

GPU Programming 2019/2020

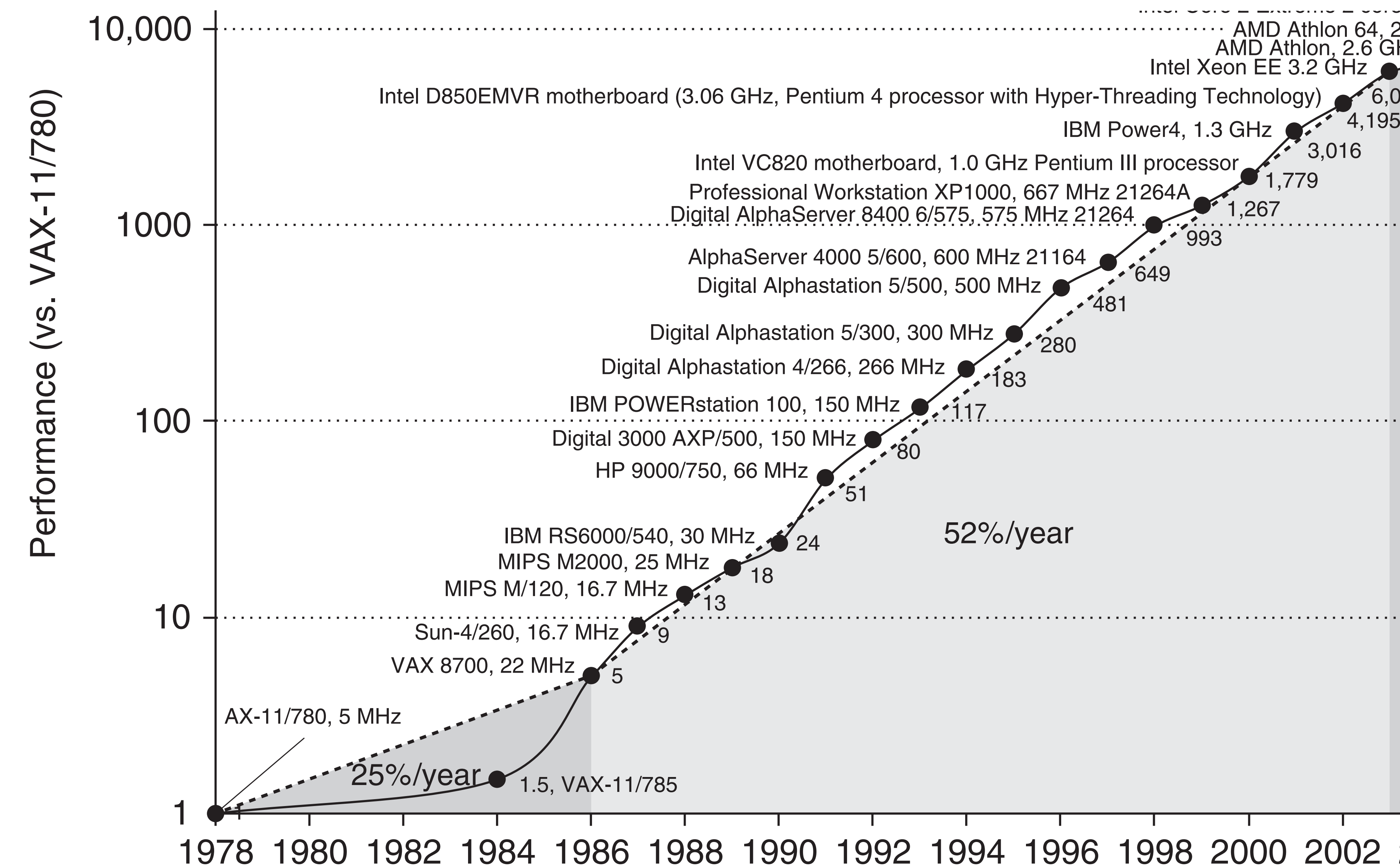
Why parallelism?

Christian Lessig

# Why parallelism?

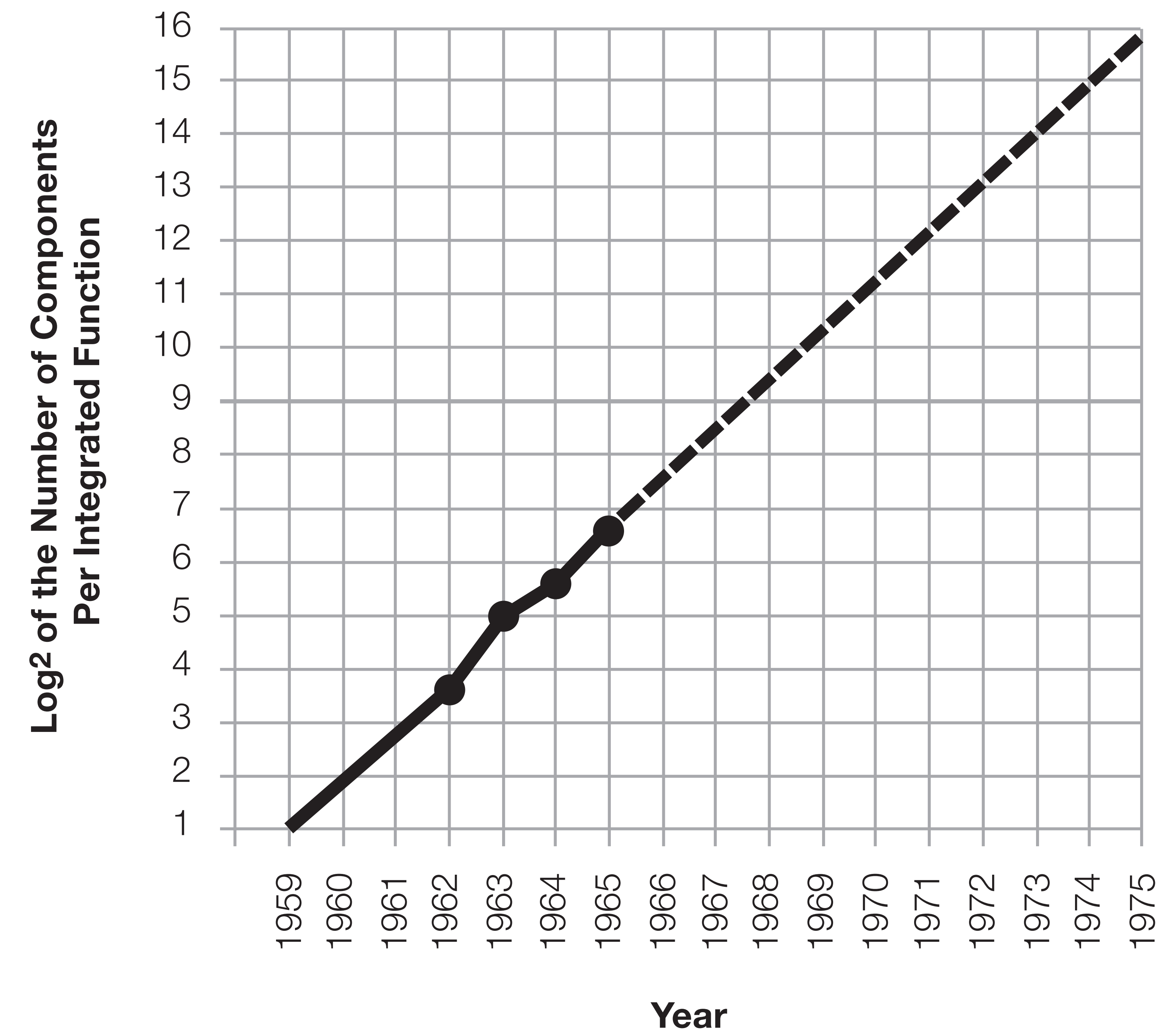
## More efficient programs!

# Why parallel?



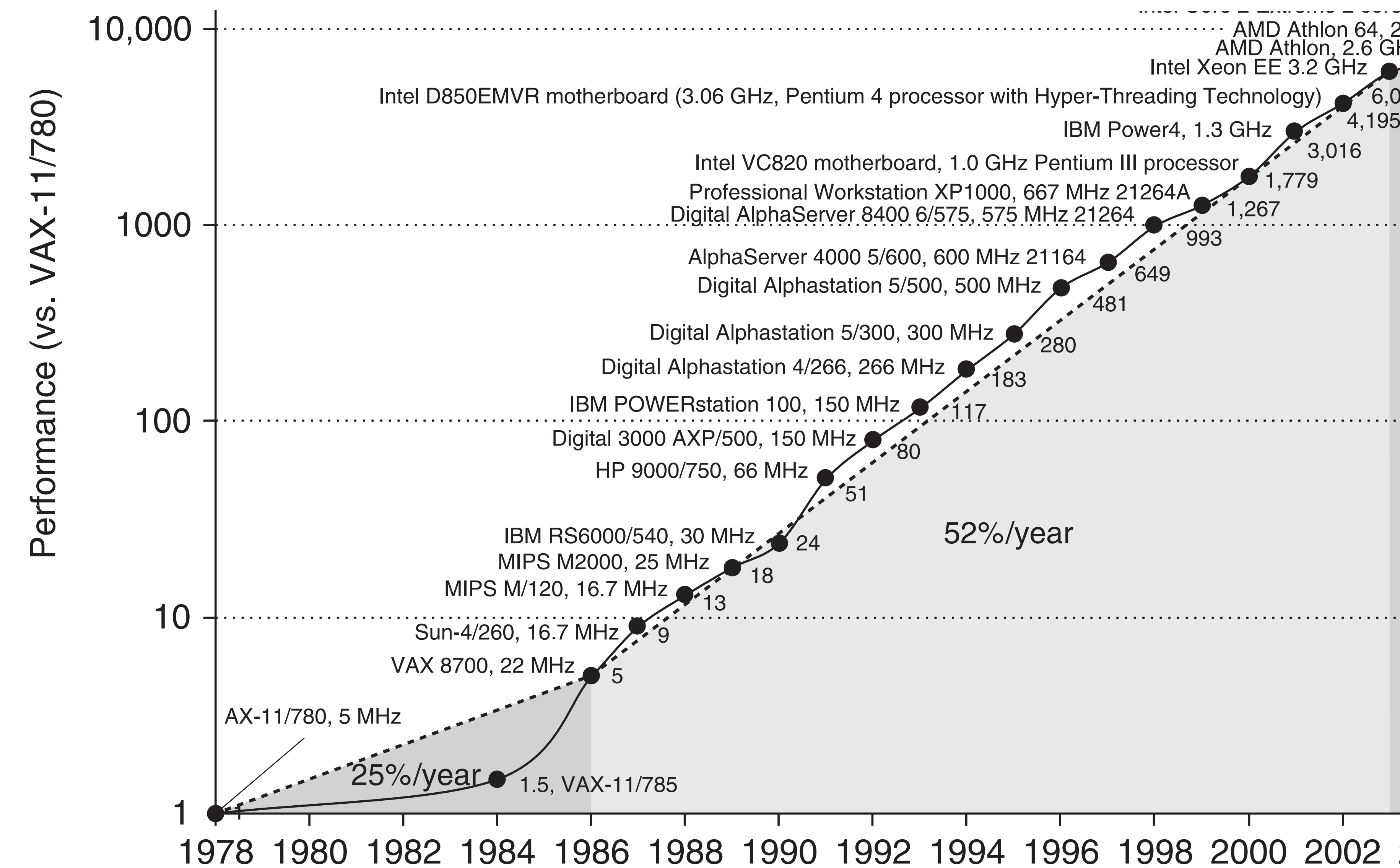
[1] J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, sixth edition. Morgan Kaufmann, 2017.

# Why parallel?



G. Moore, "Cramming more components onto integrated circuits,"  
Electronics, vol. 38, no. 8, p. 114 ff, 1965.

# Why parallel?



[1] J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, sixth edition. Morgan Kaufmann, 2017.

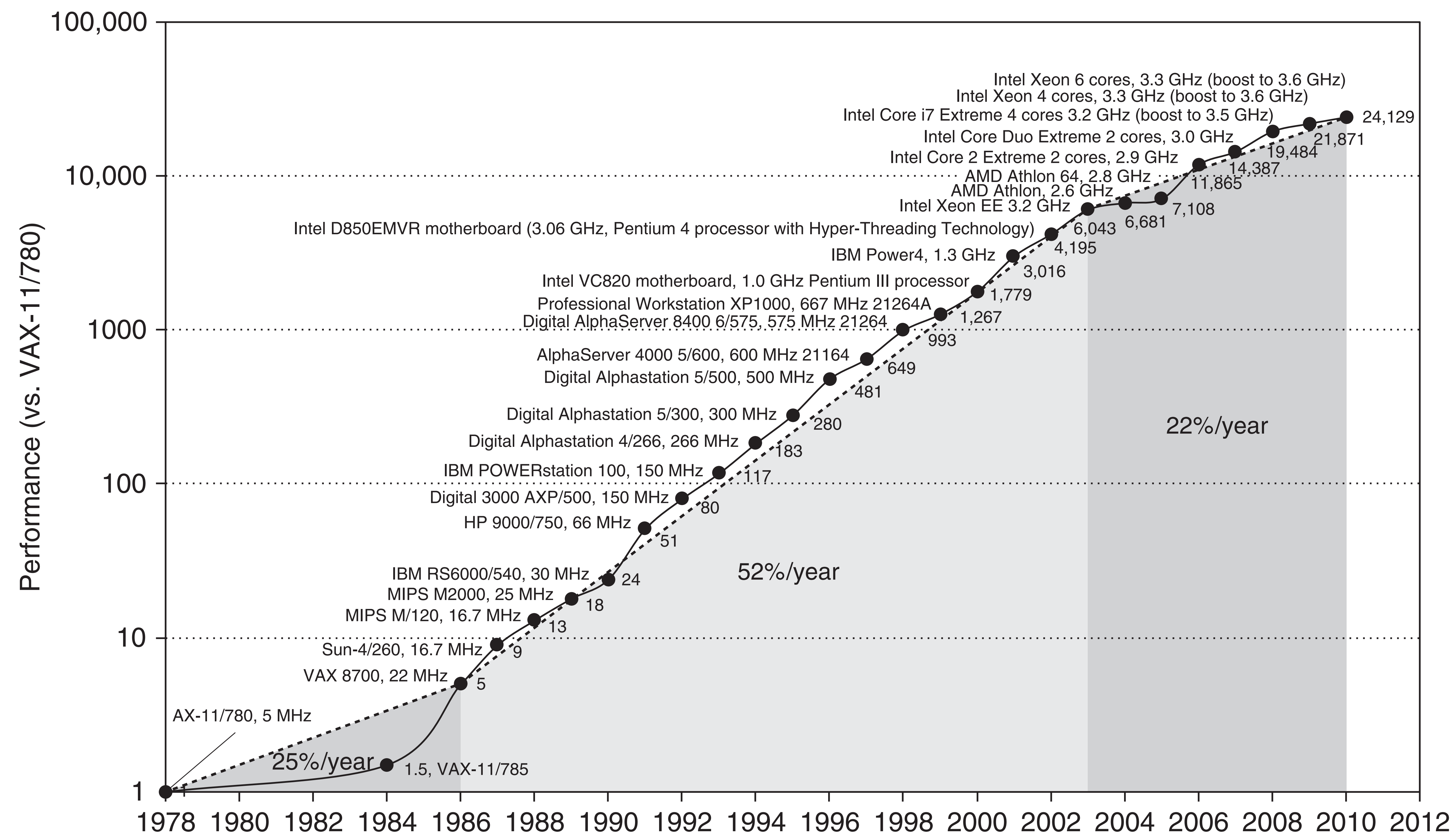
# Why parallel?

“The La-Z-Boy programmer era of relying on hardware designers to make their programs go faster without lifting a finger is officially over.”

Hennessey & Patterson [2017]

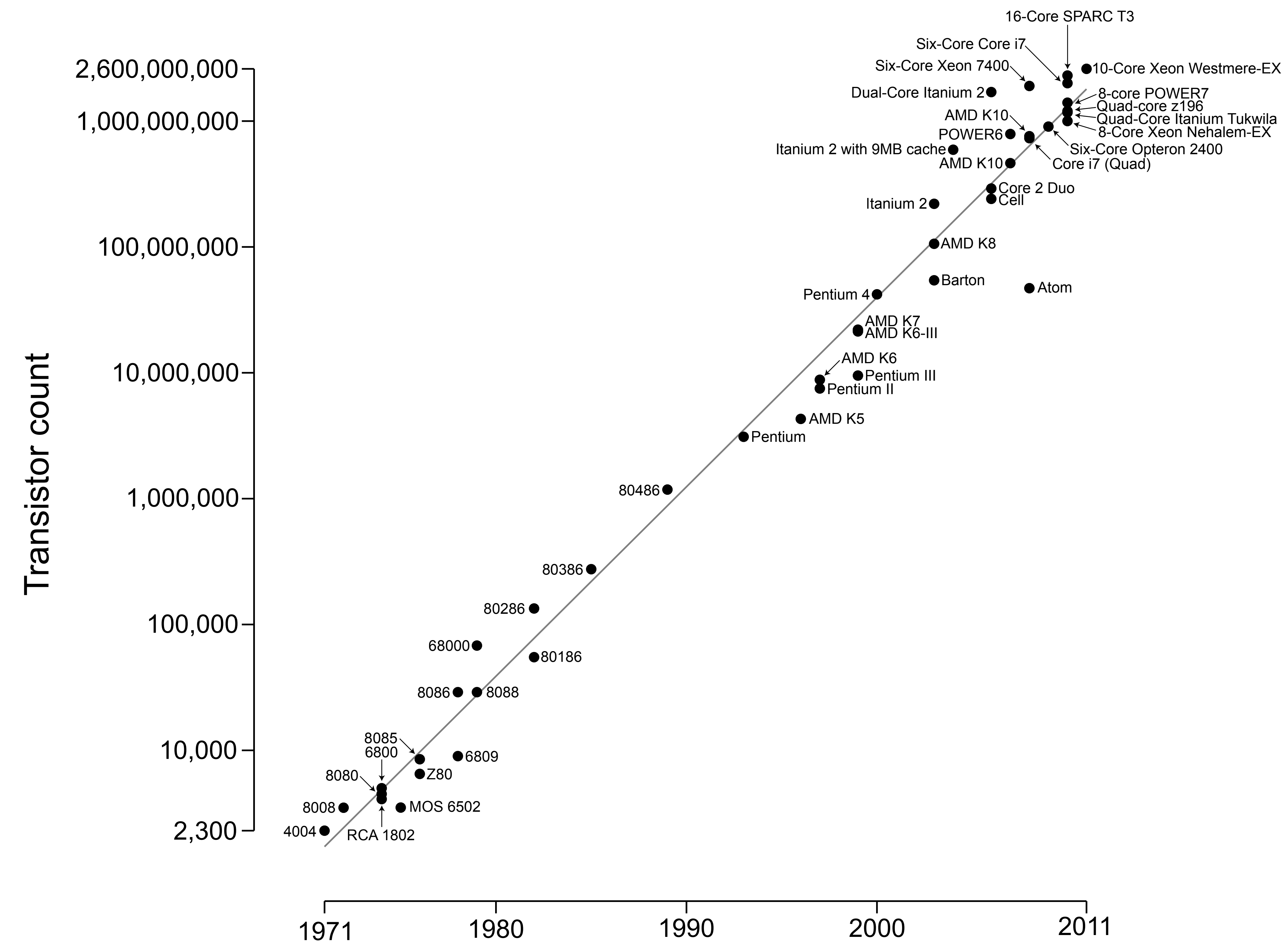


# Why parallel?



[1] J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, sixth edition. Morgan Kaufmann, 2014.

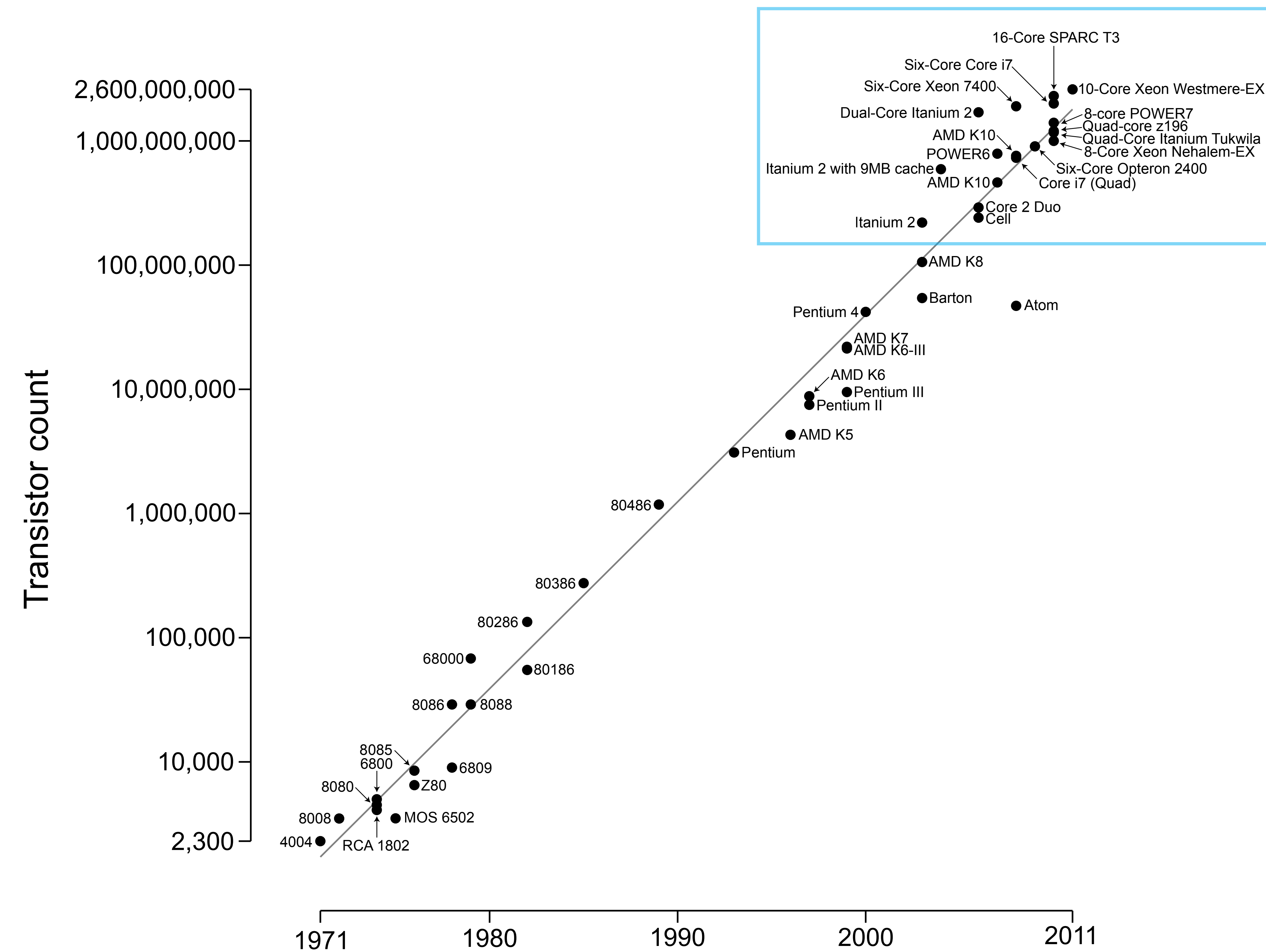
# Why parallel?



[https://upload.wikimedia.org/wikipedia/commons/0/00/Transistor\\_Count\\_and\\_Moore%27s\\_Law\\_-\\_2011.svg](https://upload.wikimedia.org/wikipedia/commons/0/00/Transistor_Count_and_Moore%27s_Law_-_2011.svg)

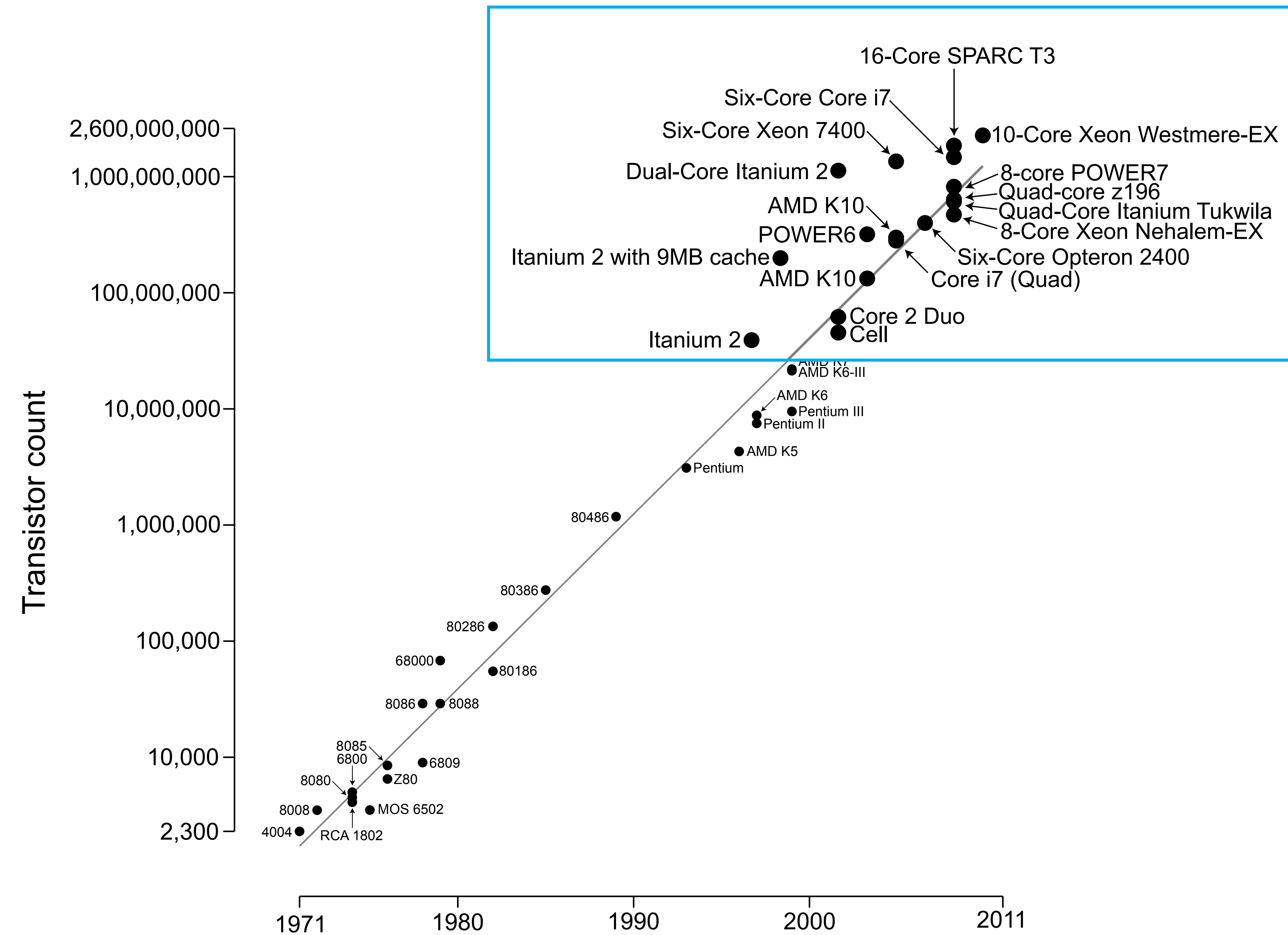


# Why parallel?



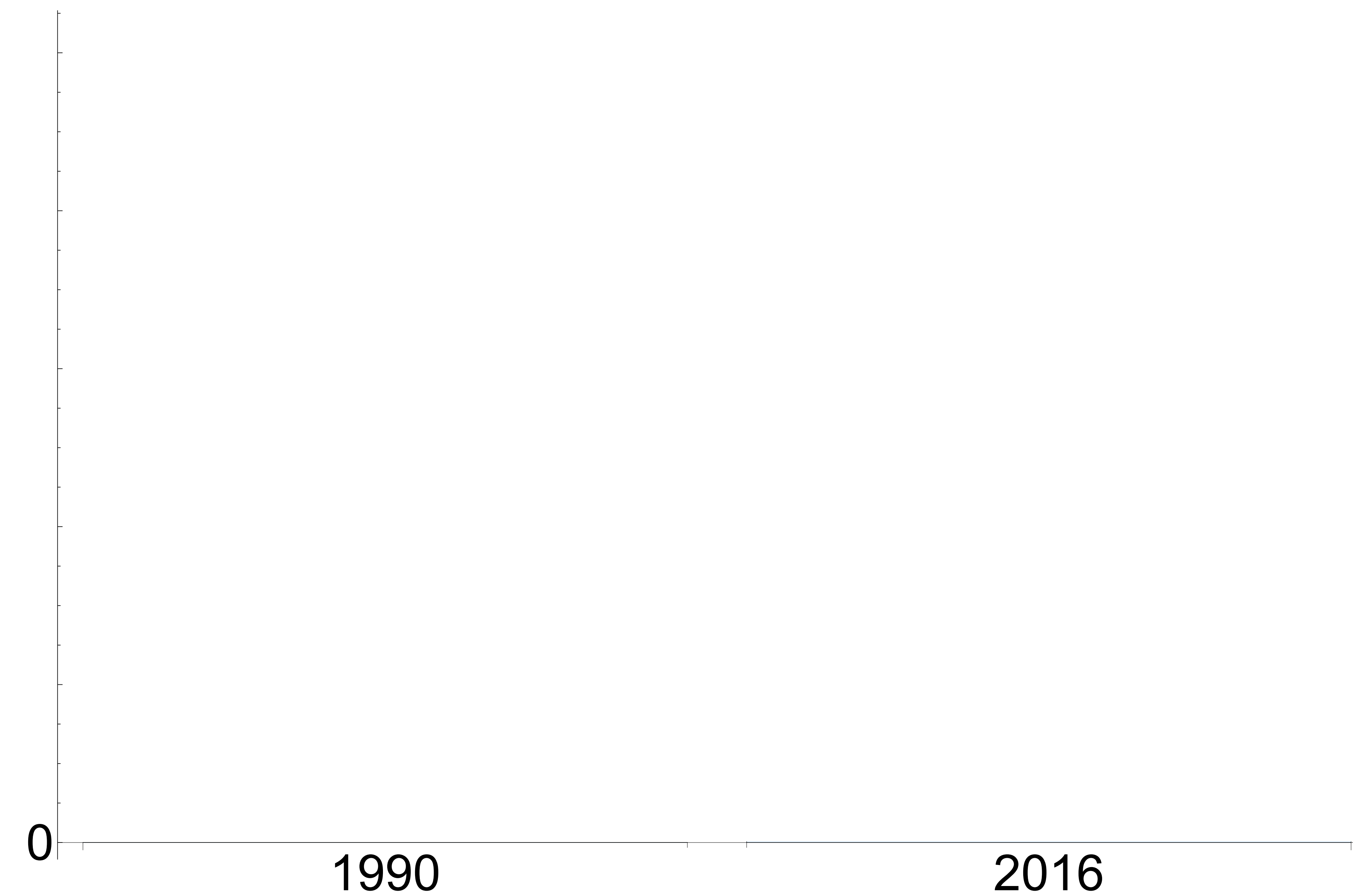
[https://upload.wikimedia.org/wikipedia/commons/0/00/Transistor\\_Count\\_and\\_Moore%27s\\_Law\\_-\\_2011.svg](https://upload.wikimedia.org/wikipedia/commons/0/00/Transistor_Count_and_Moore%27s_Law_-_2011.svg)

# Why parallel?

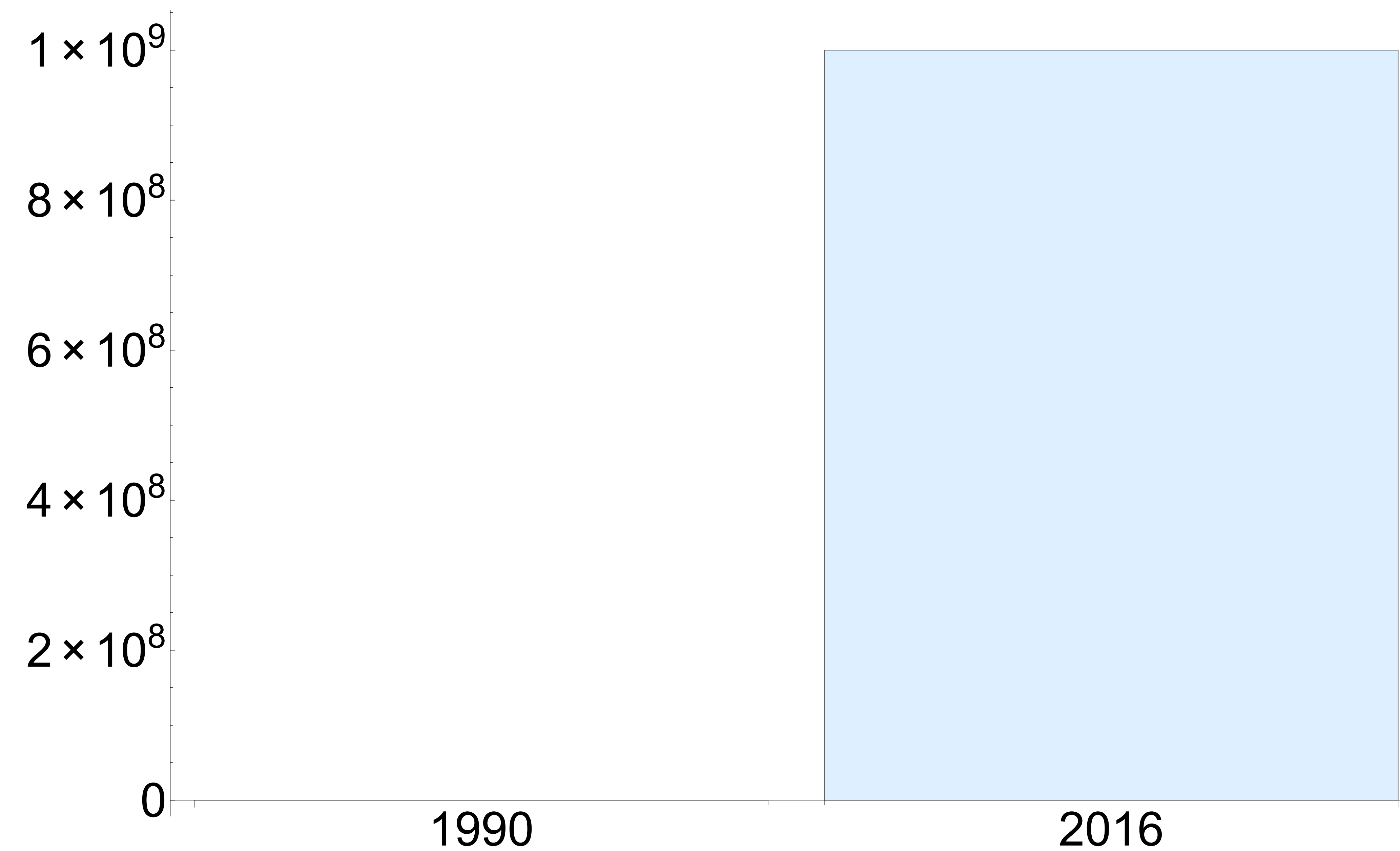


[https://upload.wikimedia.org/wikipedia/commons/0/00/Transistor\\_Count\\_and\\_Moore%27s\\_Law\\_-\\_2011.svg](https://upload.wikimedia.org/wikipedia/commons/0/00/Transistor_Count_and_Moore%27s_Law_-_2011.svg)

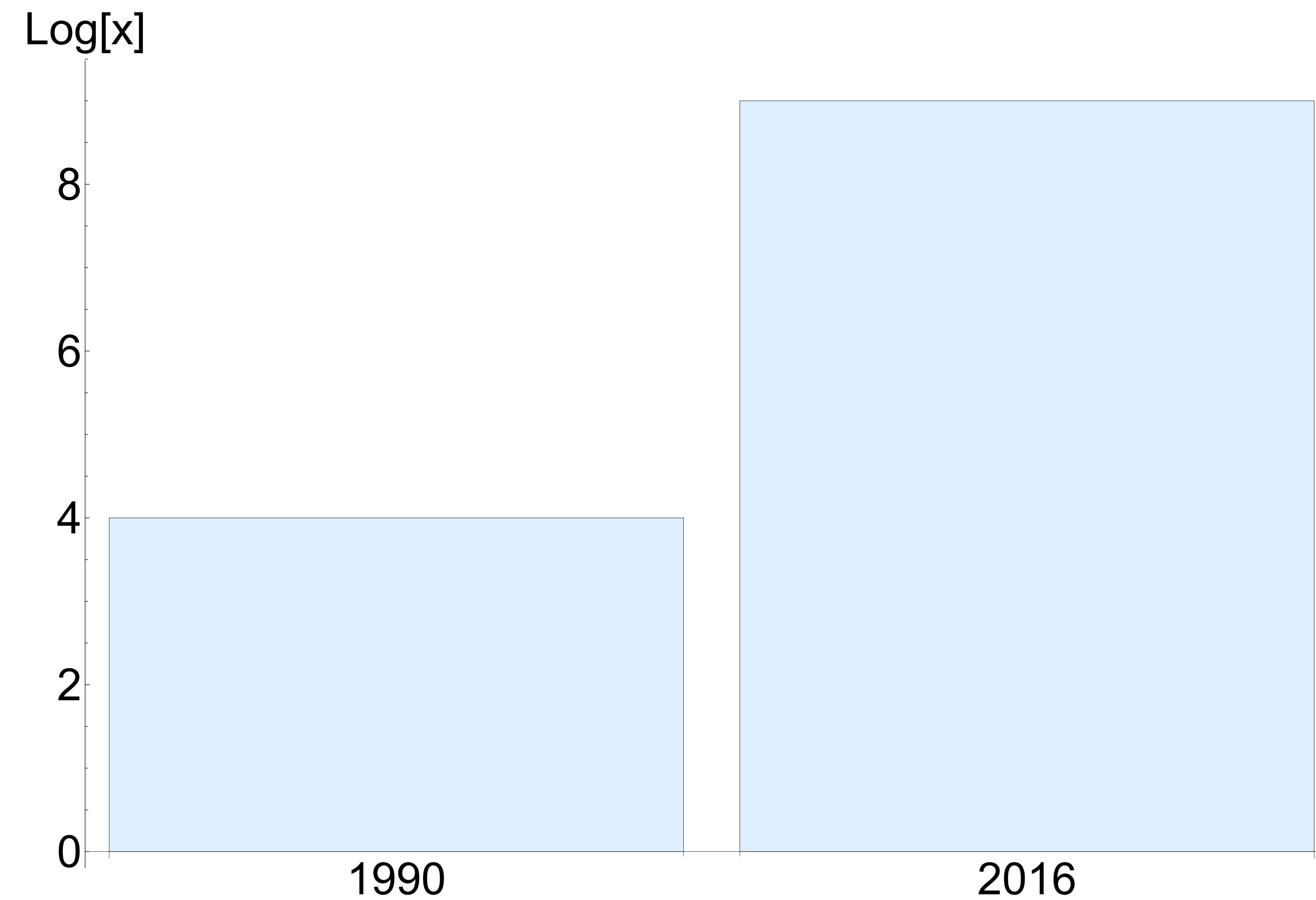
# Number of parallel computers



# Number of parallel computers

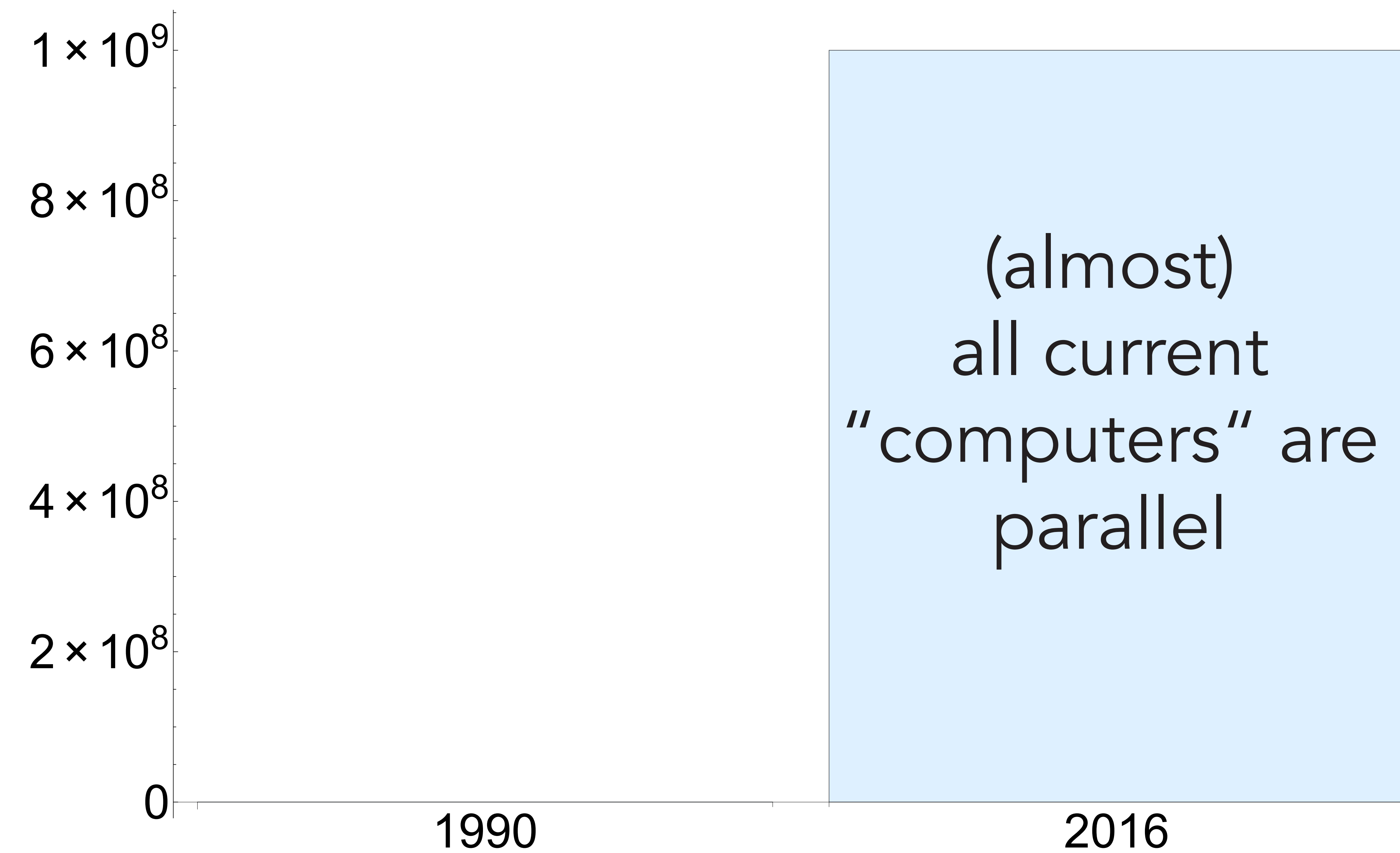


# Number of parallel computers





# Number of parallel computers



What makes parallel  
programming difficult?

# What makes parallel programming difficult?

- Many algorithms do not lend themselves naturally to a parallel implementation
- Parallel execution leads to indeterminism

# Parallel programming?

“[Serial] algorithms have improved faster than clock over the last 15 years. [Parallel] computers are unlikely to be able to take advantage of these advances because they require new programs and new algorithms.”

Gordon Bell (1992)

G. Bell, “Massively parallel computers: why not parallel computers for the masses?,” in The Fourth Symposium on the Frontiers of Massively Parallel Computation, 1992, pp. 292–297.

# Why parallel: the hardware side



# Why parallel?

$$P = C \cdot V^2 \cdot f$$

R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," IEEE J. Solid-State Circuits, vol. 32, no. 8, pp. 1210–1216, 1997.

# Why parallel?

The diagram shows the equation  $P = C \cdot V^2 \cdot f$  where each variable is enclosed in a blue box. Blue lines connect the labels to their respective variables: 'capacitance' to  $C$ , 'frequency' to  $f$ , and 'voltage' to  $V$ . The label 'power' is positioned below the  $P$  box.

$$\boxed{P} = \boxed{C} \cdot \boxed{V}^2 \cdot \boxed{f}$$

capacitance

frequency

power

voltage

R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," IEEE J. Solid-State Circuits, vol. 32, no. 8, pp. 1210–1216, 1997.

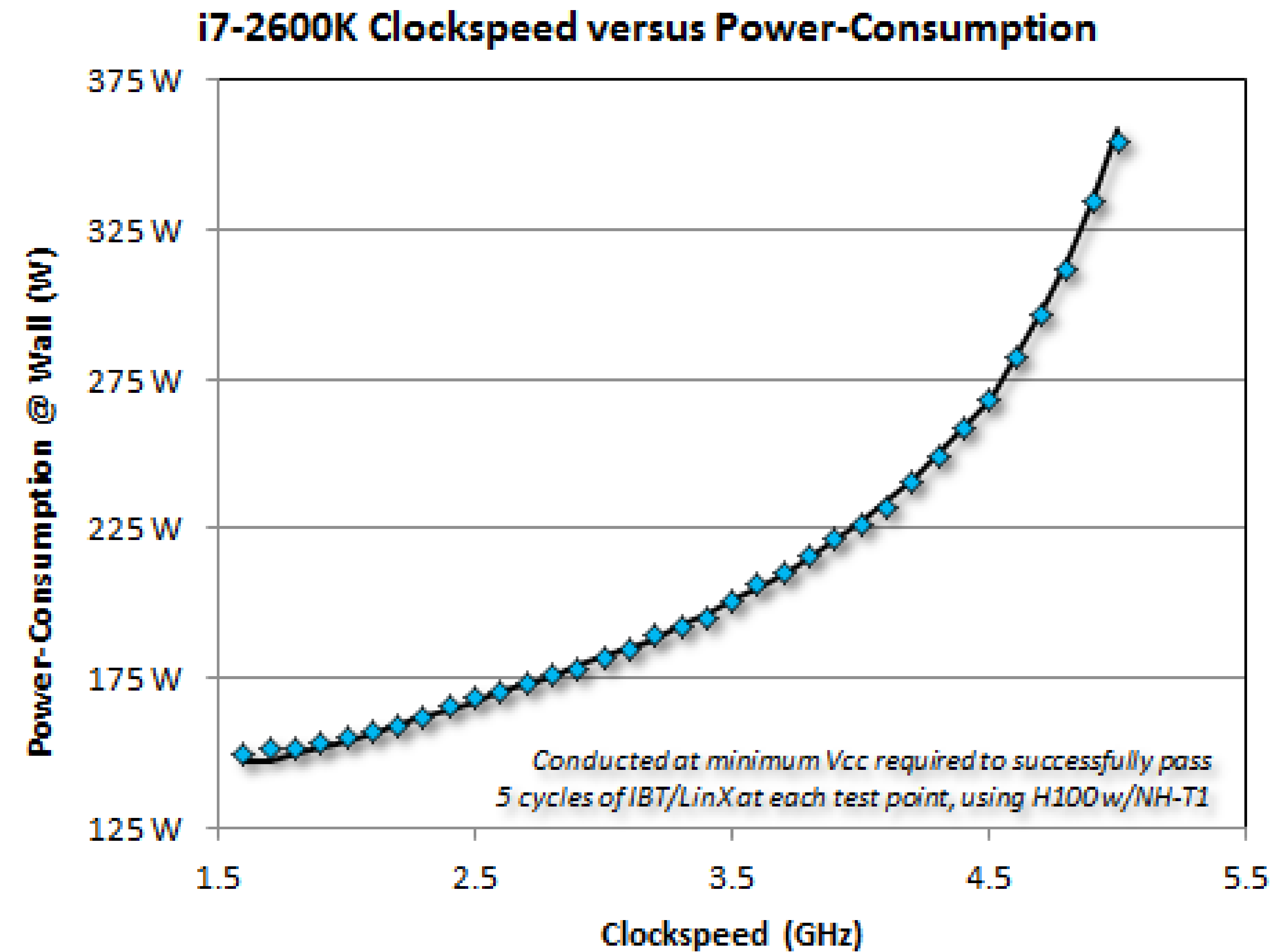
# Why parallel?

The diagram shows the equation  $P = C \cdot V^2 \cdot f$  with the following annotations:

- $P$  is labeled "power" (blue text below).
- $C$  is labeled "capacitance" (blue text above).
- $V$  is labeled "voltage" (blue text below).
- $f$  is labeled "frequency" (blue text above).
- Two red arrows point from the text "not independent" (red text) to the  $V^2$  and  $f$  terms, indicating their interdependence.

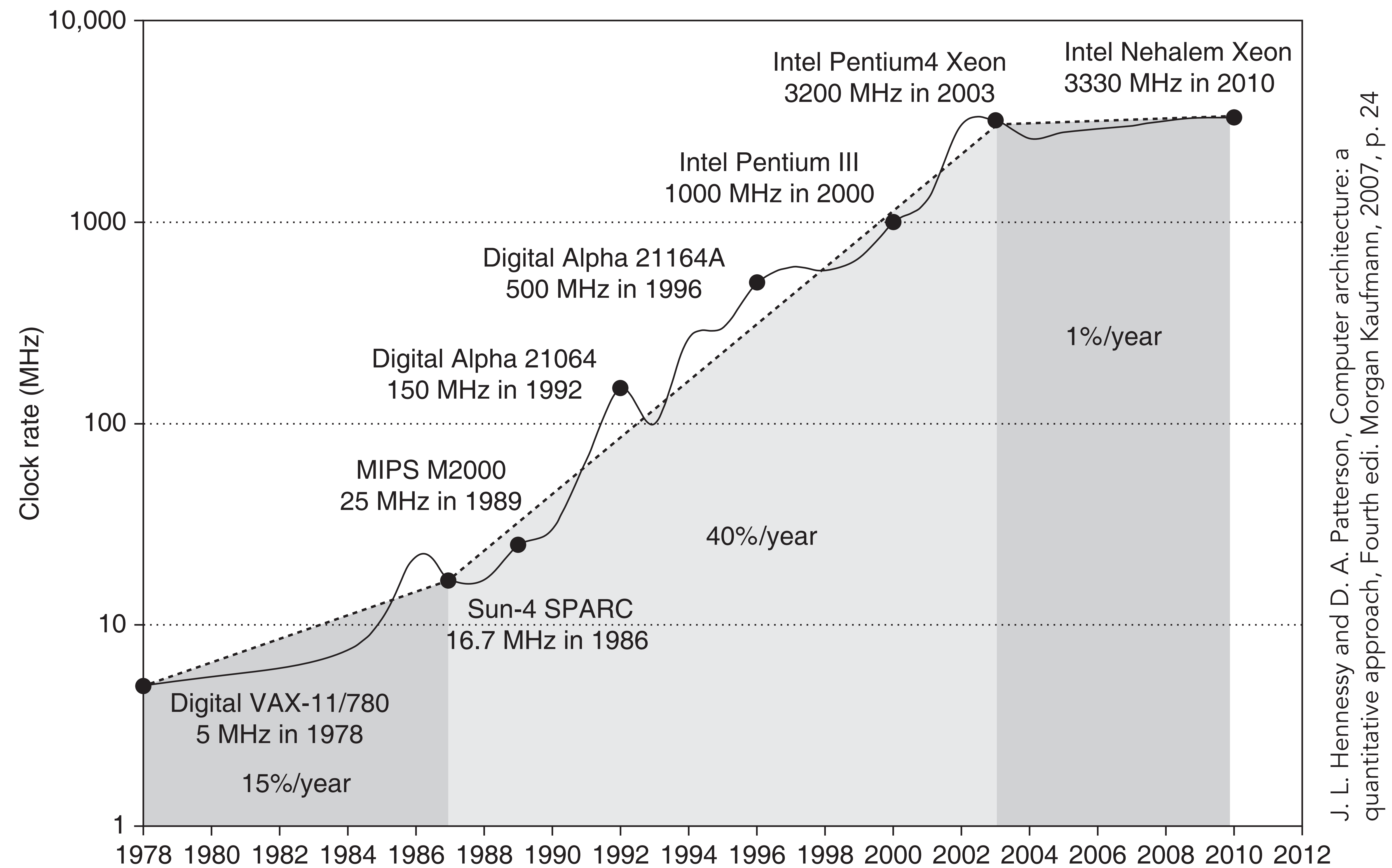
R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," IEEE J. Solid-State Circuits, vol. 32, no. 8, pp. 1210–1216, 1997.

# Why parallel?



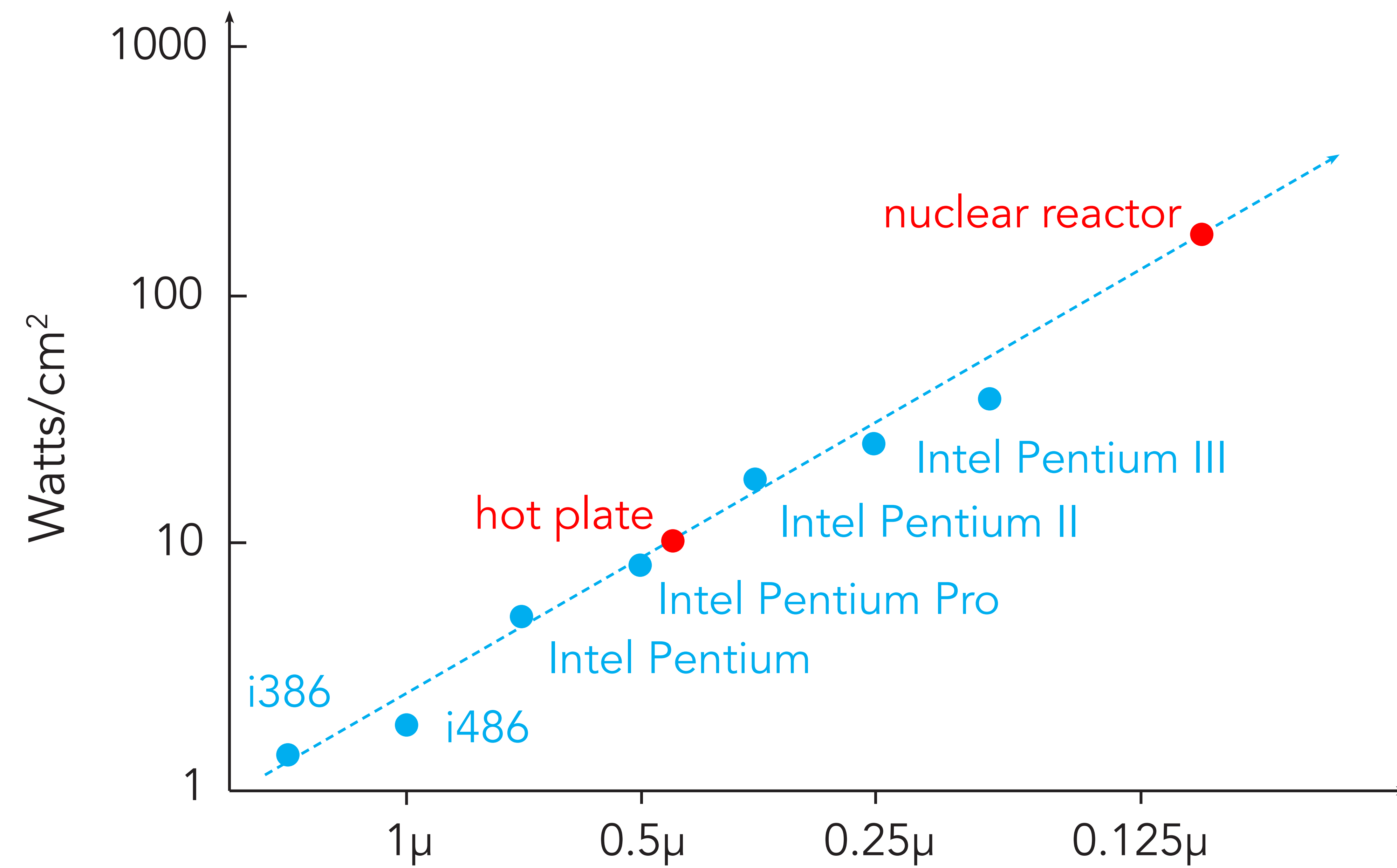
<https://forums.anandtech.com/threads/power-consumption-scaling-with-clockspeed-and-vcc-for-the-i7-2600k.2195927/>

# Why parallel?



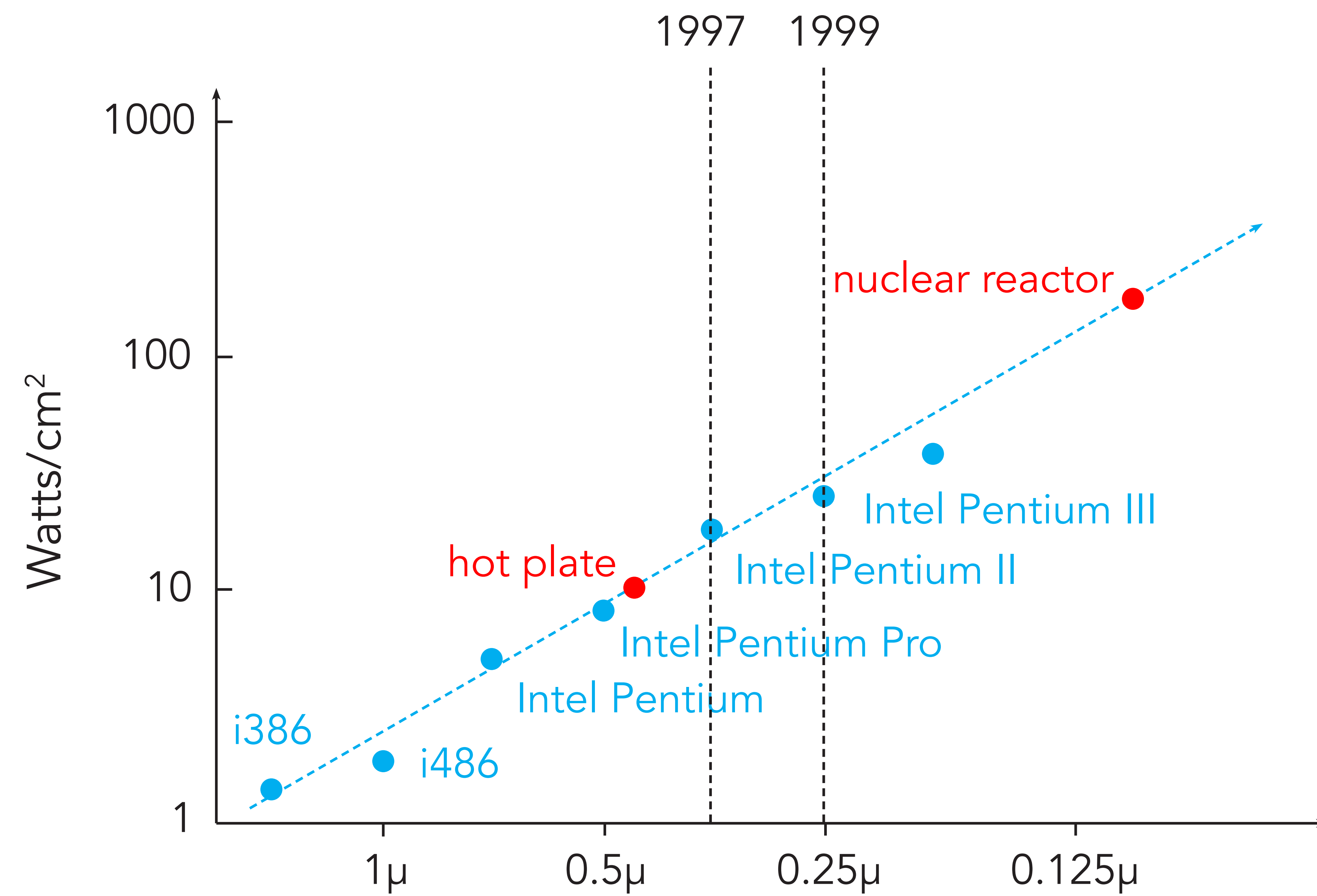


# Why parallel?



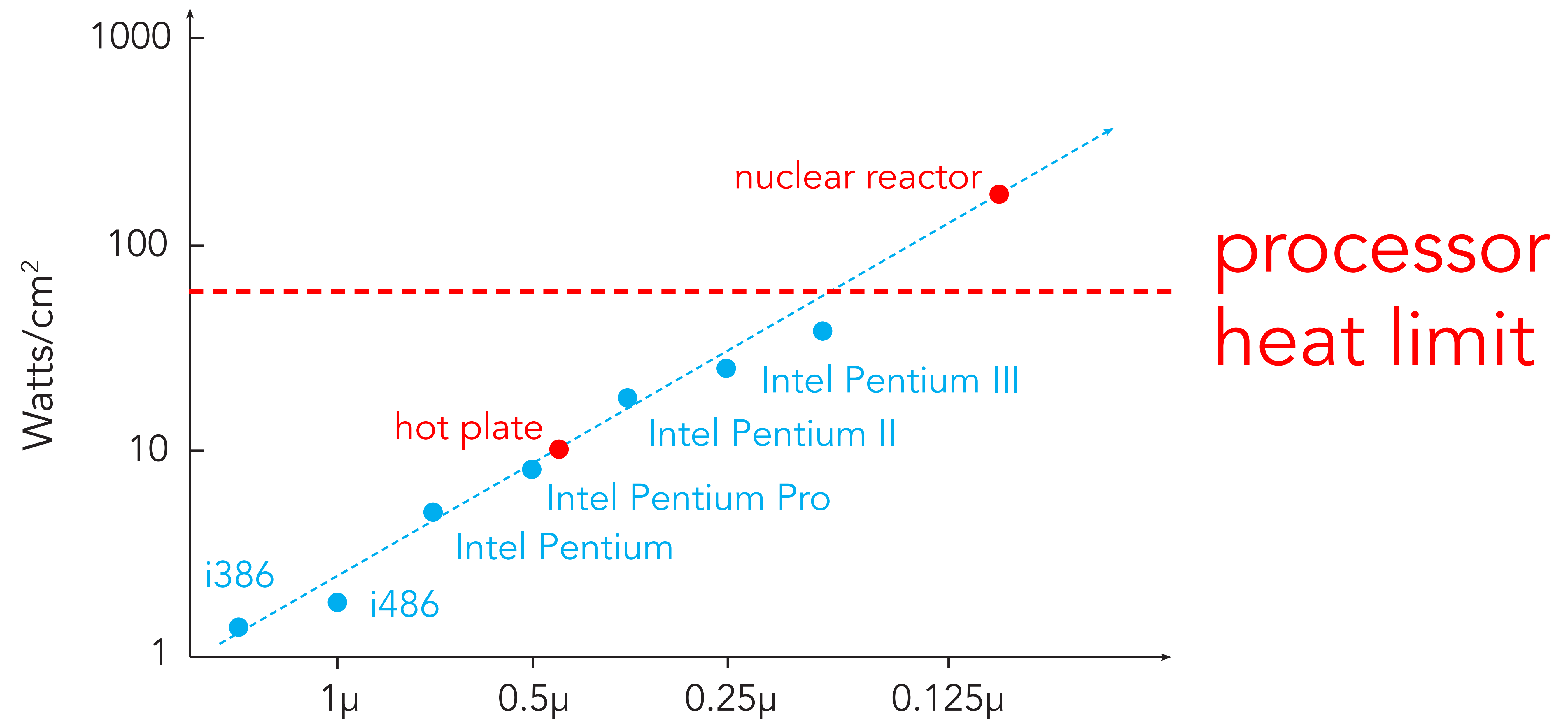
After: <http://research.ac.upc.edu/HPCseminar/SEM9900/Pollack1.pdf>

# Why parallel?



After: <http://research.ac.upc.edu/HPCseminar/SEM9900/Pollack1.pdf>

# Why parallel?



After: <http://research.ac.upc.edu/HPCseminar/SEM9900/Pollack1.pdf>

# Why parallel?

$$\boxed{E} = P \boxed{t}$$

energy                      time

# Why parallel?

Energy is critical:

- Handheld: major factor for customer satisfaction
- Warehouse scale computing: major cost factor



# Why parallel?

Energy is critical:

- Handheld: major factor for customer satisfaction
- Warehouse scale computing: major cost factor

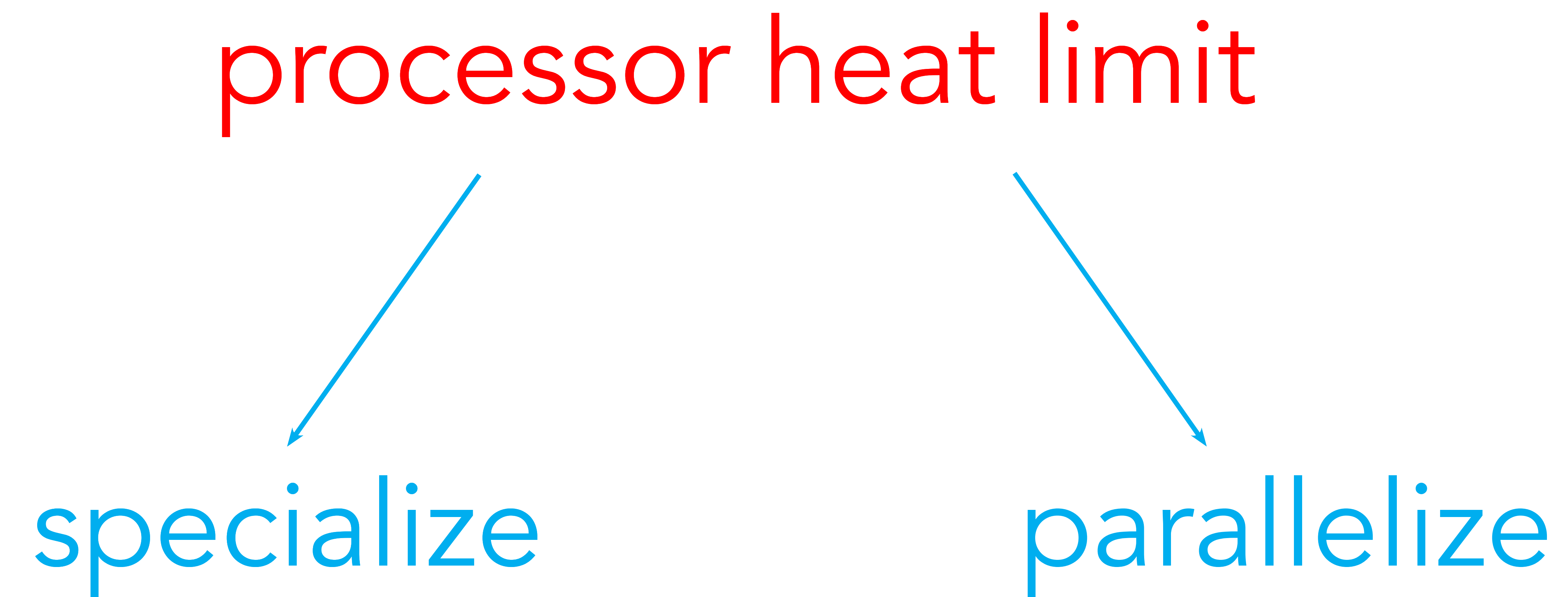
... and to keep our planet alive.

How to get around  
heat limit?

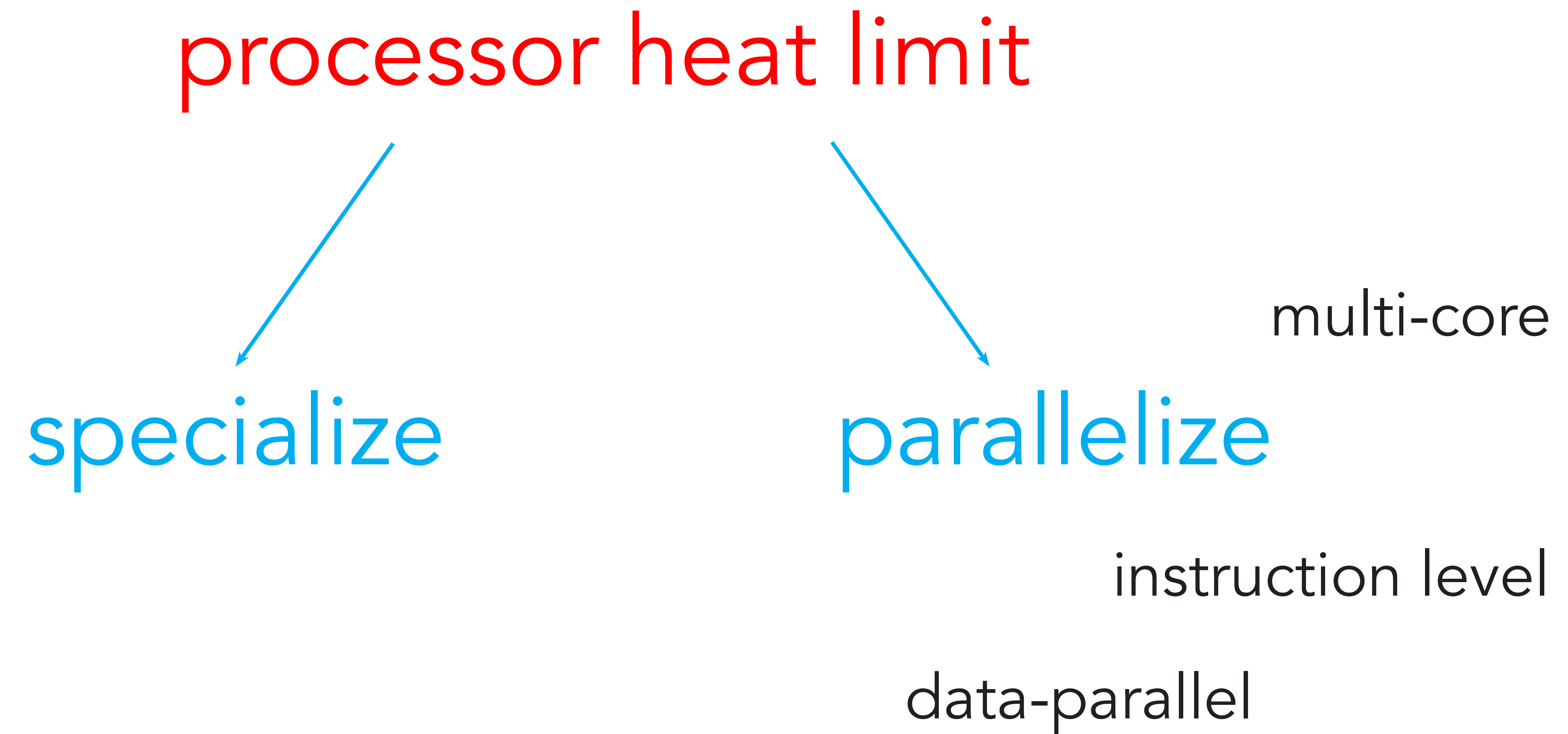
How to get around  
heat limit?

(or be as energy efficient as possible)

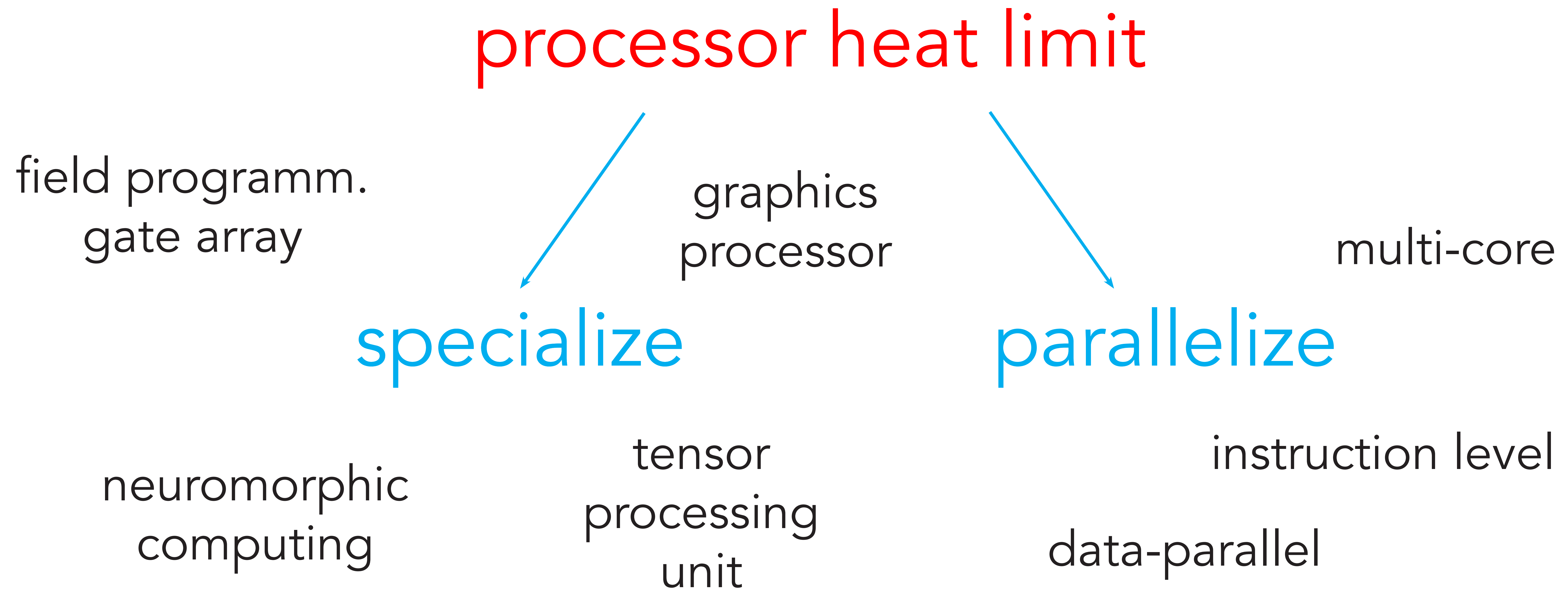
# Why parallel?



# Why parallel?

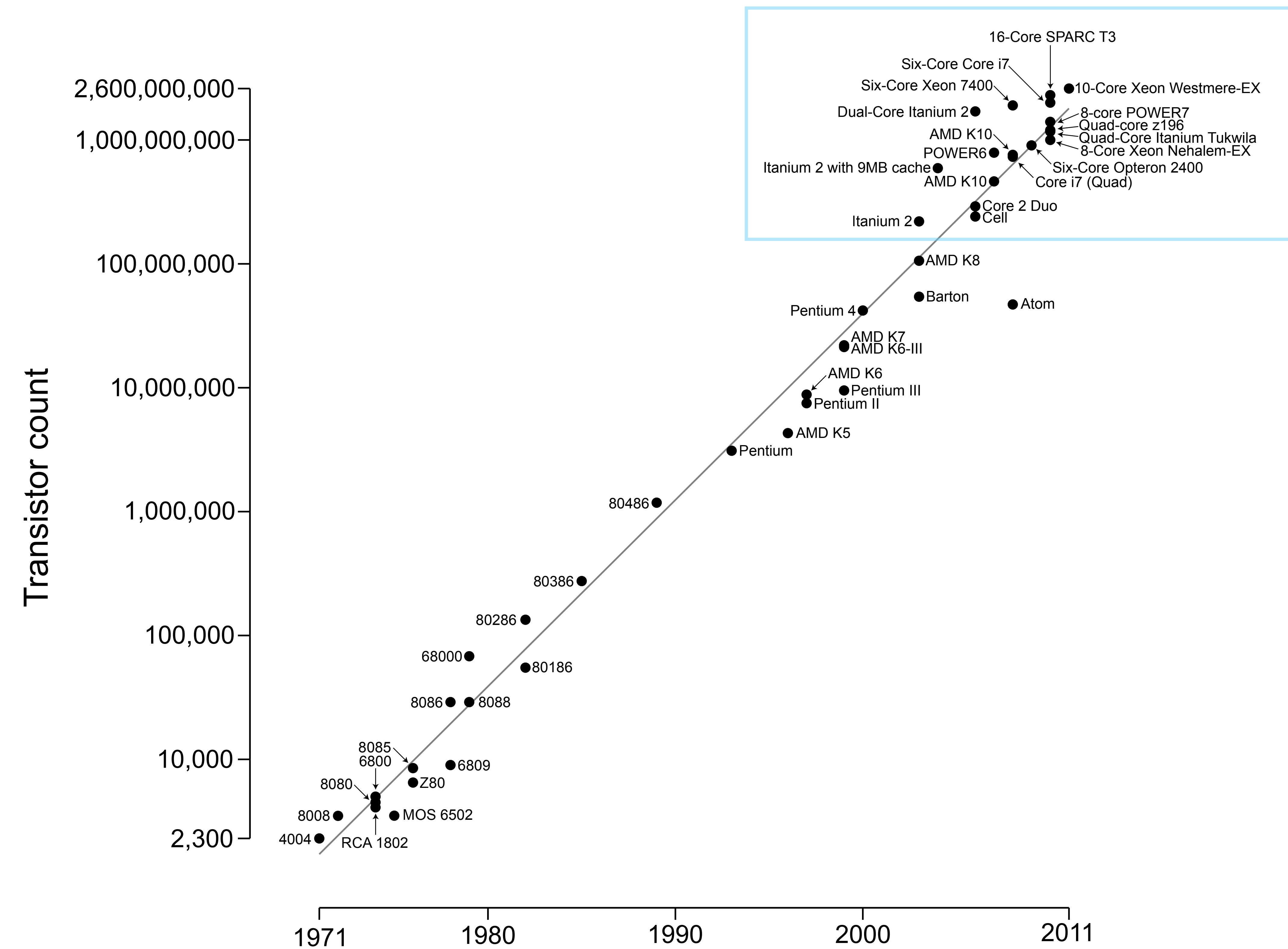


# Why parallel?





# Why parallel?



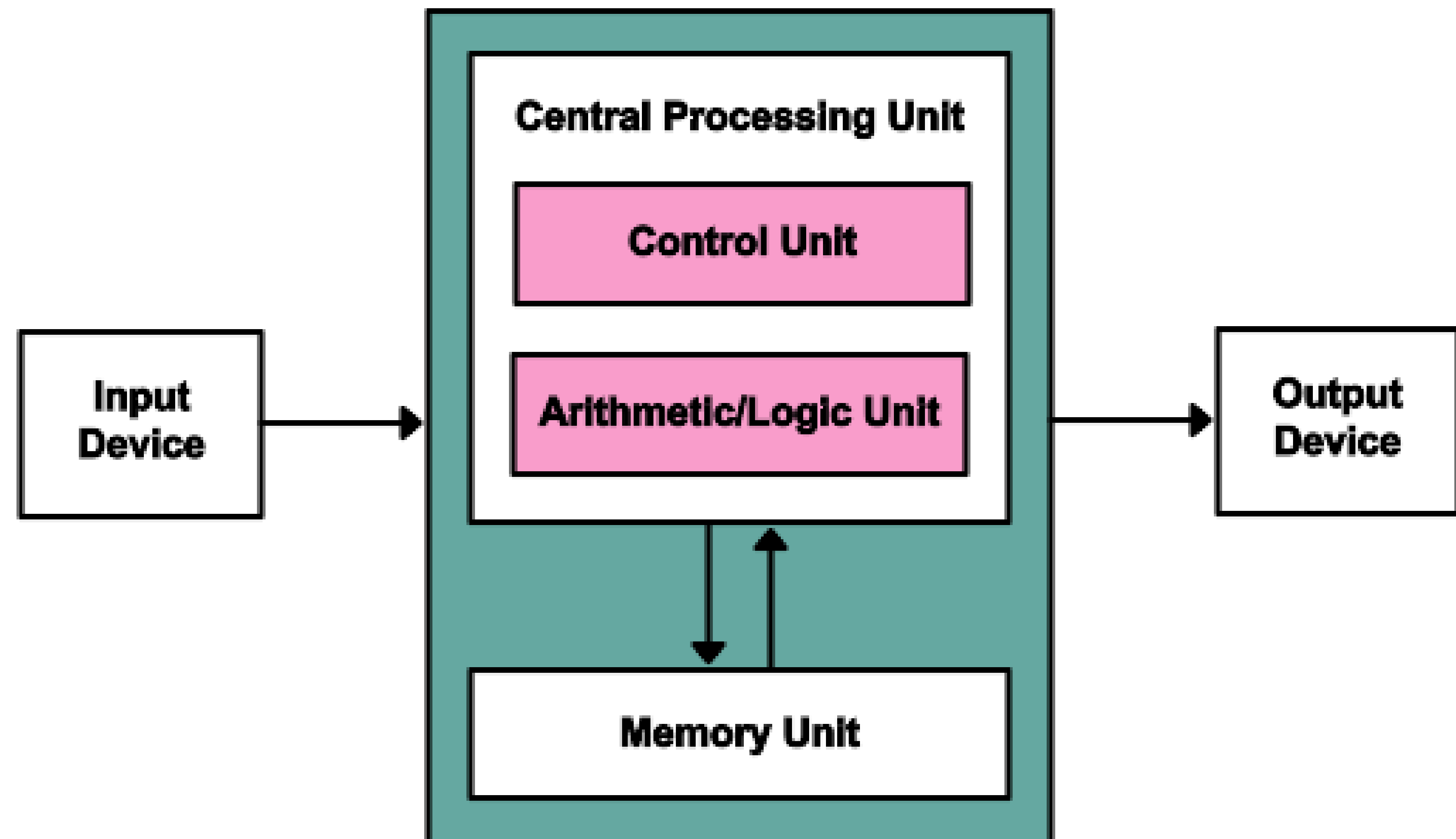
Parallelism  
to avoid heat  
limit  
(and increase  
energy  
efficiency)

# Why parallel?

	Nvidia Fermi (2010)	Nvidia Kepler (2012)
Clock frequency	1.3 GHz	1.0 GHz
Power	250 Watt	195 Watt
FP throughput	665 GFlops	1310 GFlops

<https://wiki.rice.edu/confluence/download/attachments/4435861/comp322-s16-lec1-slides.pdf>

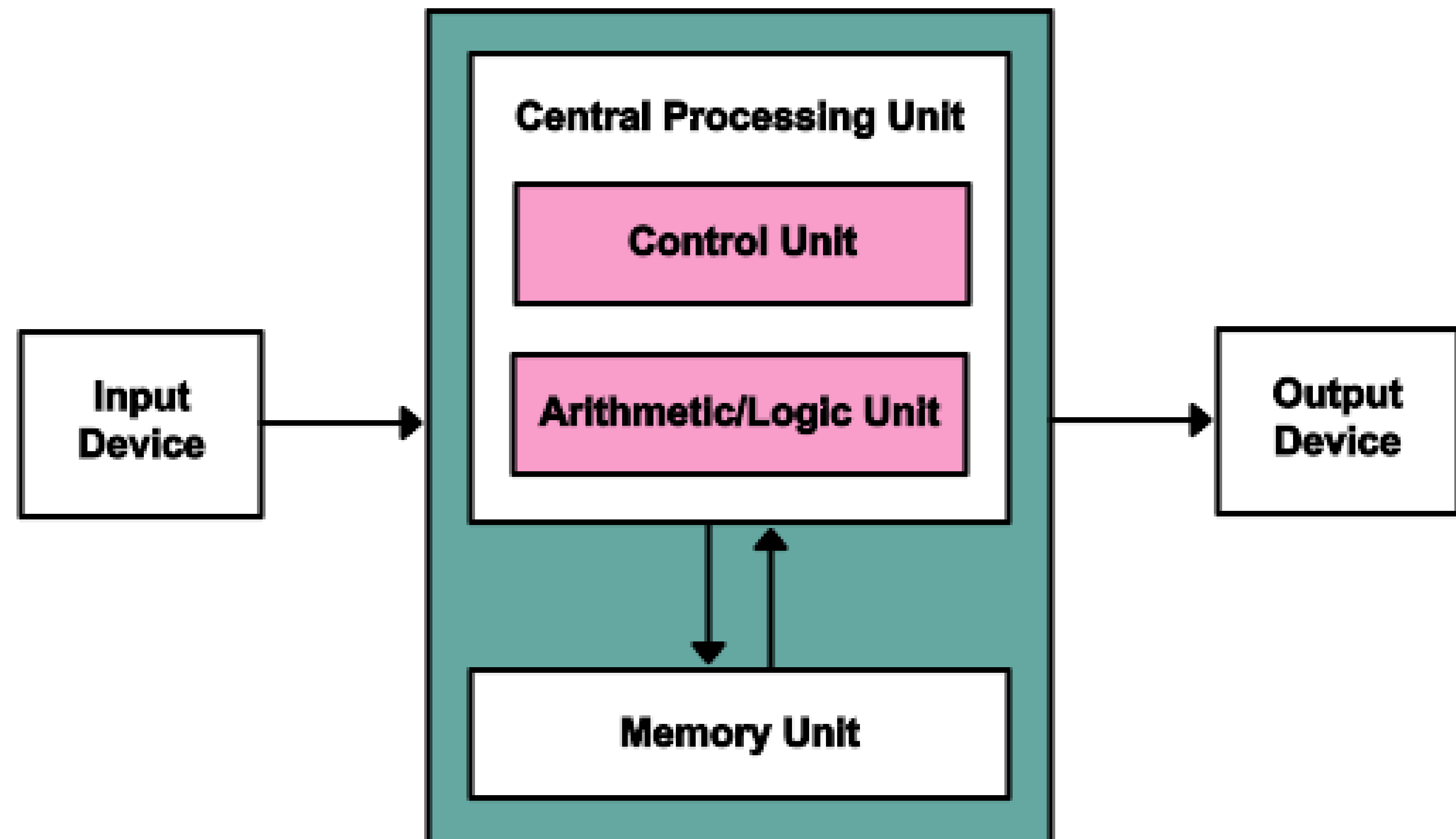
# Why parallel?



[https://en.wikipedia.org/wiki/Von\\_Neumann\\_architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture)

# Why parallel?

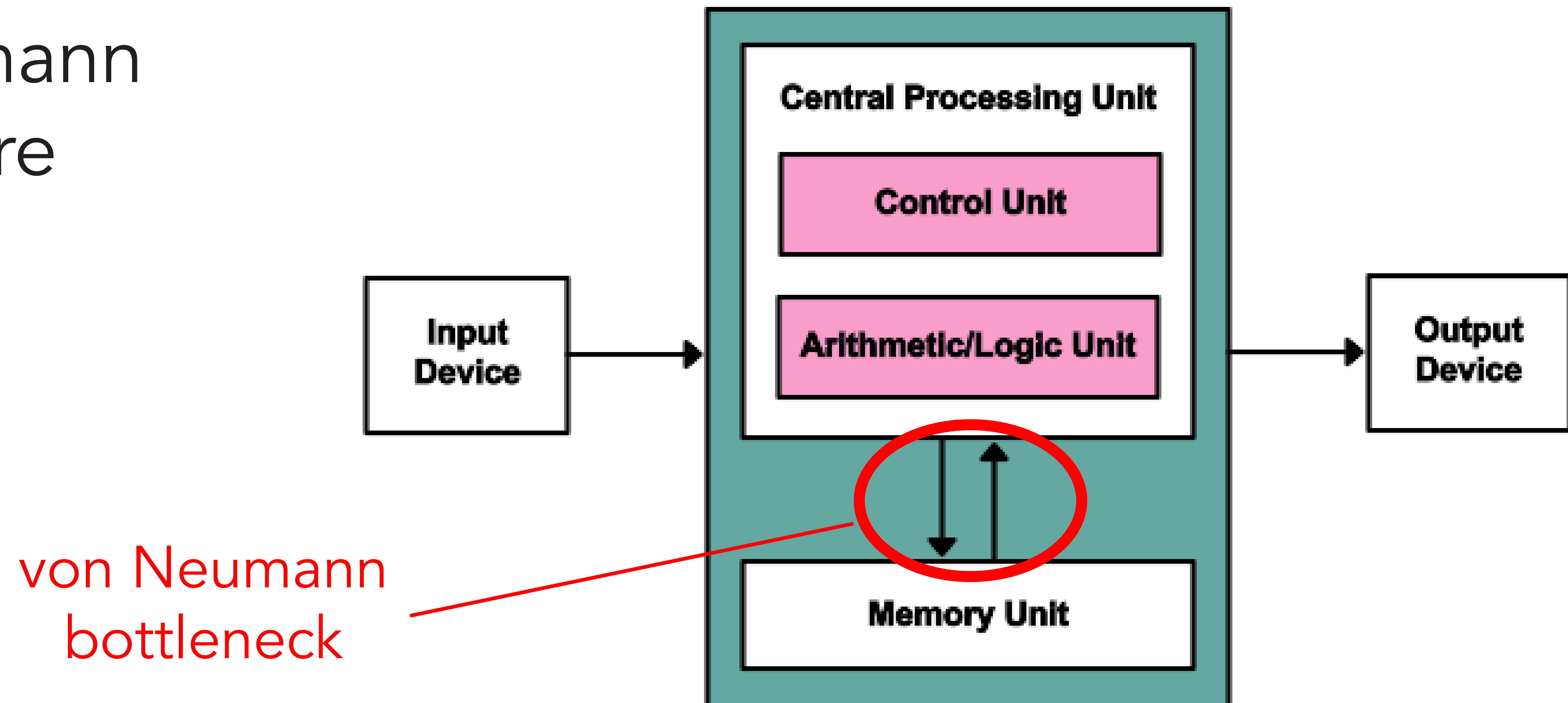
## Von Neumann architecture



[https://en.wikipedia.org/wiki/Von\\_Neumann\\_architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture)

# Why parallel?

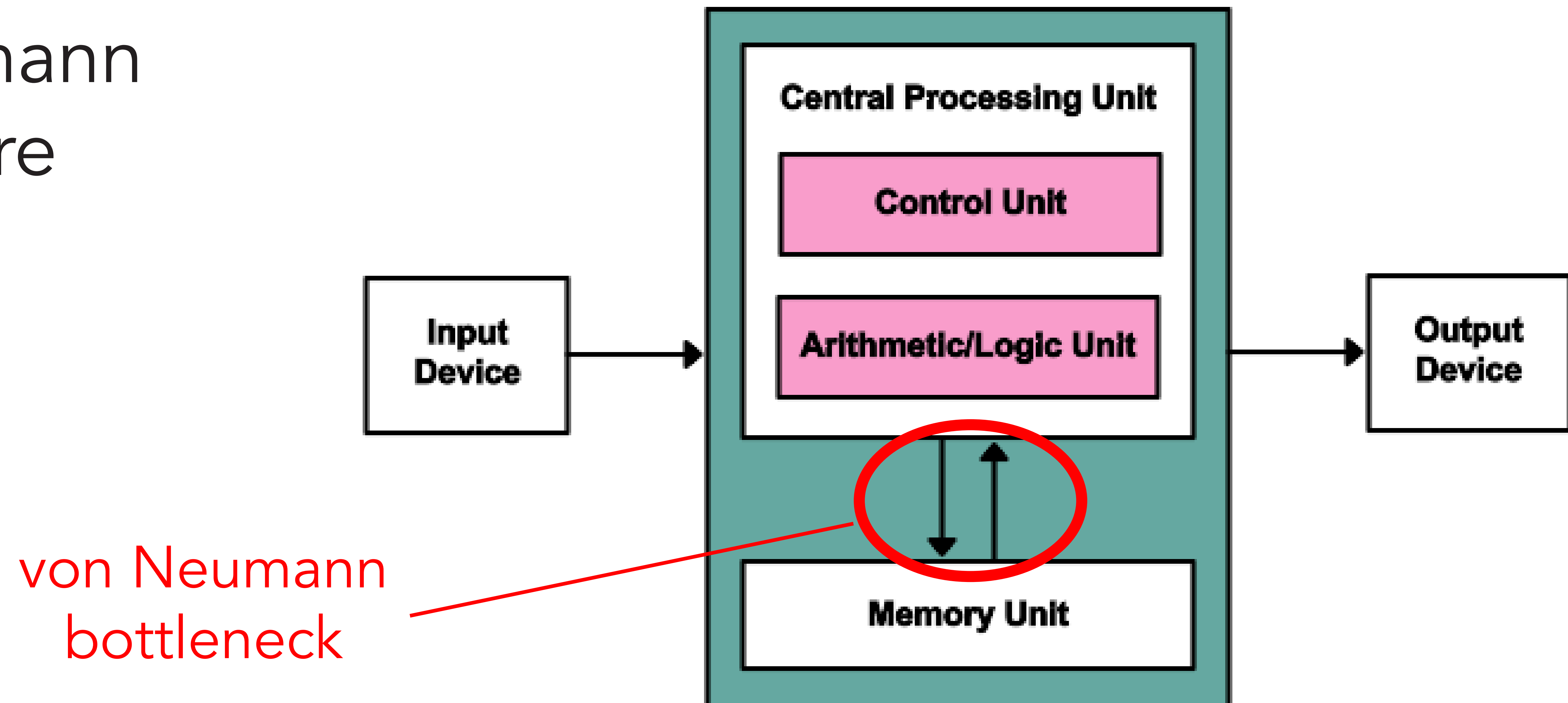
## Von Neumann architecture



[https://en.wikipedia.org/wiki/Von\\_Neumann\\_architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture)

# Why parallel?

## Von Neumann architecture



von Neumann  
bottleneck

Latency: 1990: 6 and 8 cycles  
2010: up to 180 cycles

[https://en.wikipedia.org/wiki/Von\\_Neumann\\_architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture)



# Why parallel?

	frequency	latency	bandwidth
2000	1 GHz	20 ns	100 MT/s

Data: [https://en.wikipedia.org/wiki/CAS\\_latency](https://en.wikipedia.org/wiki/CAS_latency), <http://www.intel.com/pressroom/kits/quickreffam.htm>

# Why parallel?

	frequency	latency	bandwidth
2000	1 GHz	20 ns	100 MT/s
2003	2 GHz	15 ns	333 MT/s

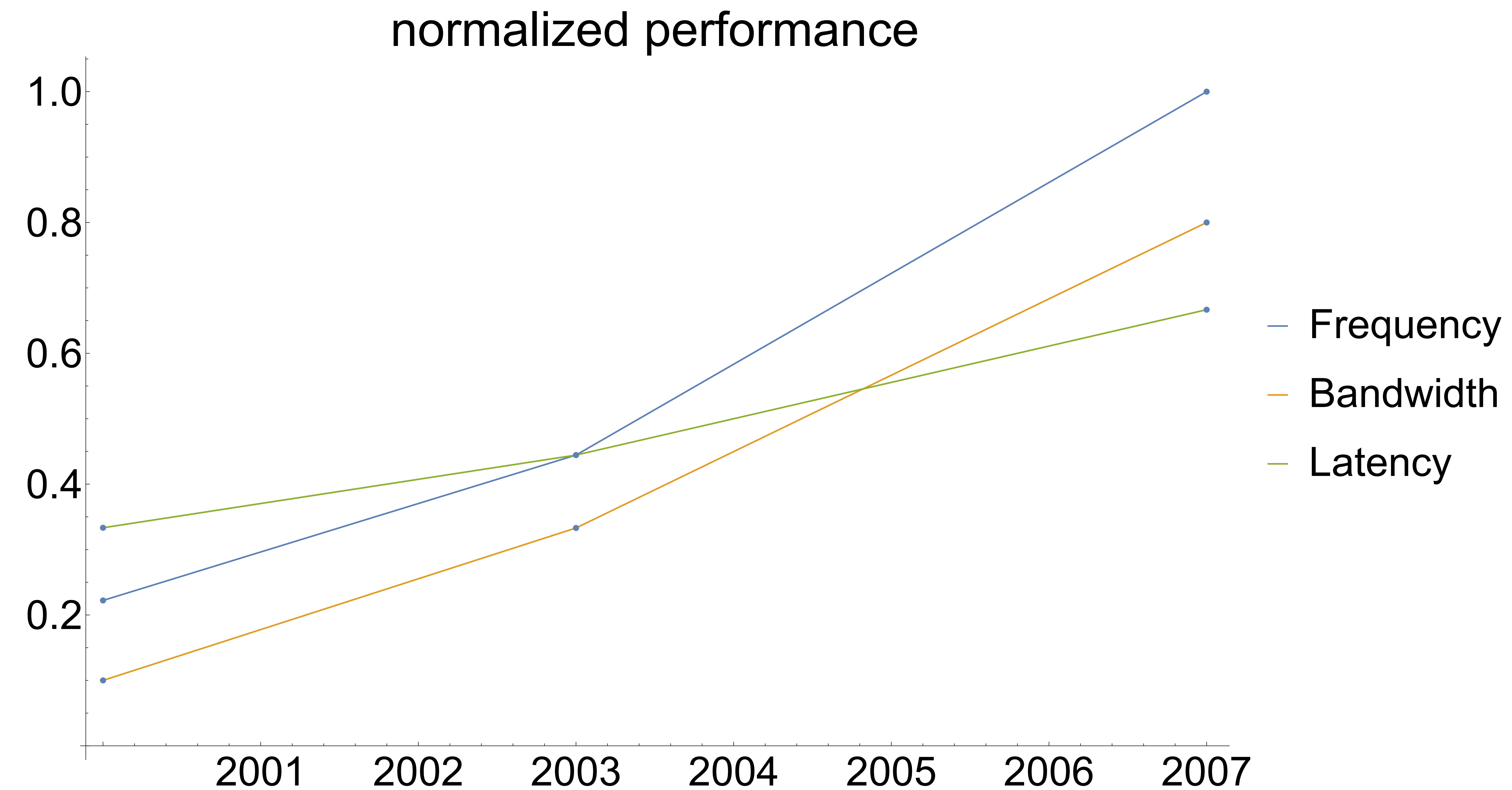
Data: [https://en.wikipedia.org/wiki/CAS\\_latency](https://en.wikipedia.org/wiki/CAS_latency), <http://www.intel.com/pressroom/kits/quickreffam.htm>

# Why parallel?

	frequency	latency	bandwidth
2000	1 GHz	20 ns	100 MT/s
2003	2 GHz	15 ns	333 MT/s
2007	4.5 GHz	10 ns	800 MT/s

Data: [https://en.wikipedia.org/wiki/CAS\\_latency](https://en.wikipedia.org/wiki/CAS_latency), <http://www.intel.com/pressroom/kits/quickreffam.htm>

# Why parallel?



Data: [https://en.wikipedia.org/wiki/CAS\\_latency](https://en.wikipedia.org/wiki/CAS_latency), <http://www.intel.com/pressroom/kits/quickreffam.htm>

Why parallel?

How to get around von  
Neumann bottleneck?

# Why parallel?

von Neumann bottleneck



caching



# Why parallel?

von Neumann bottleneck



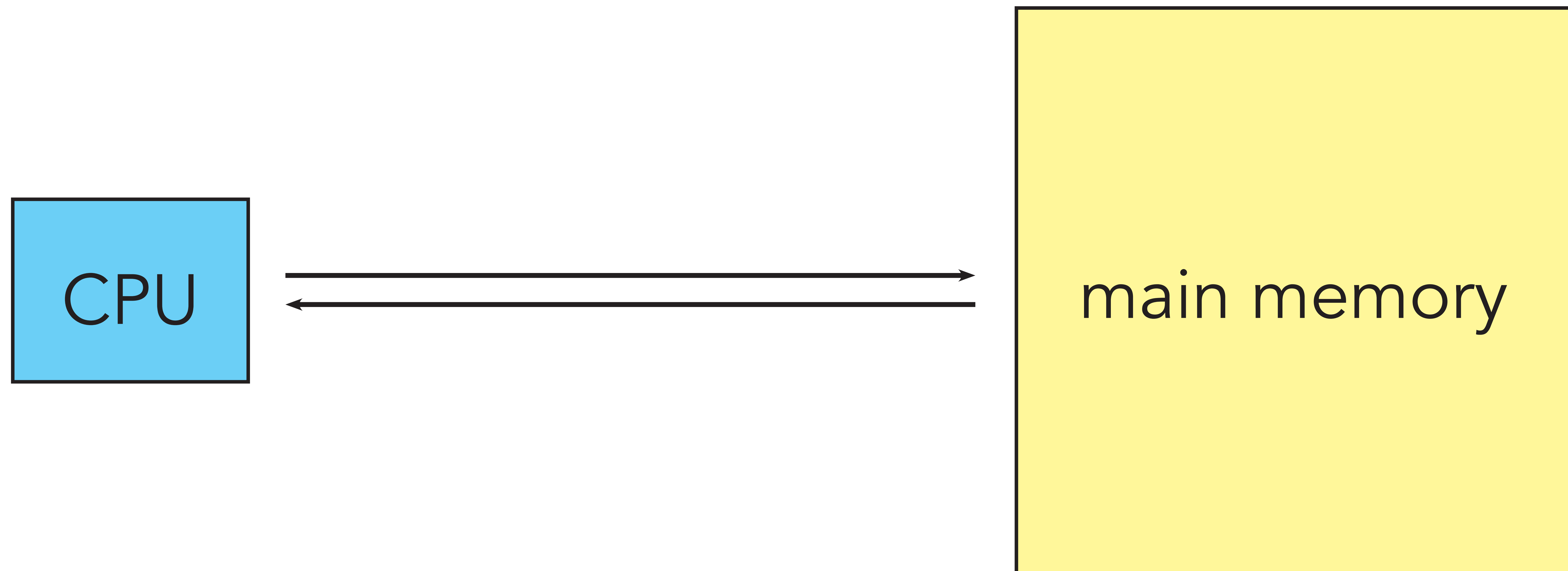
```
graph TD; A[von Neumann bottleneck] --> B[caching]; A --> C[pipelining];
```

caching

pipelining

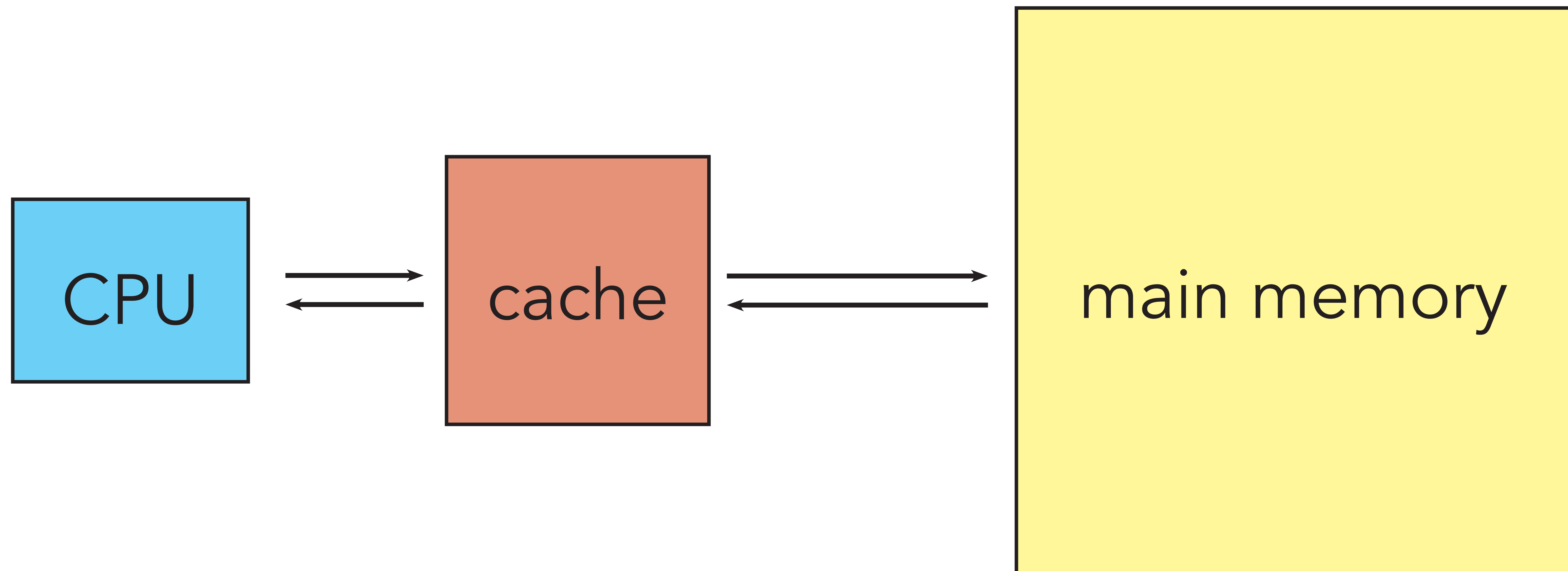
# Why parallel?

## Caching:



# Why parallel?

## Caching:



# Why parallel?

von Neumann bottleneck



```
graph TD; A[von Neumann bottleneck] --> B[caching]; A --> C[pipelining];
```

caching

pipelining

# Why parallel?

## Pipelining:



compute



memory

# Why parallel?

## Pipelining:



 compute     memory



# Why parallel?

## Pipelining:



 compute     memory

# Why parallel?

## Pipelining:



 compute     memory

# Why parallel?

## Pipelining:



 compute     memory

# Why parallel?

## Pipelining:



 compute     memory

# Why parallel?

## Pipelining:



 compute     memory

# Why parallel?

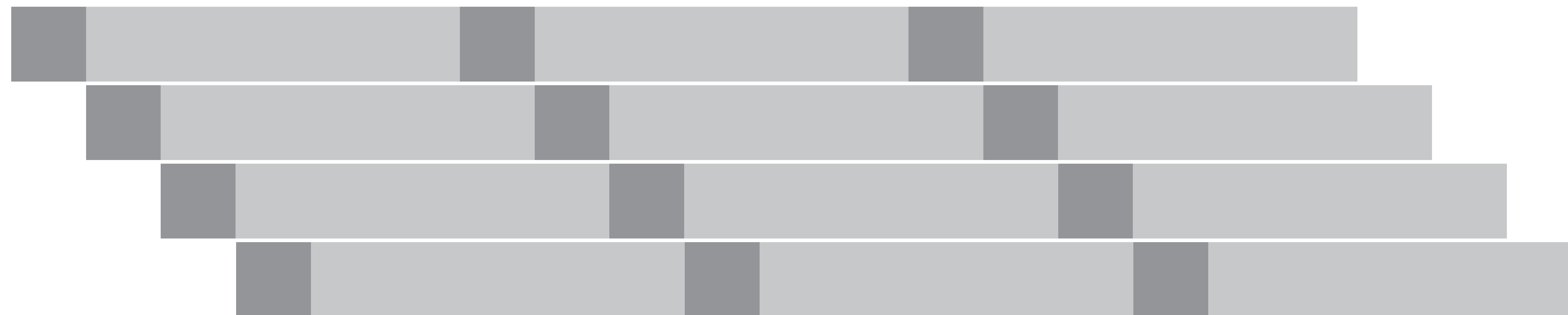
## Pipelining:



 compute     memory

# Why parallel?

## Pipelining:

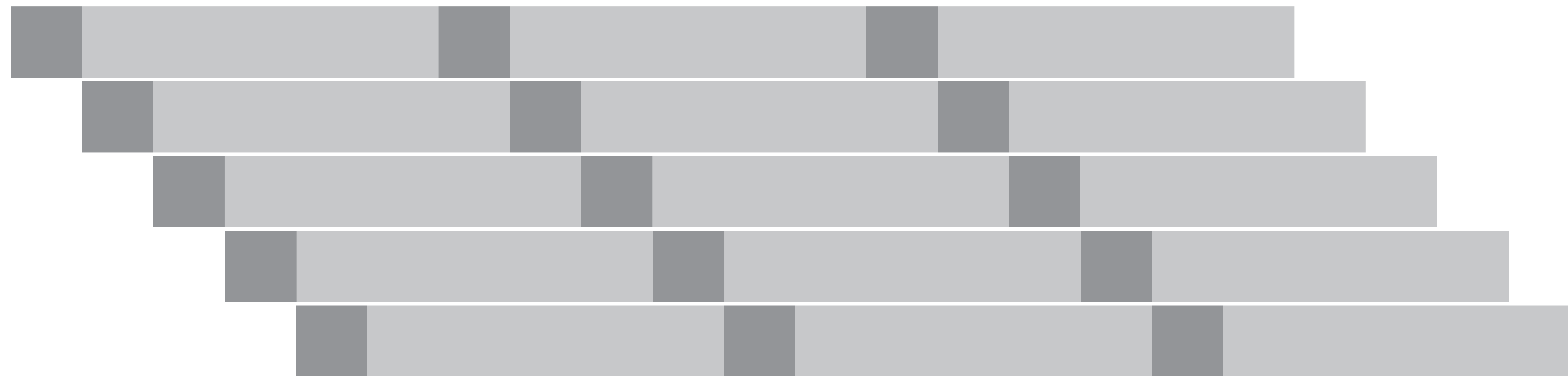


 compute     memory



# Why parallel?

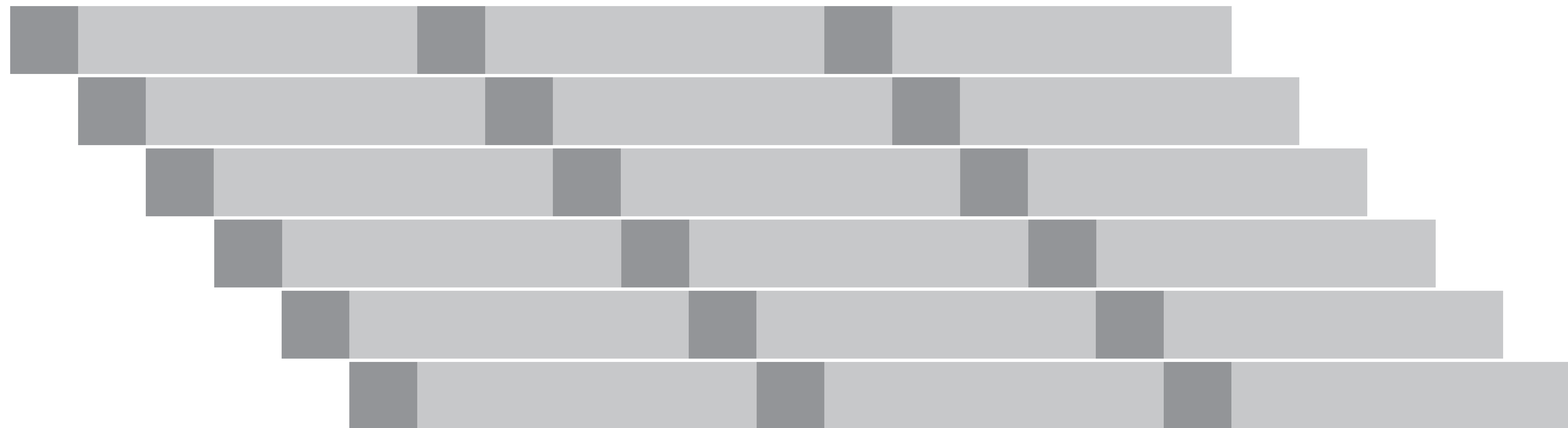
## Pipelining:



■ compute    ■ memory

# Why parallel?

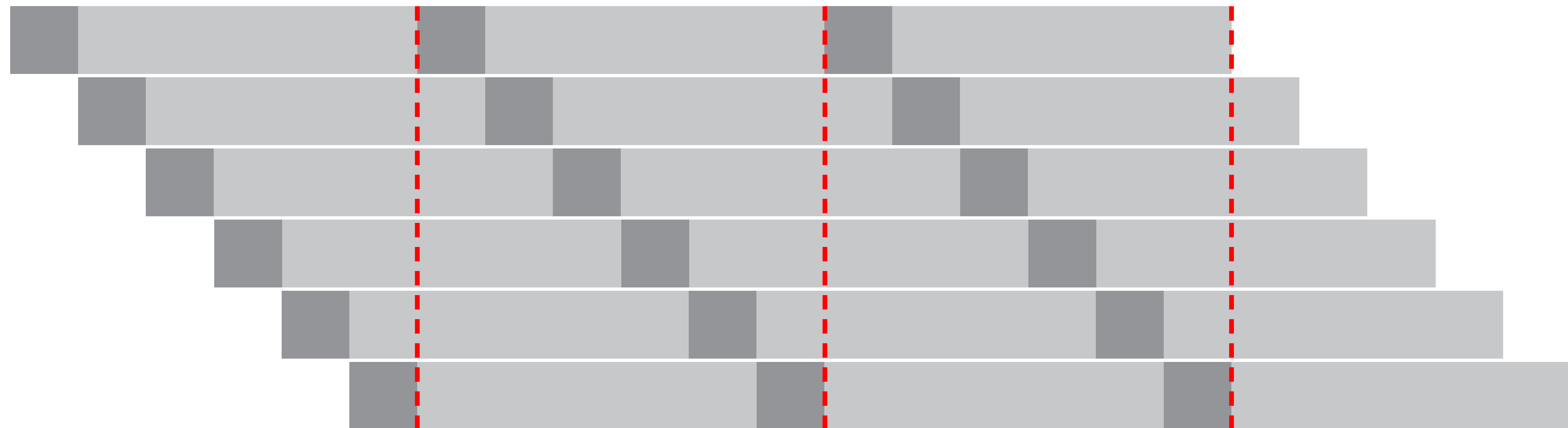
## Pipelining:



■ compute    ■ memory

# Why parallel?

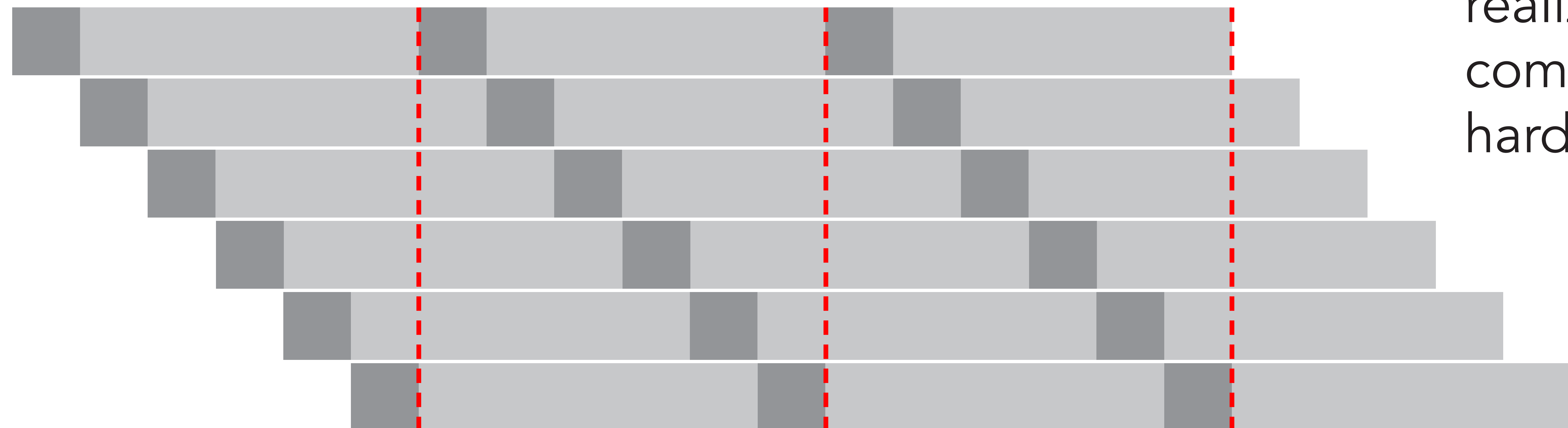
## Pipelining:



■ compute    ■ memory

# Why parallel?

## Pipelining:



typically  
realized by  
compiler or  
hardware

■ compute    ■ memory

# Why parallel?

Instruction level parallelism: exploit independence at assembler level

- Pipelining (memory access and computation)
- Different arithmetic units (diff. computations)

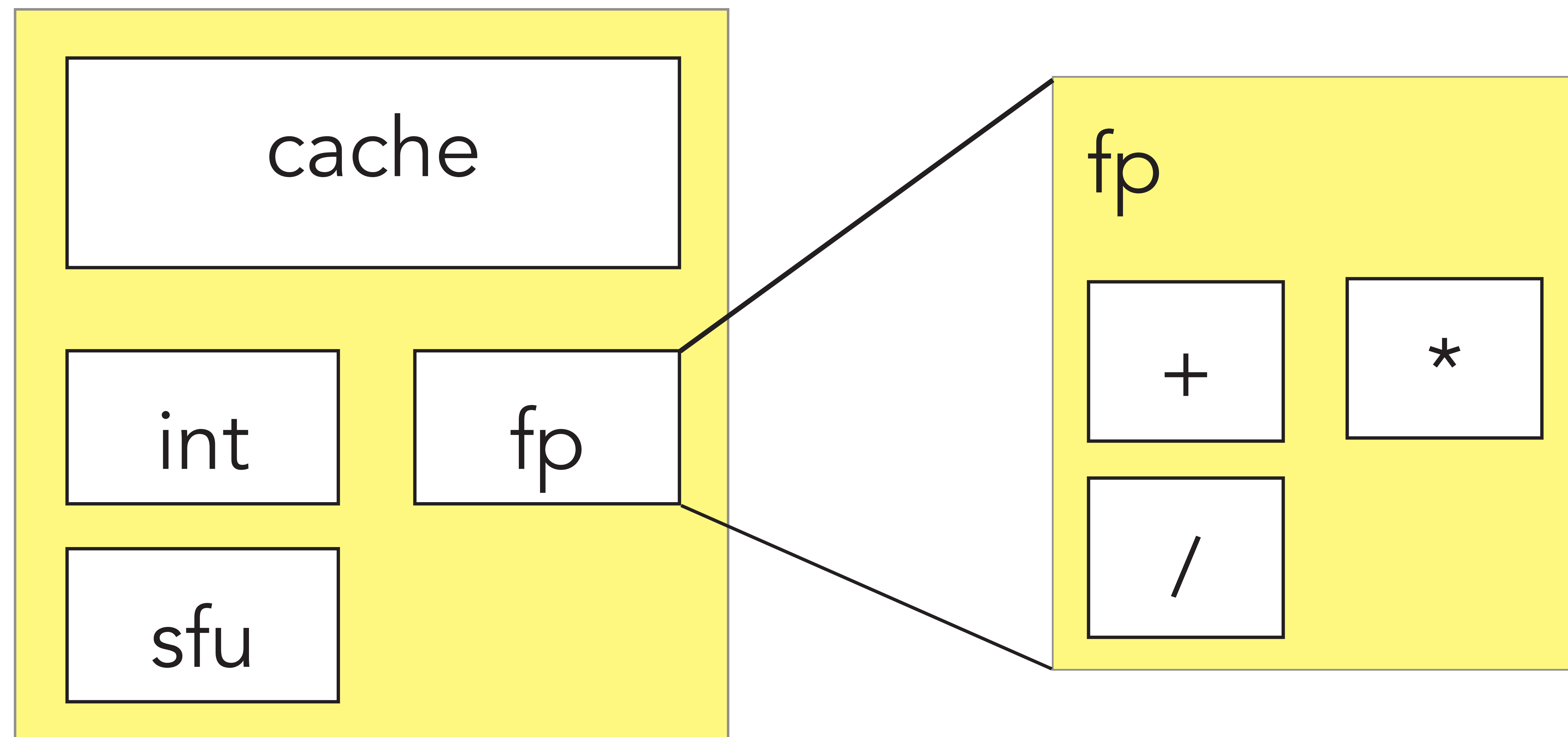
# Why parallel?

Functional unit	Latency	Initiation interval
Integer ALU	0	1
Data memory (integer and FP loads)	1	1
FP add	3	1
FP multiply (also integer multiply)	6	1
FP divide (also integer divide)	24	25

J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, Seventh ed. Morgan Kaufmann, 2017, p. C-53

# Why parallel?

core / CPU





# Instruction level parallelism

Out-of-order execution:

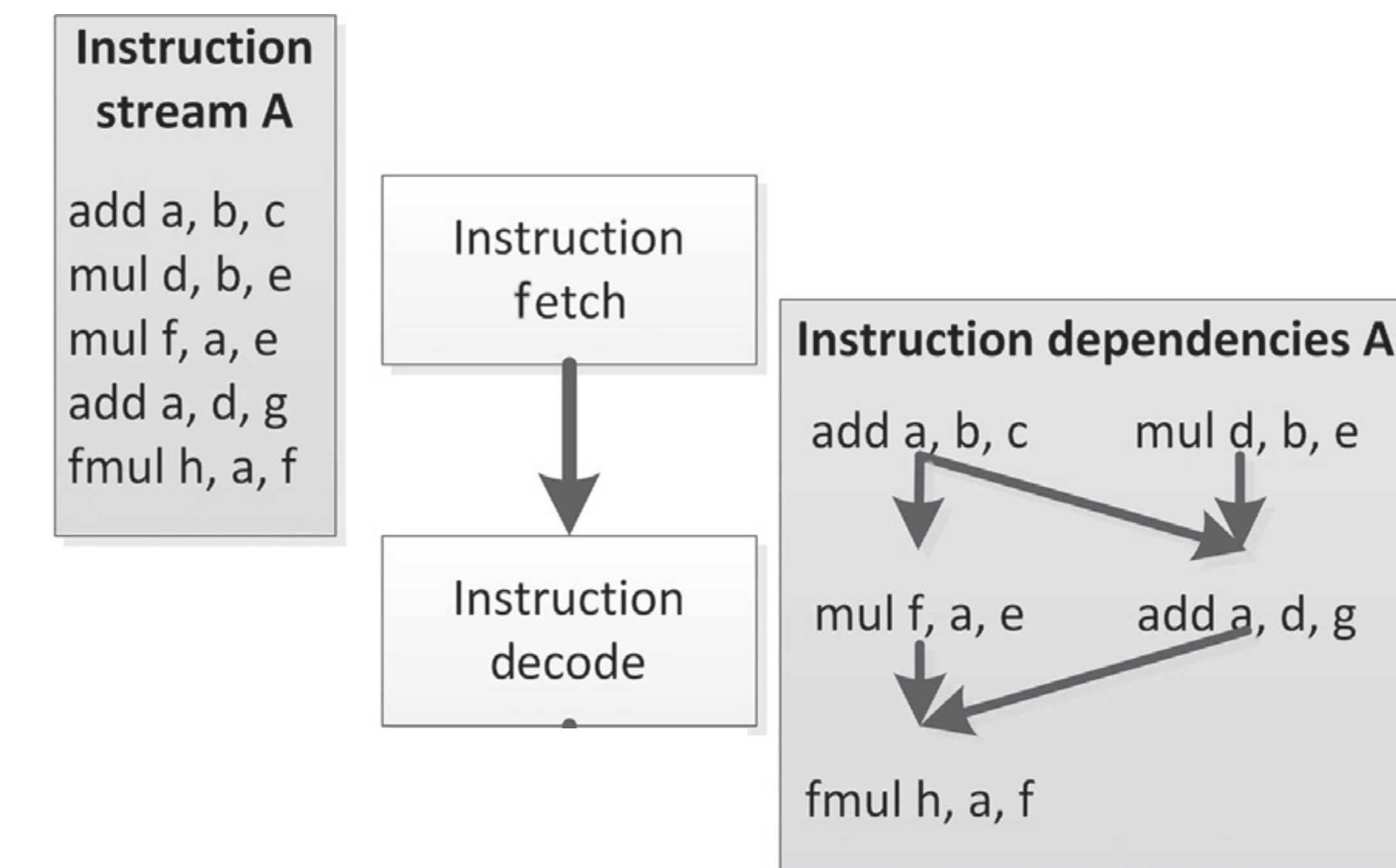
**Instruction  
stream A**

```
add a, b, c  
mul d, b, e  
mul f, a, e  
add a, d, g  
fmul h, a, f
```

D. R. Kaeli, P. Mistry, D. Schaa, and D. P. Zhang, Heterogeneous computing with OpenCL 2.0, Ch. 2.

# Instruction level parallelism

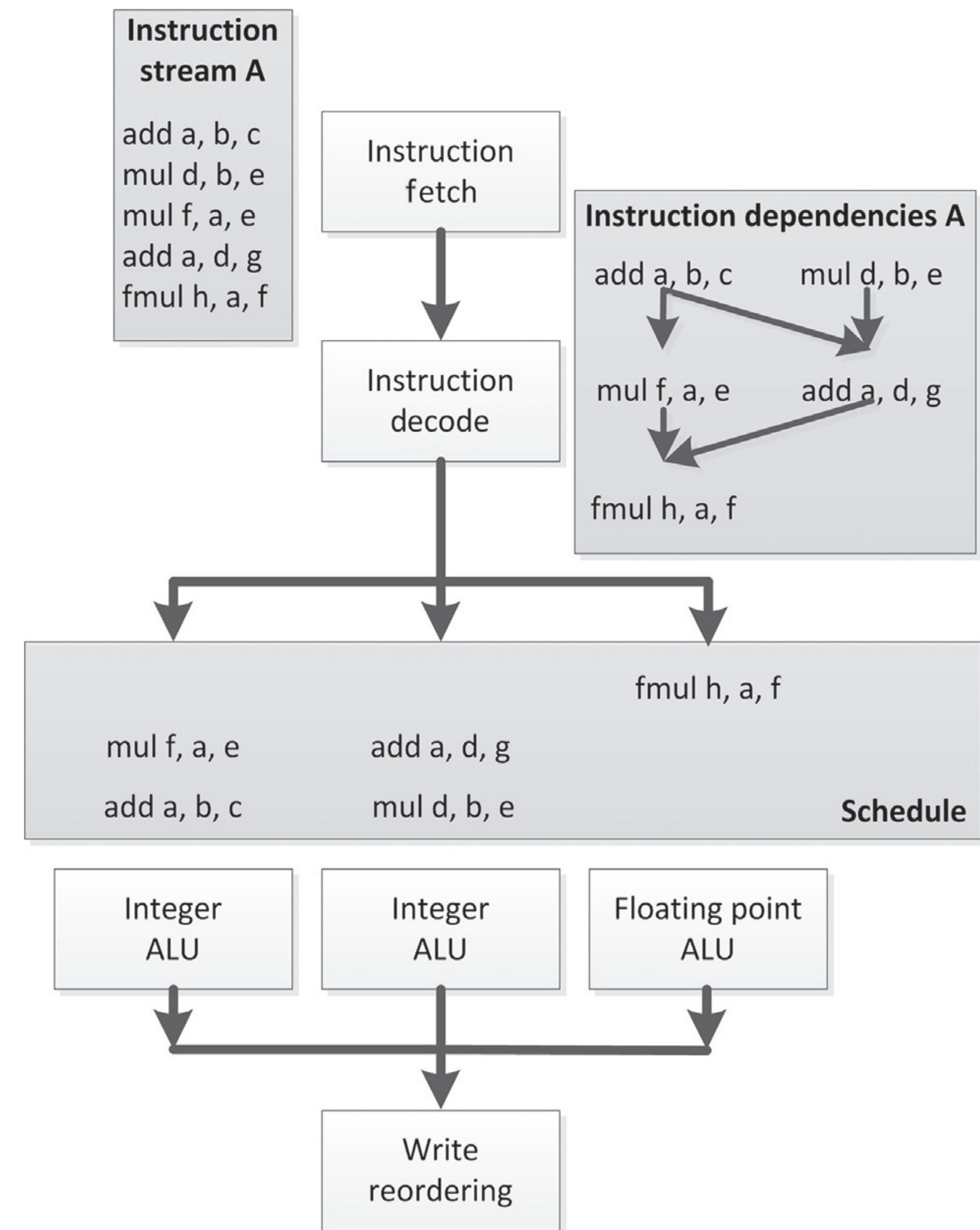
Out-of-order execution:



D. R. Kaeli, P. Mistry, D. Schaa, and D. P. Zhang, Heterogeneous computing with OpenCL 2.0, Ch. 2.

# Instruction level parallelism

Out-of-order execution:



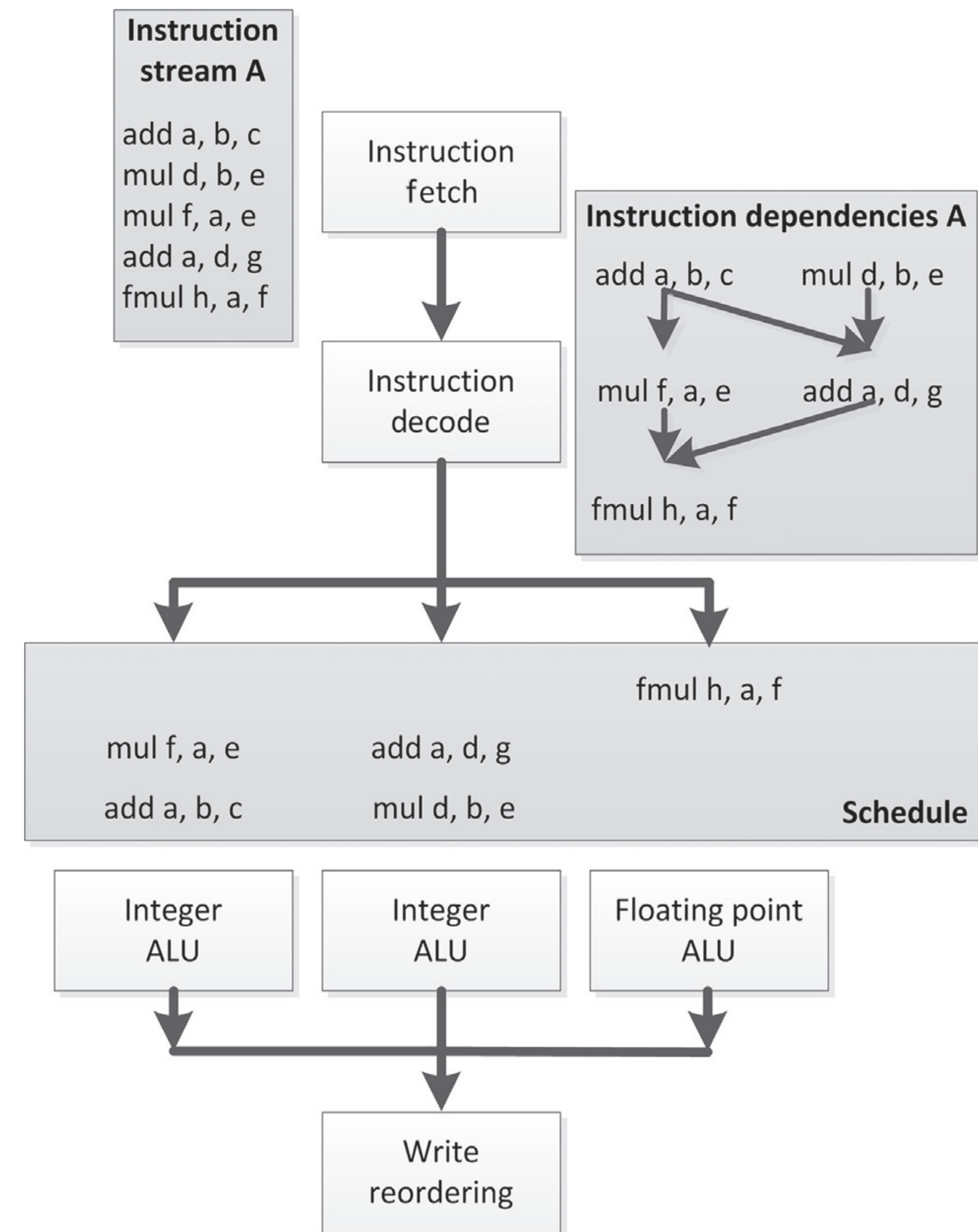
D. R. Kaeli, P. Mistry, D. Schaa, and D. P. Zhang, Heterogeneous computing with OpenCL 2.0, Ch. 2.



# Instruction level parallelism

Out-of-order execution:

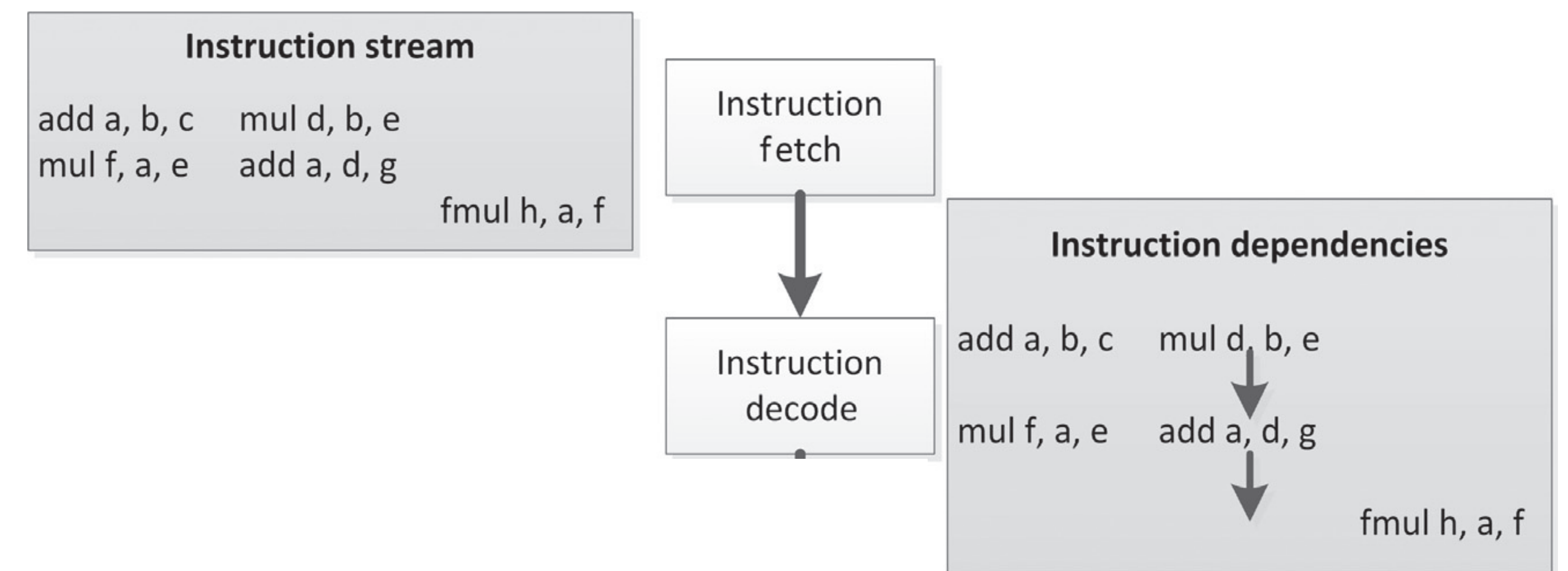
processor extracts parallelism  
from instruction stream



D. R. Kaeli, P. Mistry, D. Schaa, and D. P. Zhang, Heterogeneous computing with OpenCL 2.0, Ch. 2.

# Instruction level parallelism

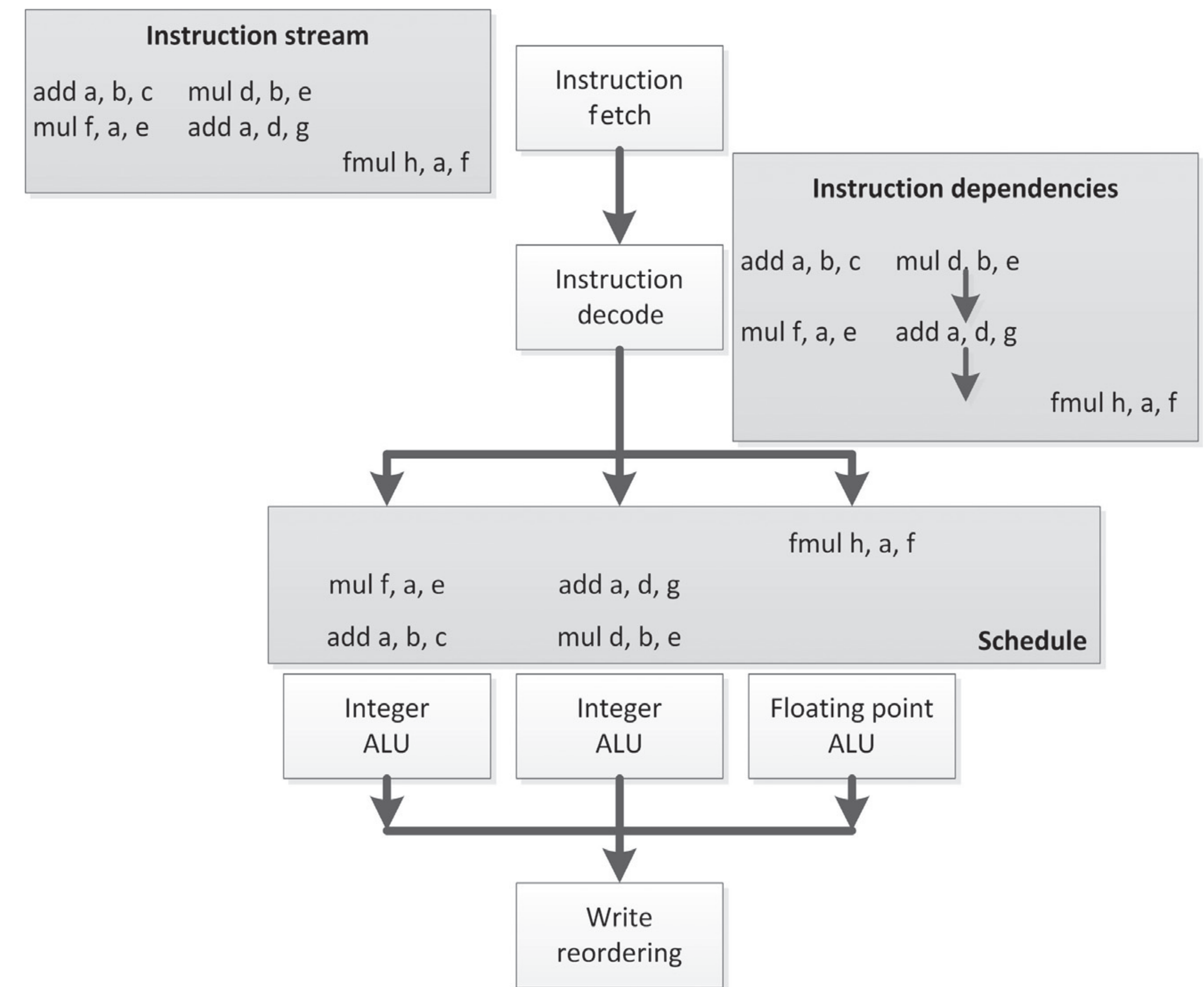
Very long instruction word processors:



# Instruction level parallelism

Very long instruction word  
processors:

compiler extracts parallelism  
from program code





# Why parallel?

Instruction level parallelism: exploit independence at assembler level

- Pipelining
- Different arithmetic units

=> Exploited since 1980s and standard, but no longer significant improvements



# Further reading

- J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, fourth edition. Morgan Kaufmann, 2007.
- <http://cva.stanford.edu/classes/cs99s/>
- <http://research.ac.upc.edu/HPCseminar/SEM9900/Pollack1.pdf>
- <http://groups.csail.mit.edu/cag/raw/documents/Waingold-Computer-1997.pdf>
- <http://cacm.acm.org/magazines/2009/5/24648-spending-moores-dividend/fulltext>