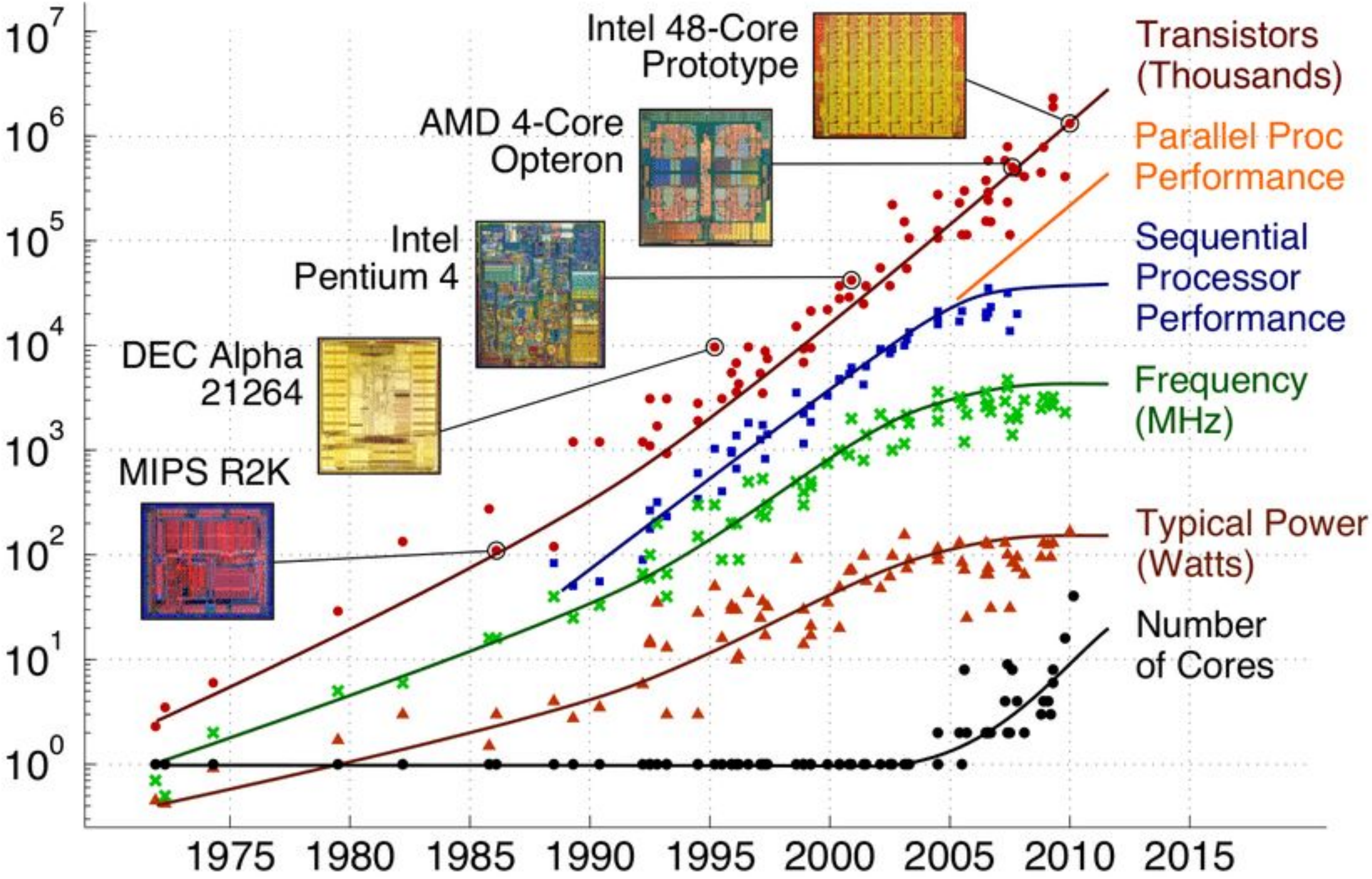
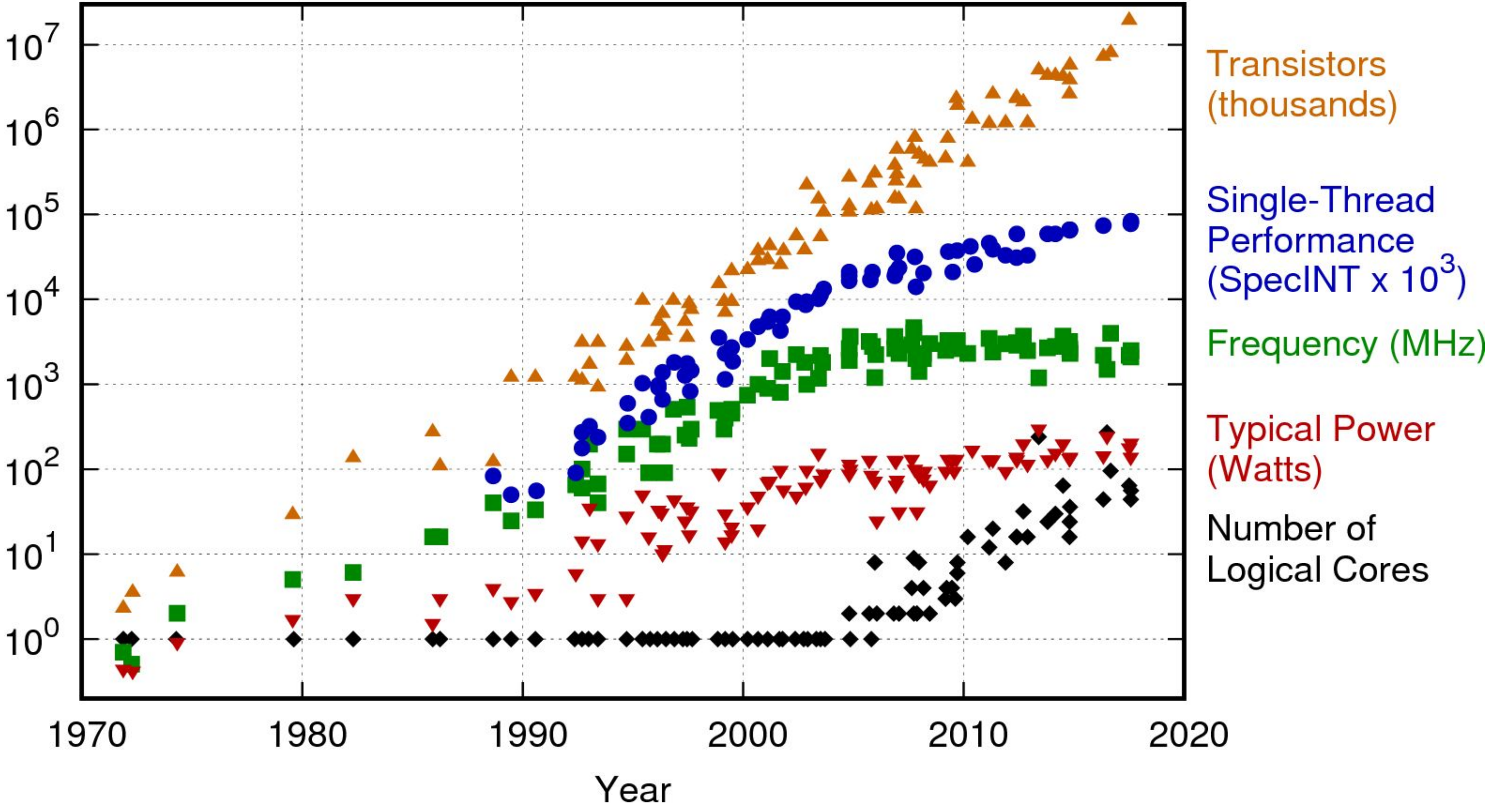
Advanced architecture

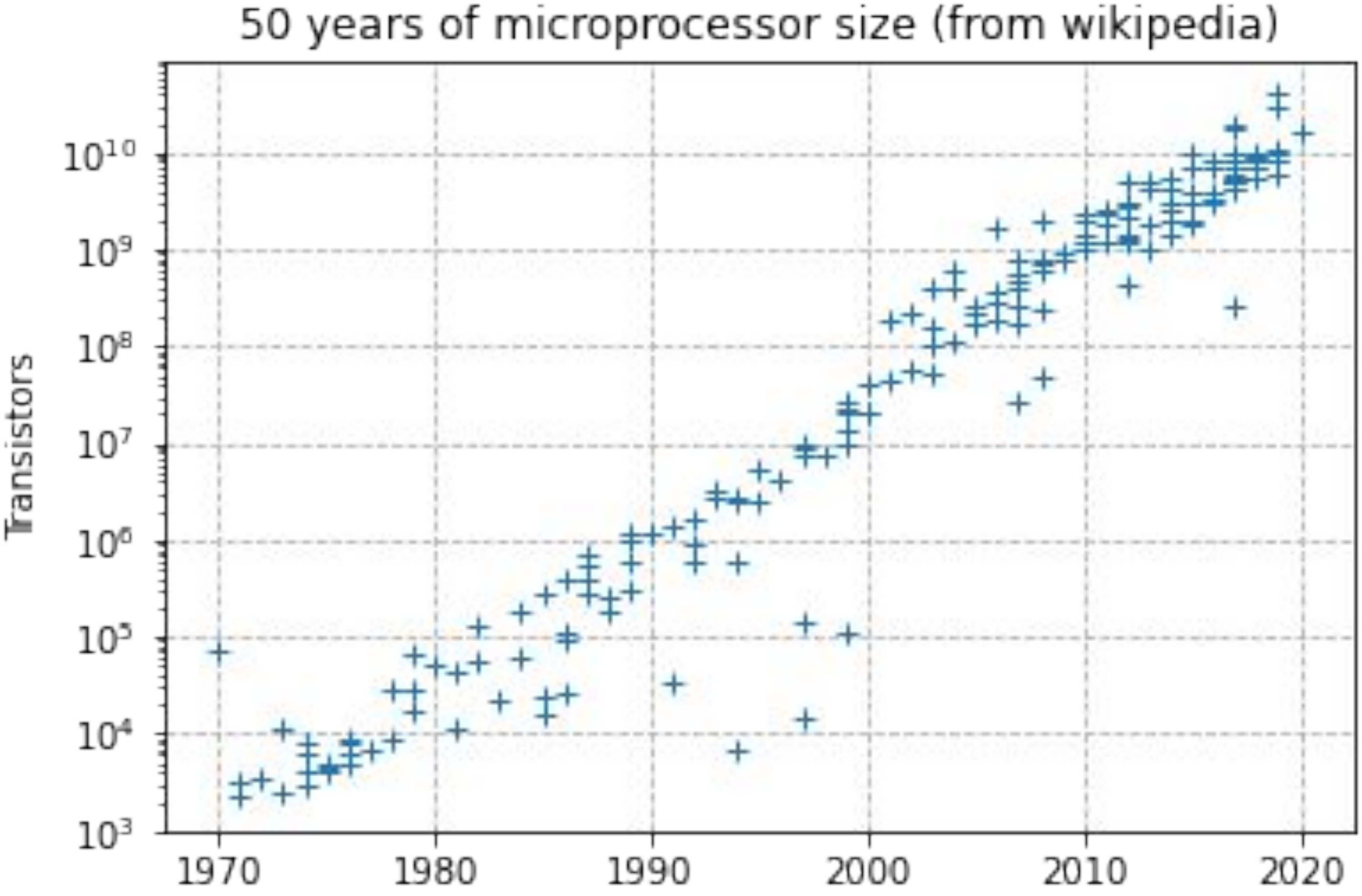


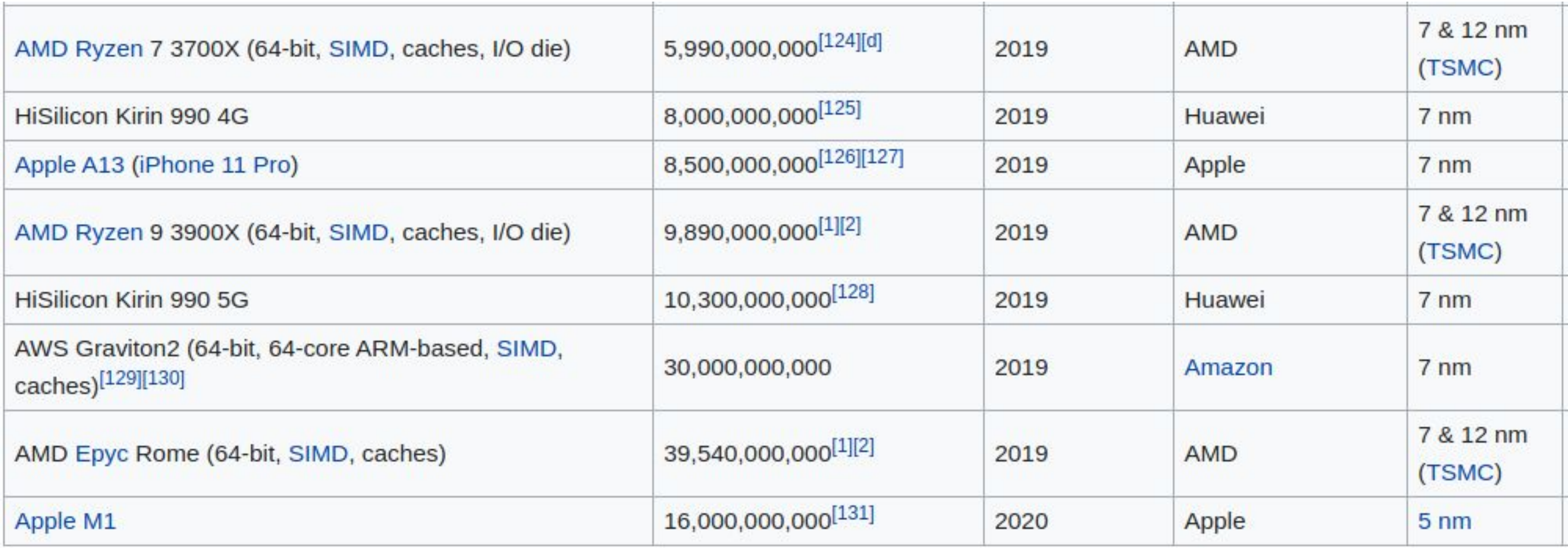
Intel 48-Core Prototype
AMD 4-Core Opteron
Intel Pentium 4
DEC Alpha 21264
MIPS R2K
Transistors (Thousands)
Parallel Proc Performance
Sequential Processor Performance
Frequency (MHz)
Typical Power (Watts)
Number of Cores
10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
1975
1980
1985
1990
1995
2000
2005
2010
2015

42 Years of Microprocessor Trend Data



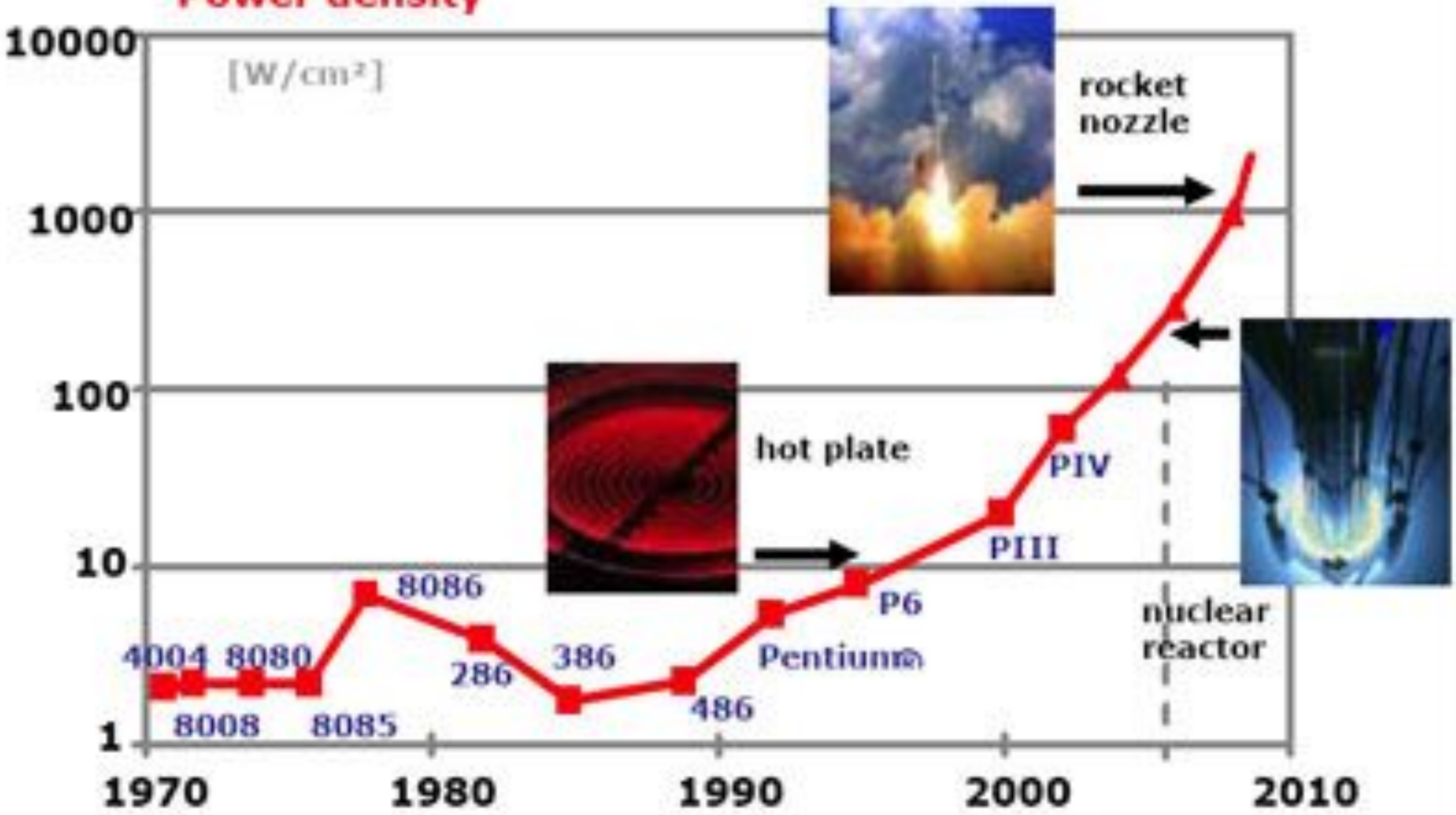
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp


50 years of microprocessor size (from wikipedia)
Transistors
10^3
10^4
10^5
10^6
10^7
10^8
10^9
10^10
1970
1980
1990
2000
2010
2020



AMD Ryzen 7 3700X (64-bit, SIMD, caches, I/O die)	5,990,000,000[124][d]	2019	AMD	7 & 12 nm (TSMC)
HiSilicon Kirin 990 4G	8,000,000,000[125]	2019	Huawei	7 nm
Apple A13 (iPhone 11 Pro)	8,500,000,000[126][127]	2019	Apple	7 nm
AMD Ryzen 9 3900X (64-bit, SIMD, caches, I/O die)	9,890,000,000[1][2]	2019	AMD	7 & 12 nm (TSMC)
HiSilicon Kirin 990 5G	10,300,000,000[128]	2019	Huawei	7 nm
AWS Graviton2 (64-bit, 64-core ARM-based, SIMD, caches)[129][130]	30,000,000,000	2019	Amazon	7 nm
AMD Epyc Rome (64-bit, SIMD, caches)	39,540,000,000[1][2]	2019	AMD	7 & 12 nm (TSMC)
Apple M1	16,000,000,000[131]	2020	Apple	5 nm

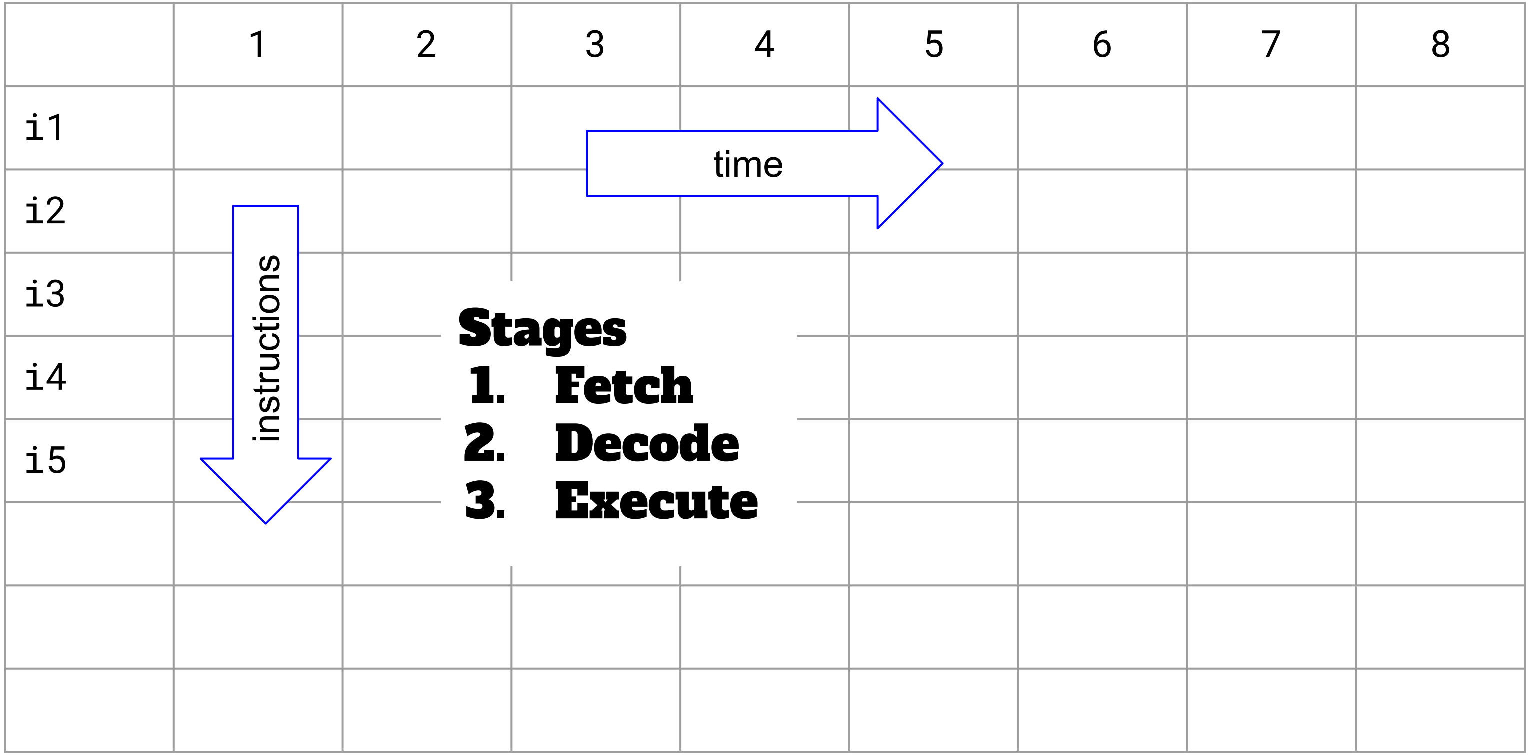
Power density



So many transistors!

- What are we doing with so many transistors?
- Lots of clever things
  - Lots of cache
  - Pipelining
  - Branch prediction
  - Delayed branching
  - Superscalar execution
  - Out-of-order execution
  - Multiple cores
  - Hyper-threading

Pipelining example



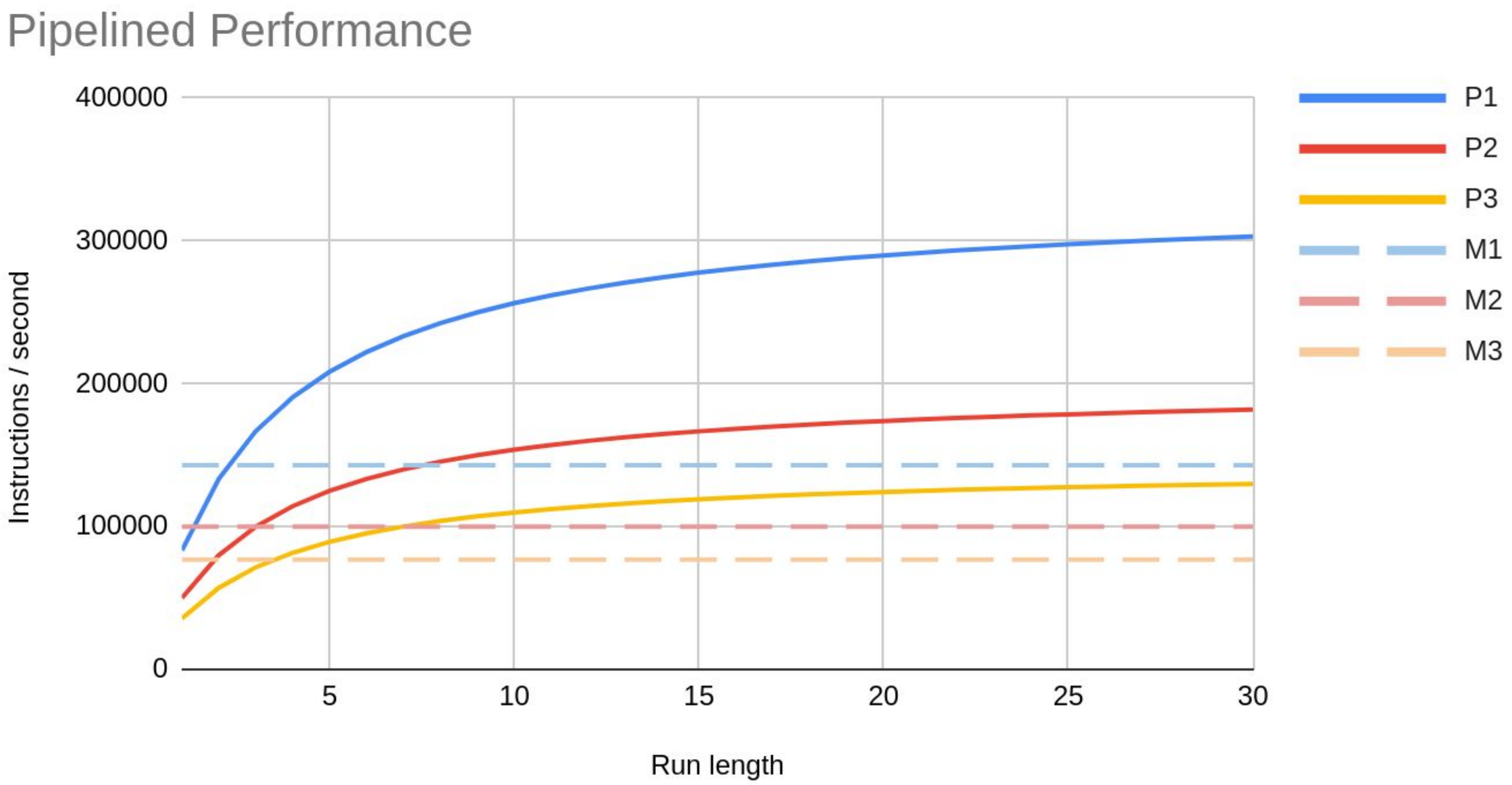
	1	2	3	4	5	6	7	8
i1								
i2								
i3								
i4								
i5								

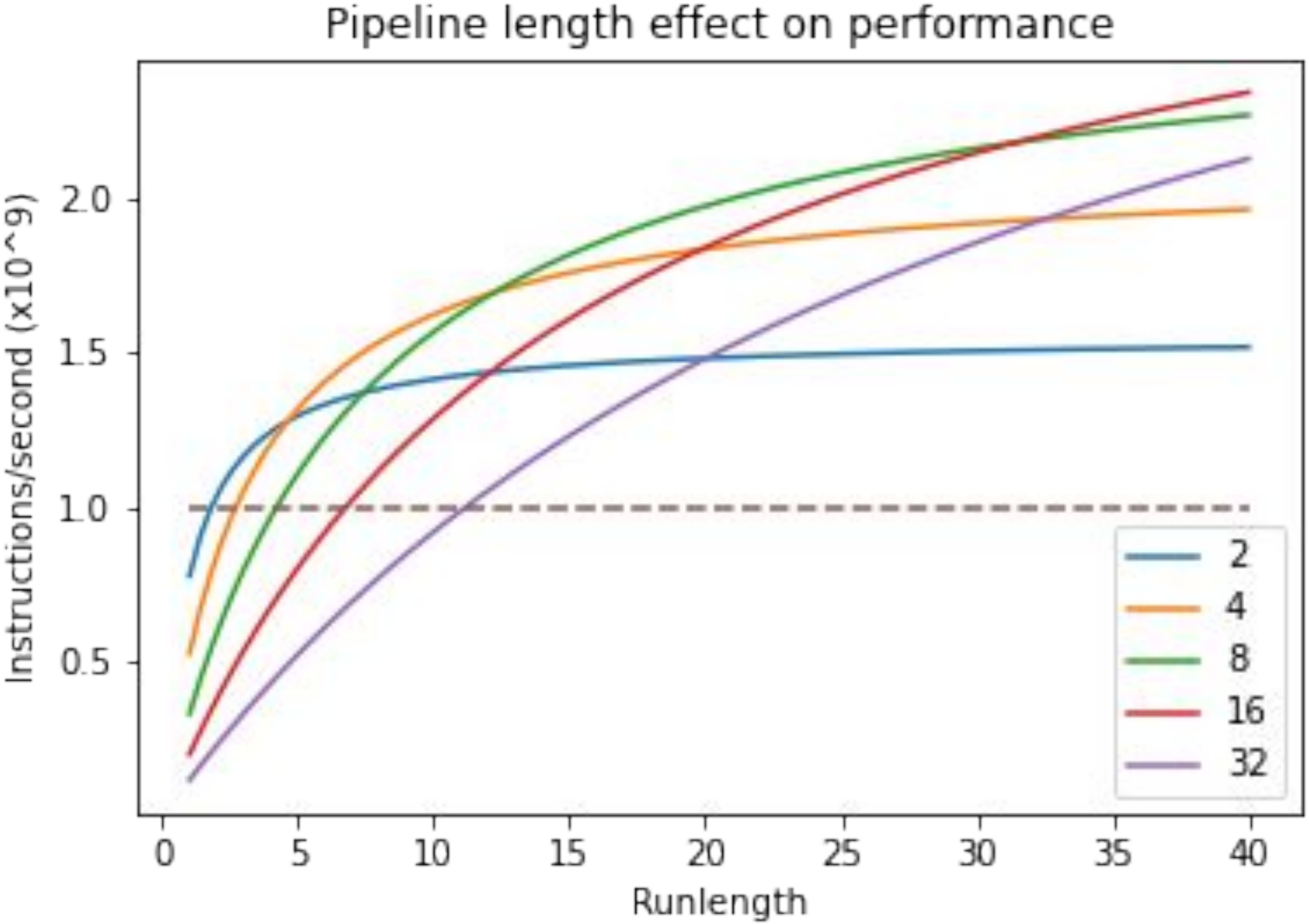
Stages

1. **Fetch**
2. **Decode**
3. **Execute**

Pipelining example

	1	2	3	4	5	6	7	8
i1	F	D	E					
i2		F	D	E				
i3			F	D	E			
i4				F	D	E		
i5					F	D	E	
i6						F	D	E


Pipelined Performance
400000
300000
200000
100000
0
Instructions / second
5
10
15
20
25
30
Run length
P1
P2
P3
M1
M2
M3



$$\text{IPS} = \frac{L^{*}f(S)}{L+S-1} \times 10^{9}$$

$$t(I) = f(S)^{*}C$$

Branch prediction

- Want to have long run lengths
  - What to do on branch?
  - Four choices

Stop

- **Drains pipeline**
- **Poor performance**

Guess

- **Right: no penalty**
- **Wrong:**
  - **Same as stopping**
  - **Well, some cost**

Predict

- **Same as guess**
- **Improved accuracy**

Fetch both

- **Best performance**
- **Highest cost**
- **Limited**

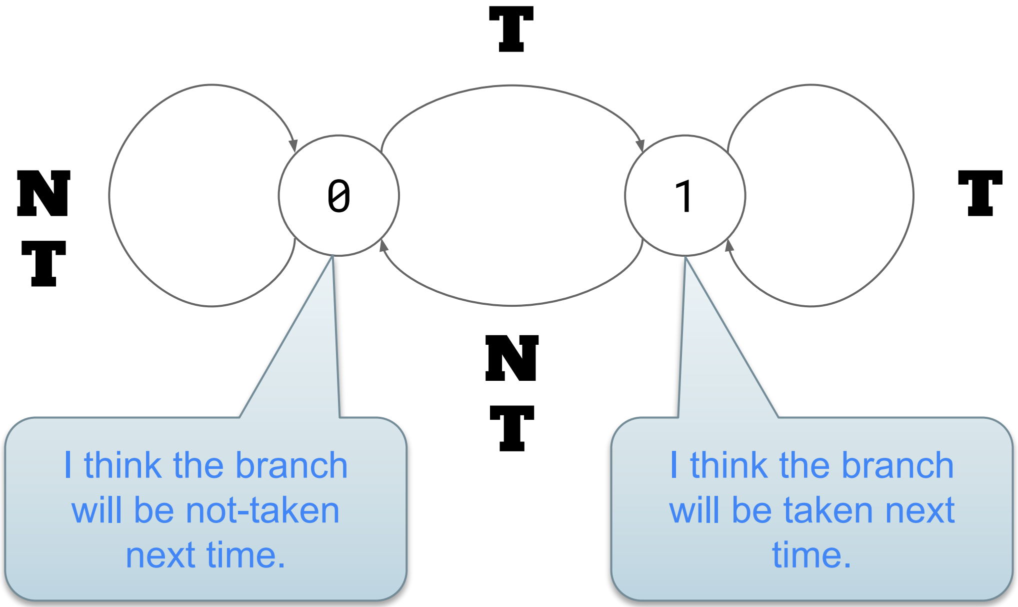
Stop

	1	2	3	4	5	6	7	8
i1	F	D	E					
i2		F	D	E				
i3			F	D	E			
je				F	D	E		
i5					🟥	🟥	F	D
i6						🟥		F
i7							F	D
i8								F

Mispredict

	1	2	3	4	5	6	7	8
i1	F	D	E					
i2		F	D	E				
i3			F	D	E			
je				F	D	E		
i5					F	D		
i6						F		
i7							F	D
i8								F

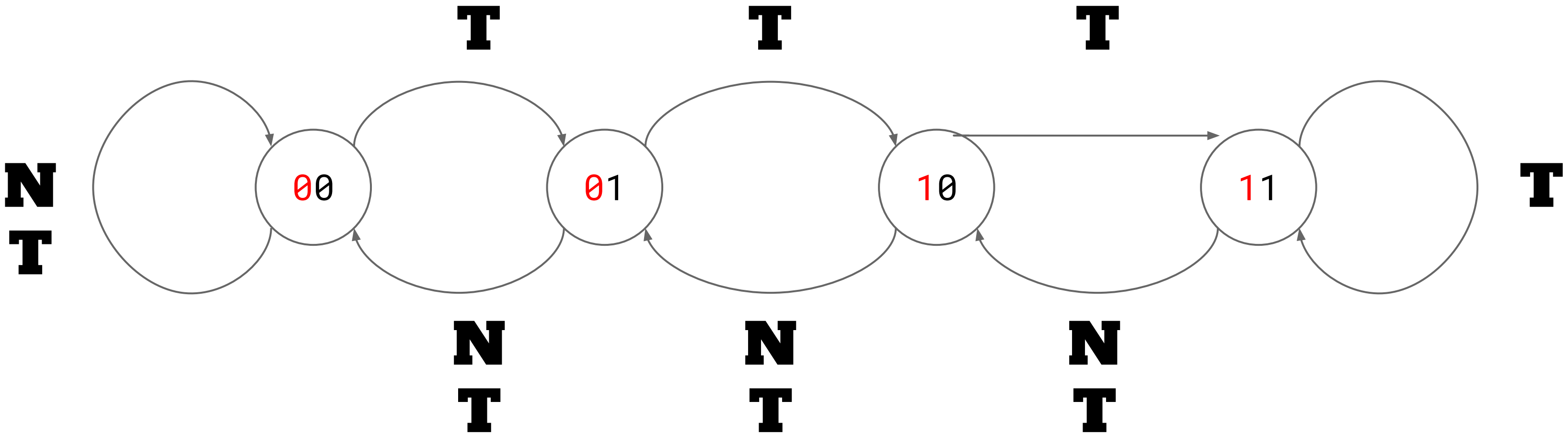
1-bit branch predictor



I think the branch will be not-taken next time.

I think the branch will be taken next time.

2-bit branch predictor



Delayed branching

```
repit:   cmp  ax,0
    je   iszero     ;ax=0
    add  ax,1
    jmp  repit

         …

iszero:             ;ax=1
```

- Instruction following conditional jump is **always** executed

Delay slot

```
rep: cmp  [si],0
     jge  skip
     ??????????????
     mov  [si],0
skip:
     inc  si
     loop rep
```

```
rep: cmp  [si],0
     jge  skip
     inc  si
     mov  [si-1],0
skip:
     loop rep
```

Must move an instruction into the *delay slot*

Delayed branching

	1	2	3	4	5	6	7	8
mov	F	D	E					
add		F	D	E				
jo			F	D	E			
inc				F	D			
mov					F			

Delayed branching

	1	2	3	4	5	6	7	8
add	F	D	E					
jo		F	D	E				
mov			F	D	E			
inc				F	D			
mov					F			

ILP — instruction-level parallelism

Three statements (instructions)

```
e = a + b

f = c + d

g = e * f
```

independent

Can execute first two in either order or in parallel

```
e = a + b   f = c + d

g = e * f
```

ILP is 3/2 for these instructions.

- **Superscalar execution : finish two or more instructions in the same cycle.**

Superscalar

	1	2	3	4	5	6	7	8
i1	F	D	E	S				
i2	F	D	E	S				
i3		F	D	E	S			
i4		F	D	E	S			
i5			F	D	E	S		
i6			F	D	E	S		
i7				F	D	E	S	
i8				F	D	E	S	

Multiple functional units

- For two-way superscalar must have 2 of everything
  - Two fetch units
  - Two decode units
  - Two execute units, including all parts of the ALU
  - …

Multiple functional units

- For two-way superscalar must have 2 of everything
    - Two fetch units
    - Two decode units
    - Two execute units, including all parts of the ALU
    - …
- Can we really build a processor with duplicate hardware?

  Yes. Yes we can.
- Even the the 8086 has multiple functional units
    - Consider `add ax, [si+dx]`
    - How many adders?

Superscalar execution

- Some pairs of instructions won’t occur in a typical program

```
jmp     left
jmp     right
```

```
cmp     ax, 25
cmp     ax, 30
```

```
inc     bx
dec     bx
```

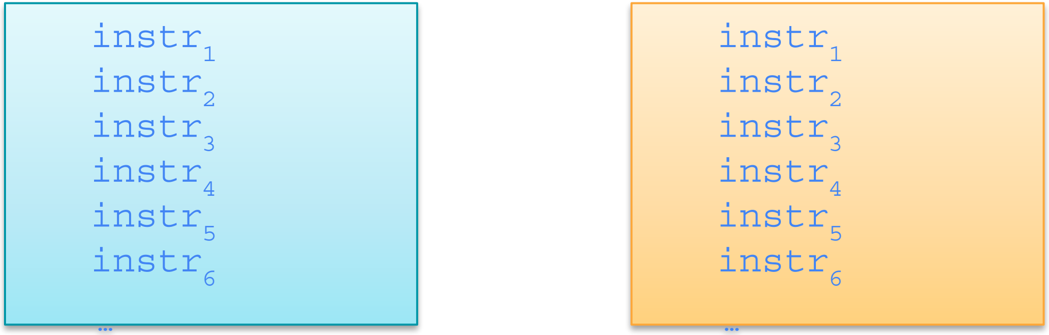
- Some pairs of instructions may have interdependencies
  - Preventing them from being run concurrently.

```
add     ax, [var1]
add     bx, ax
```

- Where can we get lots of code with no interdependency between instructions?

Running Multiple Threads

- Instructions in one program (or thread) are typical independent of those in another thread.



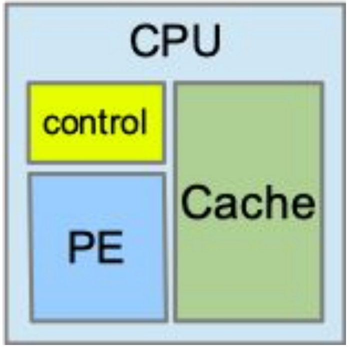
instr$_1$
instr$_2$
instr$_3$
instr$_4$
instr$_5$
instr$_6$
…

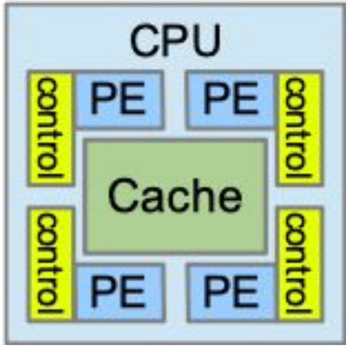
instr$_1$
instr$_2$
instr$_3$
instr$_4$
instr$_5$
instr$_6$
…

- This provides this is an opportunity for parallelism.
- Support execution of multiple instruction streams at the same time.

Multiple Cores

- Modern processors have multiple CPU cores.
  - Like having multiple processors on the same package.

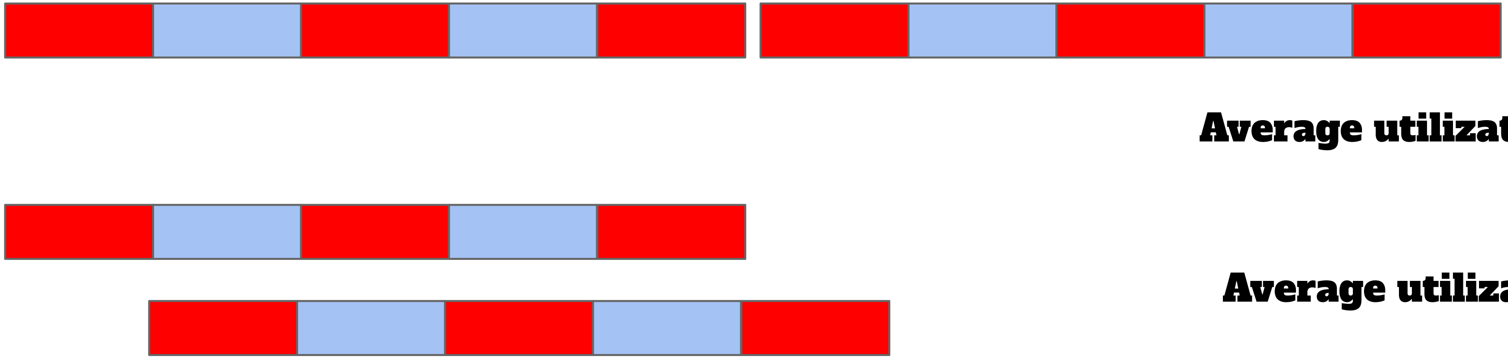




- We could accomplish this with multiple CPU packages.
- But that would require fabricating more than one CPU
- Multiple motherboard slots …

Multi-threading

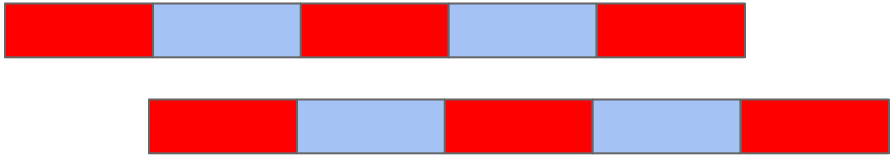
- Threads may not always keep their CPU busy
- … because of memory stalls, unused parts of the ALU, etc.



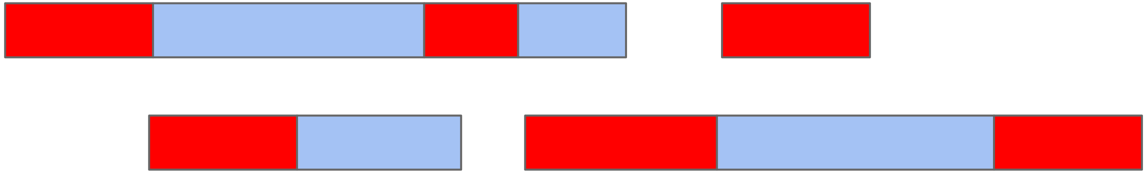
Average utilization = 60%

Average utilization = 100%

Multi-threading



Highly idealized, ie, false



Overly idealized, ie, false

There is a cost to switch between threads

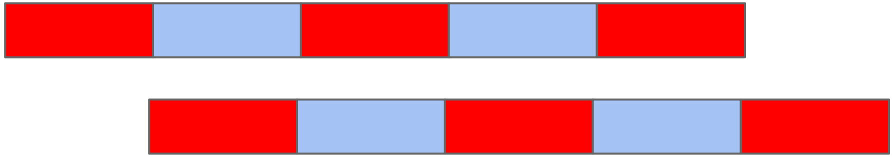
Hyperthreading

- Intel's name for *simultaneous multithreading*
- First, what is multithreading?
  - OS executes processes or threads
  - Multiple concurrent threads
  - OS *swaps* between threads
- Why?

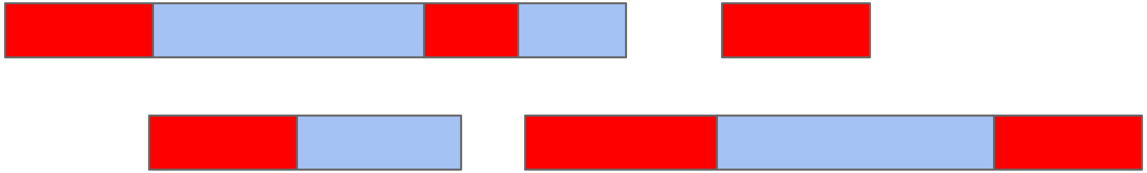
Content switch

- Thread context
  - Process info (PID, owner, …)
  - Page table, TLB
  - Stack
  - Microprocessor state (registers, …)
- Context switch
  - Save current thread’s context
  - Restore next thread’s context
- Secondary cost
  - Flush/refresh cache (especially the TLB)
  - Page buffers

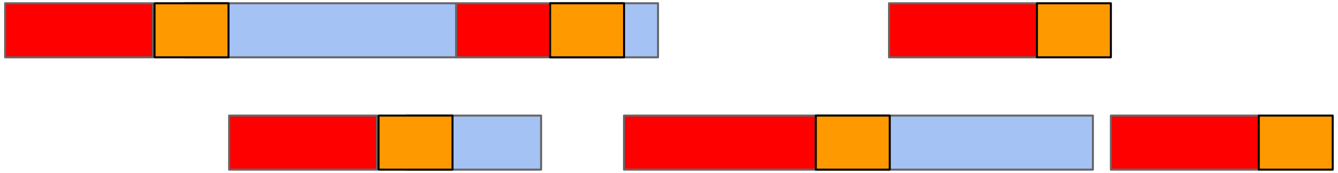
Multi-threading



Highly idealized, ie, false



Overly idealized, ie, false

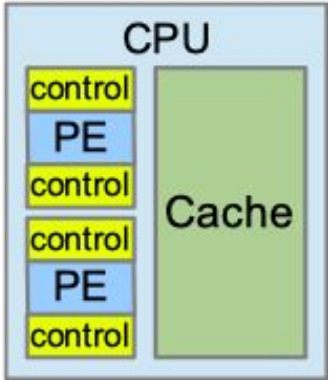


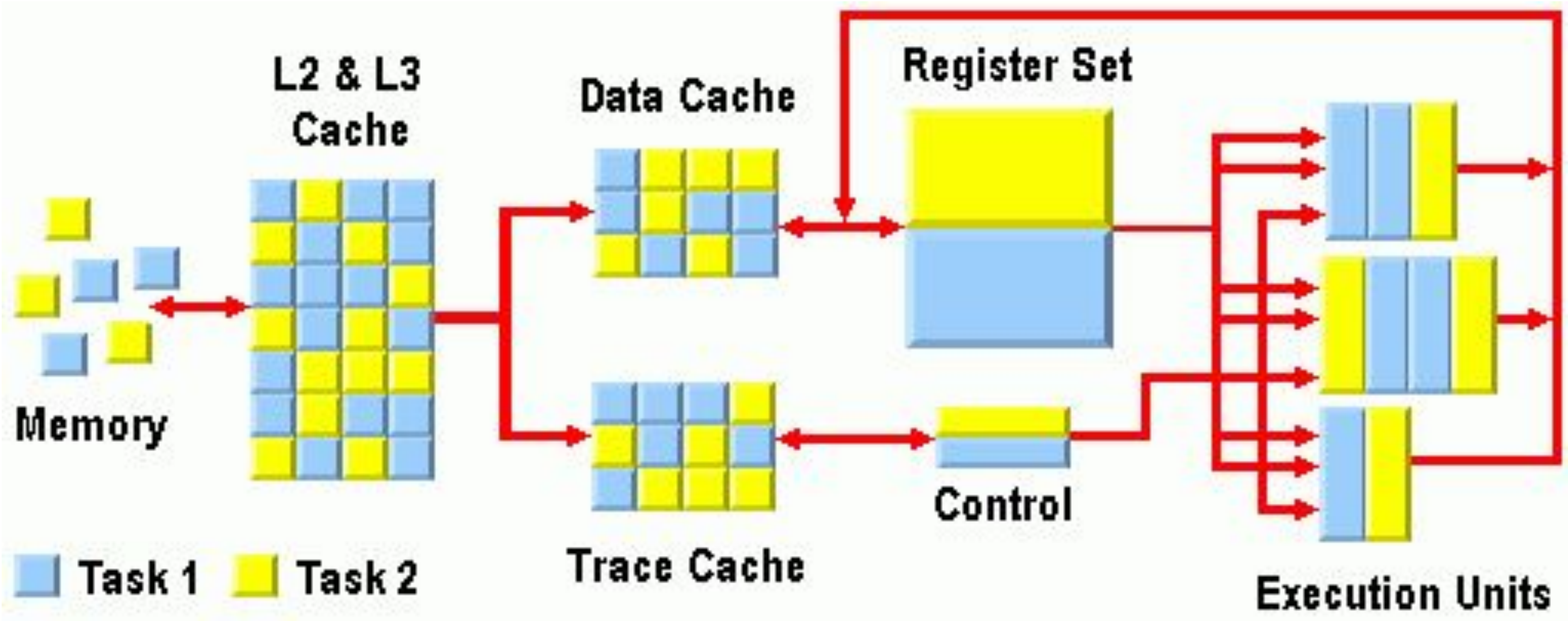
Hyperthreading

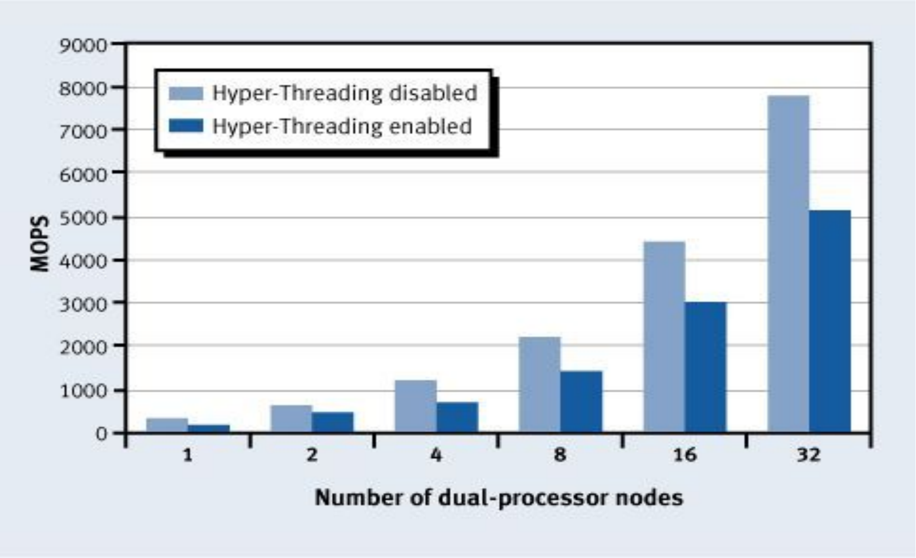
- Normally, the OS switches between running threads
    - When one thread has to wait for something (e.g., I/O).
    - But, some waits are too short for this to be practical
- Idea : let the CPU trade-off between a small number of running threads
- The CPU can pretend it has more cores than it really does
    - The OS can schedule these *virtual cores* as if they were real CPU cores
    - Internally, the processor can switch between them when one can't make progress.
- Intel calls this *hyper-threading.*

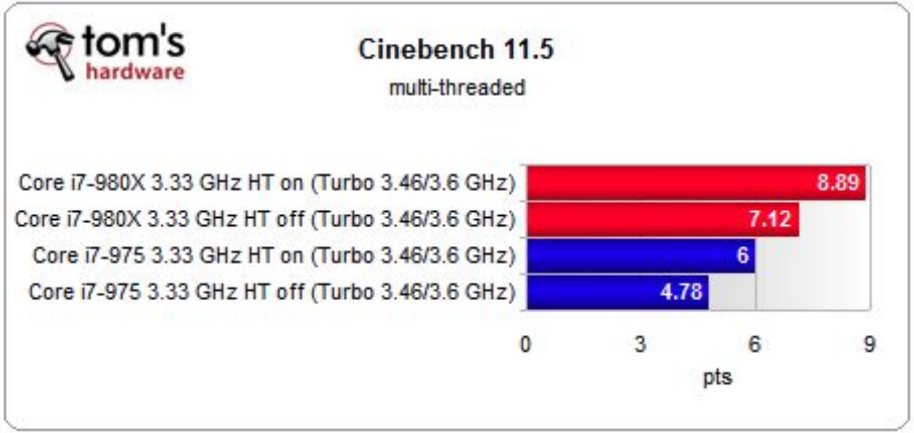
Hyperthreading

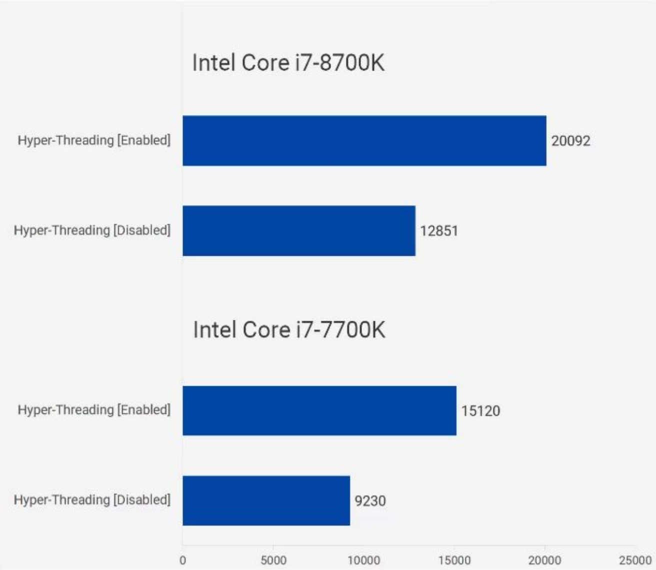
- The CPU can maintain a limited number of contexts for executing threads
  - Copies of the CPU registers
  - Cache contents
- OS sees each of these as if it was a CPU core.
- Internally, they share ALU and other hardware.

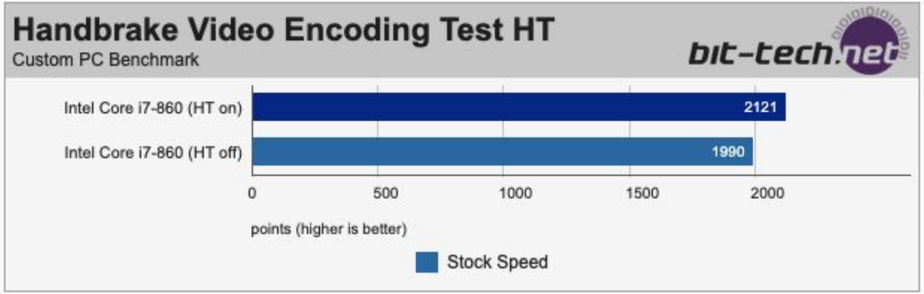



L2 & L3 Cache
Data Cache
Register Set
Memory
Control
Trace Cache
Execution Units
Task 1
Task 2


Hyper-Threading disabled
Hyper-Threading enabled
MOPS
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1
2
4
8
16
32
Number of dual-processor nodes


tom's hardware
Cinebench 11.5
multi-threaded
Core i7-980X 3.33 GHz HT on (Turbo 3.46/3.6 GHz) 8.89
Core i7-980X 3.33 GHz HT off (Turbo 3.46/3.6 GHz) 7.12
Core i7-975 3.33 GHz HT on (Turbo 3.46/3.6 GHz) 6
Core i7-975 3.33 GHz HT off (Turbo 3.46/3.6 GHz) 4.78
0
3
6
9
pts


Intel Core i7-8700K
Hyper-Threading [Enabled] 20092
Hyper-Threading [Disabled] 12851
Intel Core i7-7700K
Hyper-Threading [Enabled] 15120
Hyper-Threading [Disabled] 9230
0
5000
10000
15000
20000
25000


Handbrake Video Encoding Test HT
Custom PC Benchmark
bit-tech.net
Intel Core i7-860 (HT on) 2121
Intel Core i7-860 (HT off) 1990
0
500
1000
1500
2000
points (higher is better)
Stock Speed

References

- https://www.researchgate.net/figure/Evolution-of-multi-core-processors-1_fig1_281534326
- https://www.extremetech.com/extreme/203490-moores-law-is-dead-long-live-moores-law
- https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/
- http://www.nanowerk.com/spotlight/spotid=1762.php
- http://www.pctechguide.com/pentium-cpus/hyper-threading-technology
- https://www.makeuseof.com/tag/hyperthreading-technology-explained/
- https://en.wikipedia.org/wiki/Transistor_count