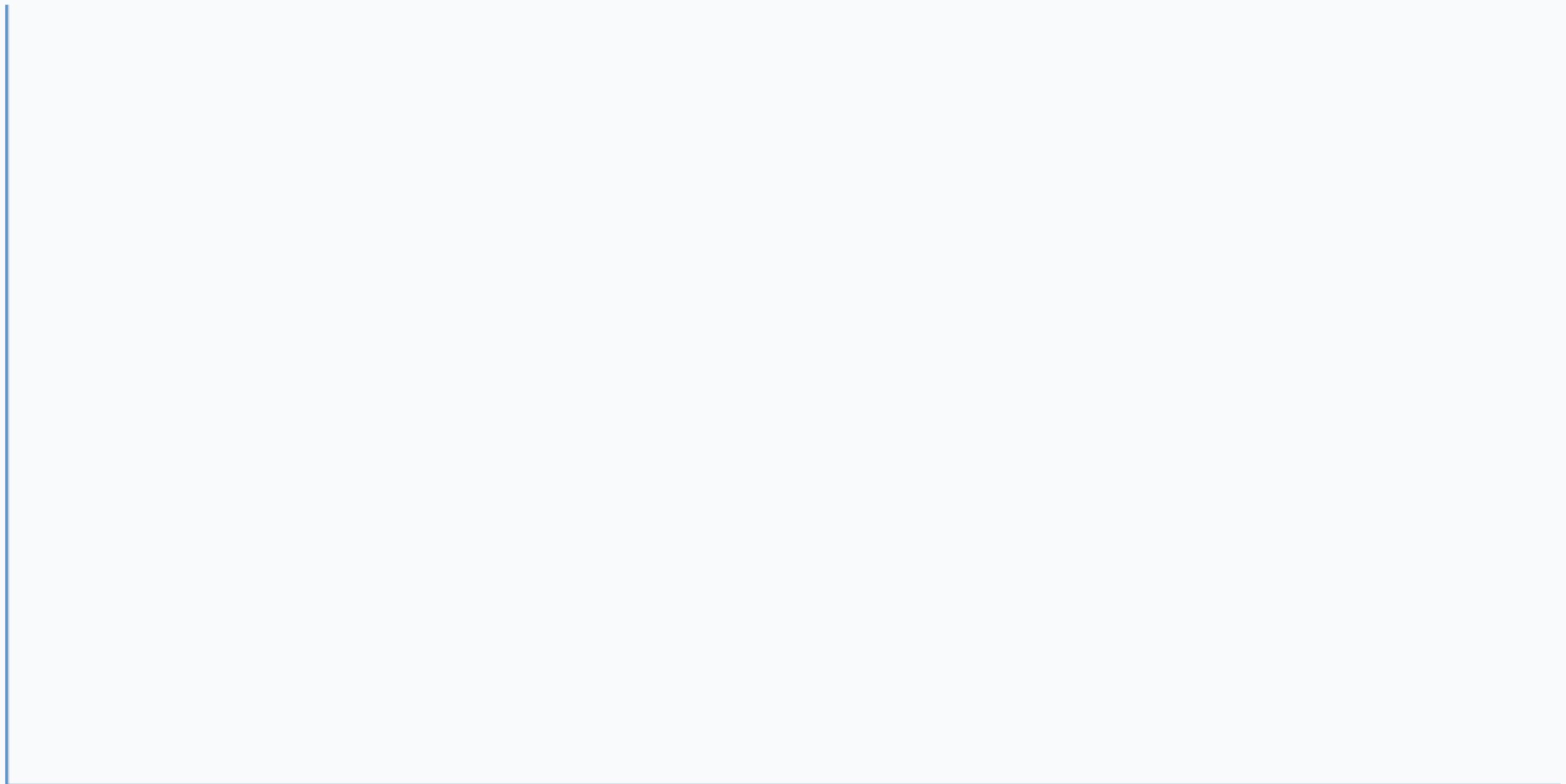
How can deeper memory hierarchy help in performance?

- 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





A

B

C

D

E

How can deeper memory hierarchy help in performance?

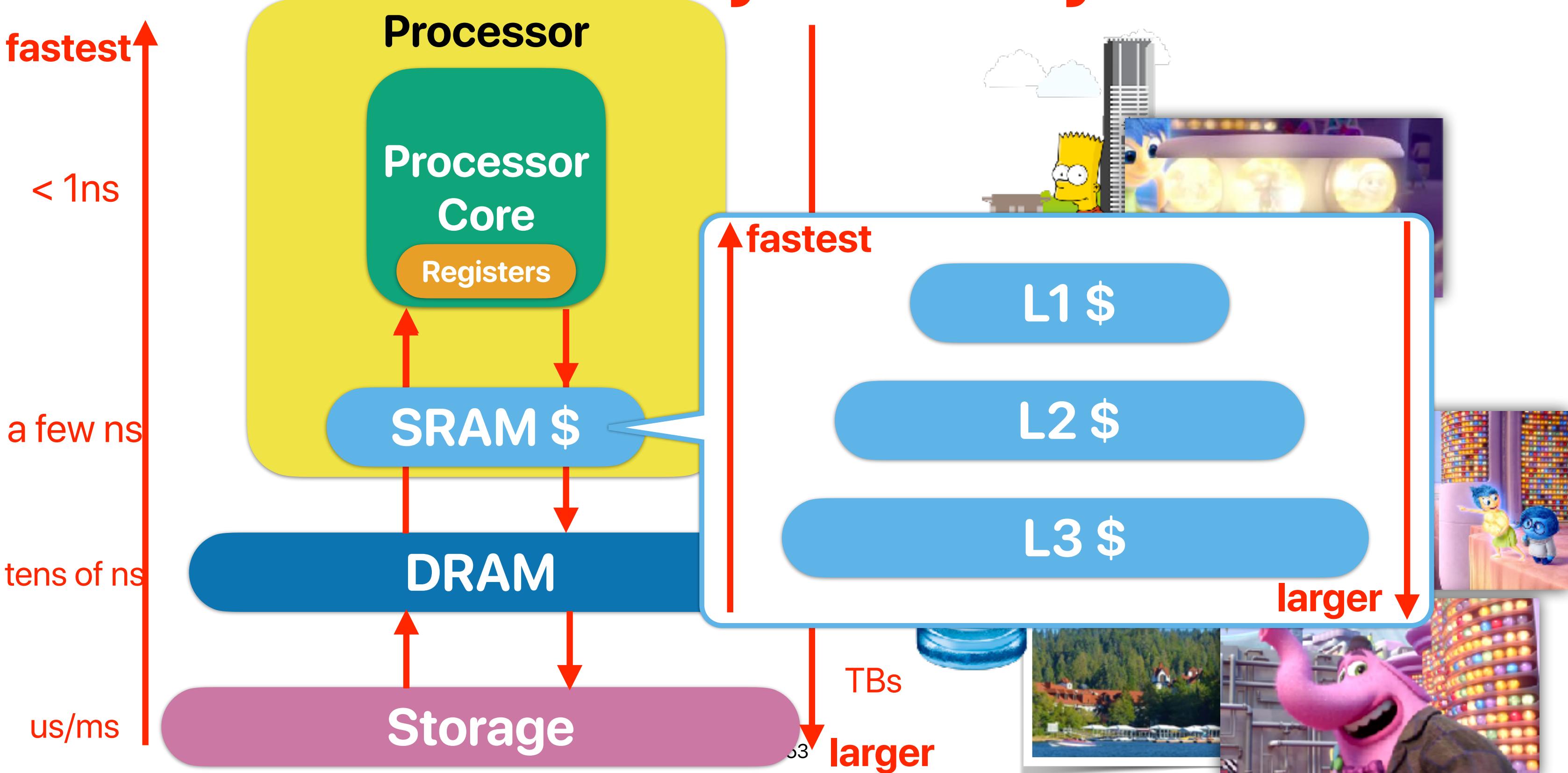
- Assume that we have a processor running @ 2 GHz and a program with 30% of load/store instructions. If the computer has “perfect” memory, the CPI is just 1. Now, in addition to DDR4, whose latency 26 ns, we also got a 2-level SRAM caches with
 - it’s 1st-level one at latency of 0.5ns and can capture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

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

Memory Hierarchy



L1? L2? L3?

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

The image shows two CPU-Z interface windows side-by-side. Both windows have a top navigation bar with tabs: CPU, Caches, Mainboard, Memory, SPD, Graphics, Bench, and About. The left window is for an AMD Ryzen 7 2700X (Pinnacle Ridge) and the right window is for an Intel Core i7 9700K (Comet Lake). Both processors have 8 cores and 16 threads.

AMD Ryzen 7 2700X Cache Details:

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

Intel Core i7 9700K Cache Details:

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

The cache sections for both processors are highlighted with red boxes.

Why adding small SRAMs would work?

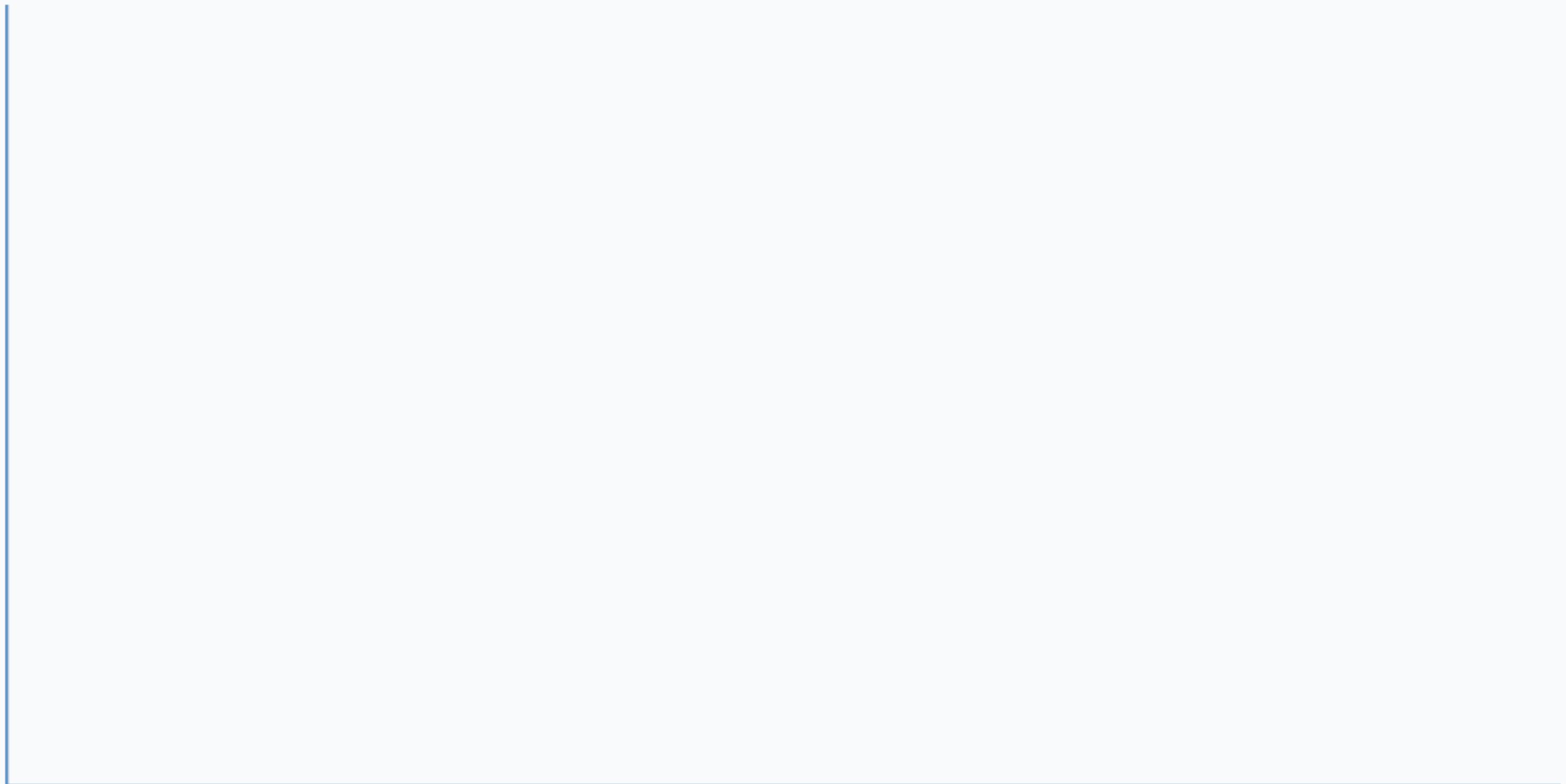
Because of localities of memory references!

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



A

B

C

D

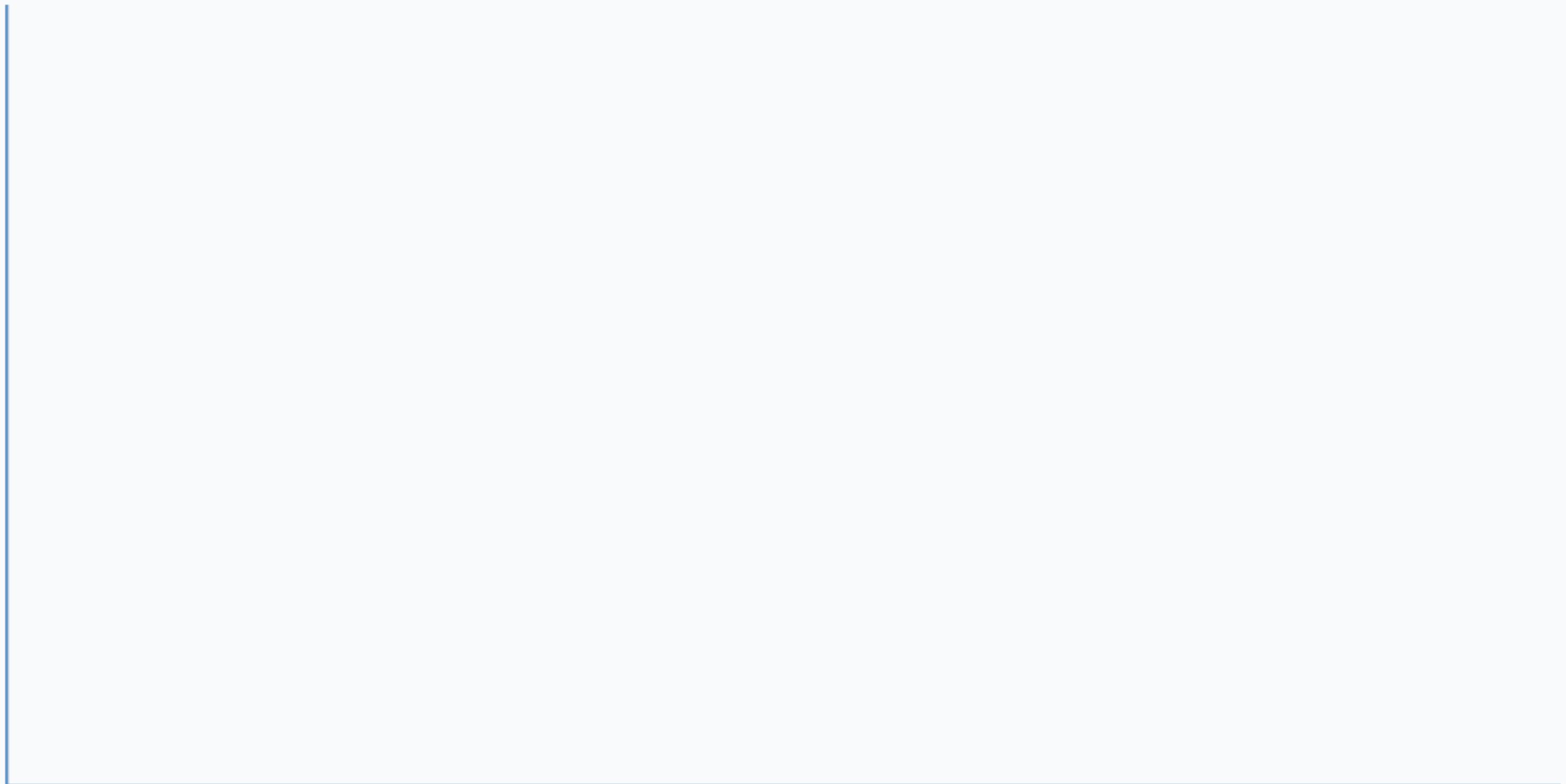
E

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



A

B

C

D

E

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;  
}
```

**repeat many times —
temporal locality!**

```
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**

Locality

- Spatial locality — application tends to visit nearby stuffs in the memory
 - Code — the current instruction, and then the next

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

- Data — the current element in an array, then the next and again
- Temporal locality — application revisit the same thing again and again
- Code — loops, frequently visiting the same code
- Data — the same data can be read/write many times

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



Start the presentation to see live content. Still no live content? Install the app or get help at PollEv.com/app

Announcement

- Reading quiz due next Tuesday BEFORE the lecture
 - We will drop two of your least performing reading quizzes
 - You have two shots, both unlimited time
- Assignment #1 due 4/20
 - We typically give you two weeks to work on an assignment
 - We never allow late submission and we will never have deadline extension
- Fun fact
 - 5538 tasks submitted to our cluster last two weeks
 - 73 tasks from each of you on average
- Check our website for slides, eLearn for quizzes, piazza for discussions and submit your assignments on Gradescope
- Youtube channel for lecture recordings:
<https://www.youtube.com/c/ProfUsagi/playlists>

Computer Science & Engineering

203

つづく

