Post x86

8086

1 MB in 64KB segments.

- 1978
- 16-bit
- 1MB (64KB) memory
- 5 Mhz
- 2 stages
- Data width 16 bits
- Address width 20 bits
- 40 pins
- 3 um process
- 29,000 transistors / 33 mm^2

80186

- 1982
- 16-bit
- 1MB (64KB) memory
- 25 Mhz
- 2 stages

Data width	16 bits
Address width	20 bits

- 68 pins
- 3 um process
- 55,000 transistors / 60 mm^2

80286

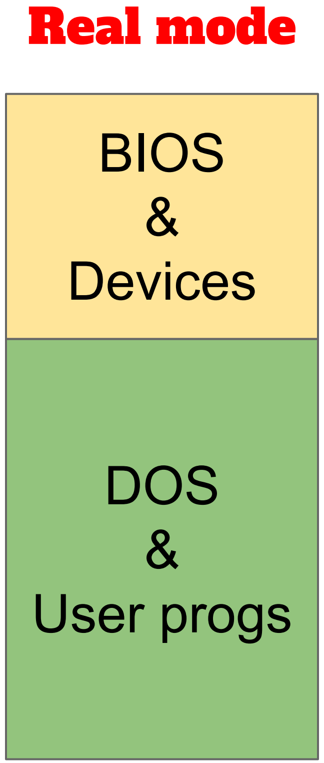
- 1982
- 16-bit
- 16MB (64KB) memory
- 25 Mhz
- 3 stages

Data width	16 bits
Address width	24 bits

- 68 pins
- 1.5 um process
- 134,000 transistors / 49 mm^2

Additional processor modes

- Introduced in 80286
- Problem — insufficient memory
  - 20-bit address ⇒ 1MB
  - 384KB reserved for BIOS & device memory
  - 640KB for operating system and applications
  - Memory is now cheaper
  - Applications have become larger
  - But 8086/80186 cannot use more memory
- Also, all applications share memory
  - No protection



Real mode segmentation

1MB ranges from

- 00000$_{16}$ **to**
- FFFFF$_{16}$

Segment address is 5 hex digits

But 16-bits are 4 digits

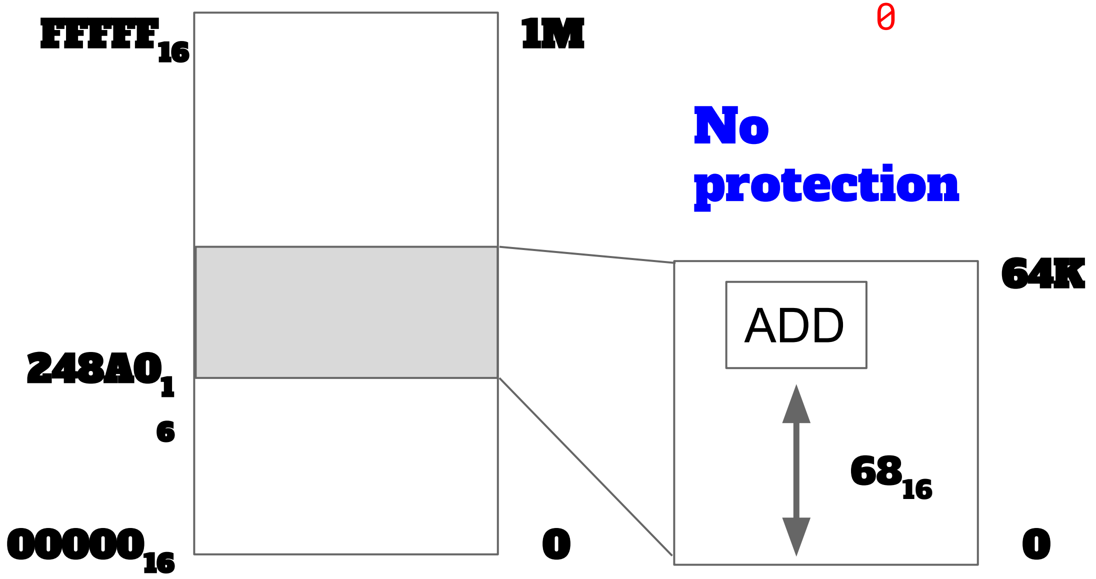
Use these as 4 high-order digits
Assume a low-order 0 digit

16-bit segment register holds 4 hex digits

x	x	x	x	0

processor implicitly adds 0

No protection



Protected mode segmentation

16MB ranges from

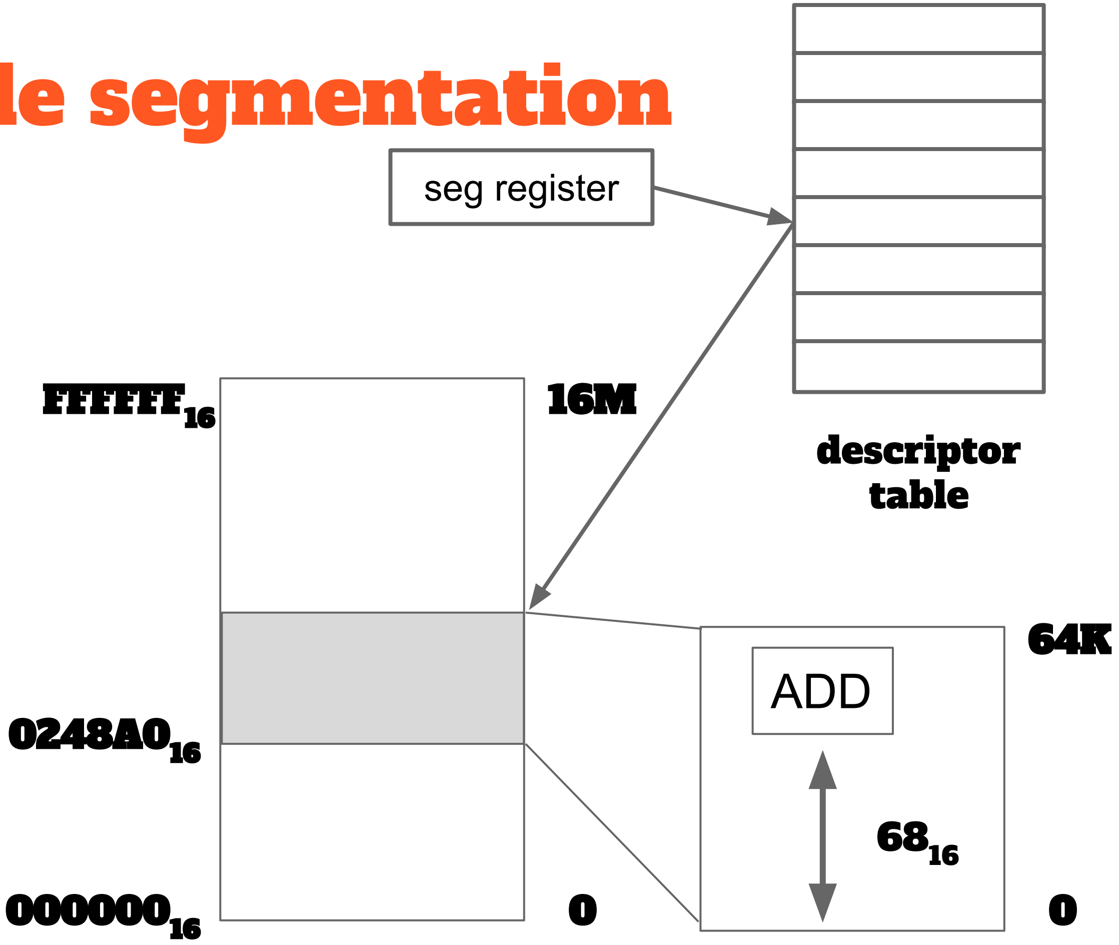
- 000000_{16} **to**
- $FFFFFF_{16}$

Segment address is 6 hex digits

Segment register is 16 bits

Doesn't directly determine address

Instead, references a segment *descriptor table*



Protected mode

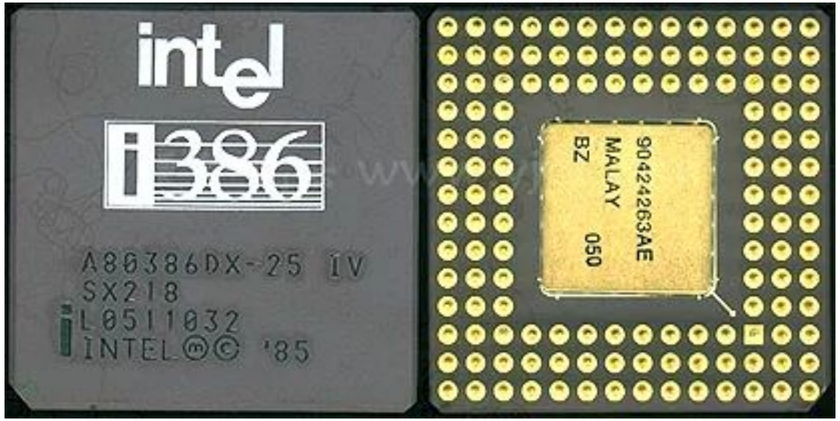
- Can now access 16MB
- Still via only four 64KB segments at a time
- More expensive to change segments
- Privilege levels — four “rings”
    - Ring 0 — for OS, can access all memory
    - Ring 3 — for applications (restricted to its own memory)
    - Other rings not used
- Some instructions designated as *privileged*
    - Can only be executed from ring 0

In 80286

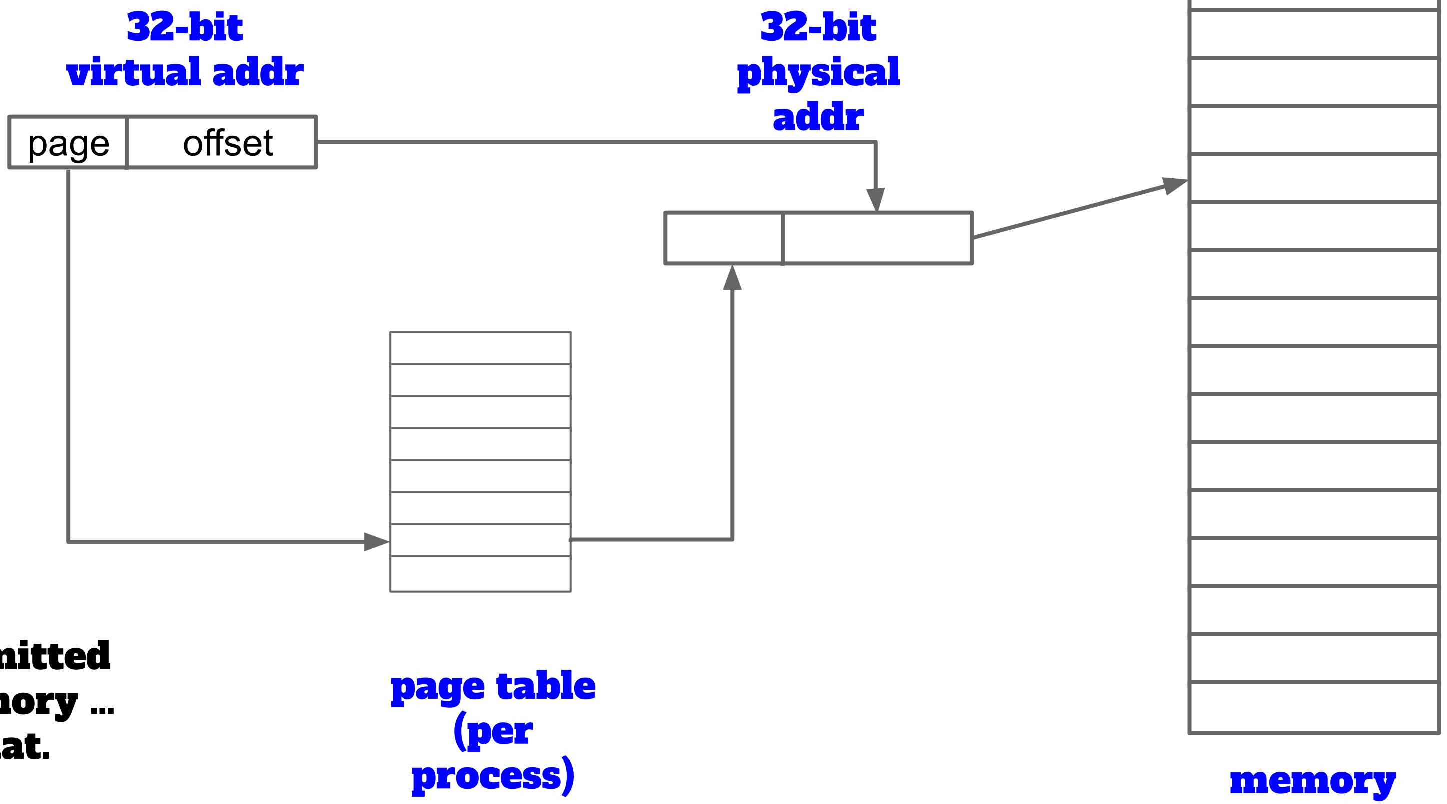
- Protected mode not widely used
- Too much trouble
- Still only four 64KB segments

80386

- 1985
- 32-bit (finally ☺️)
- 4GB (4GB) memory
- 33 Mhz
- 3 stages
- Data width 32 bits (386SX: 16 bit)
- Address width 32 bits (386SX: 24 bits)
- 132 pins
- 1.5 um process
- 275,000 transistors / 104 mm^2

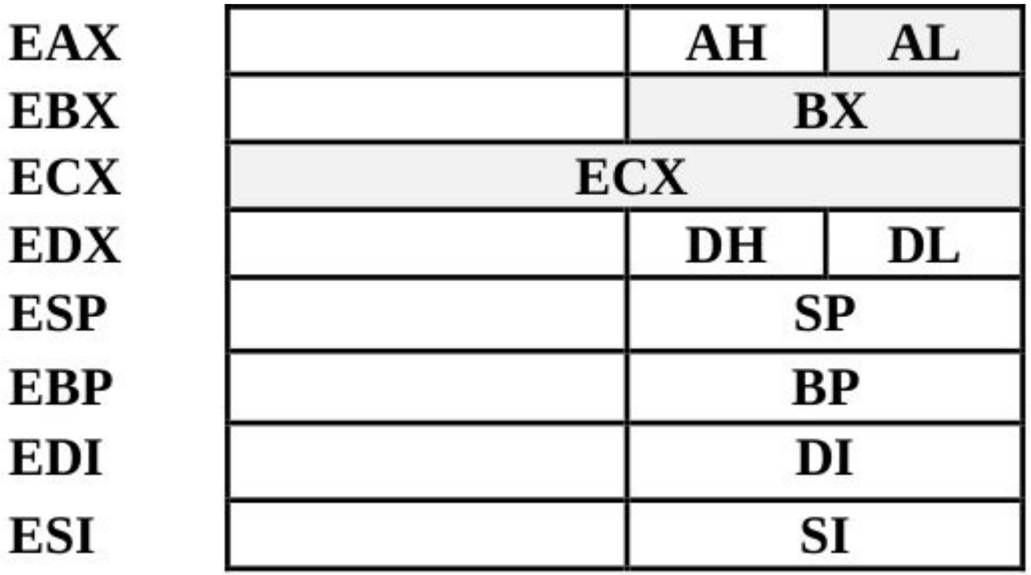


Demand paging / virtual memory



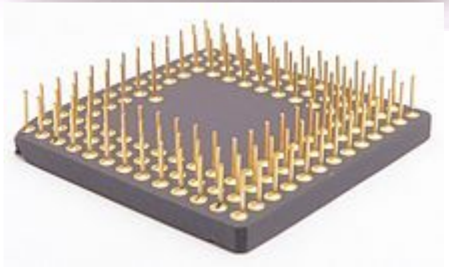
The 80386 permitted segmented memory … but we'll skip that.

80386 has 32-bit registers



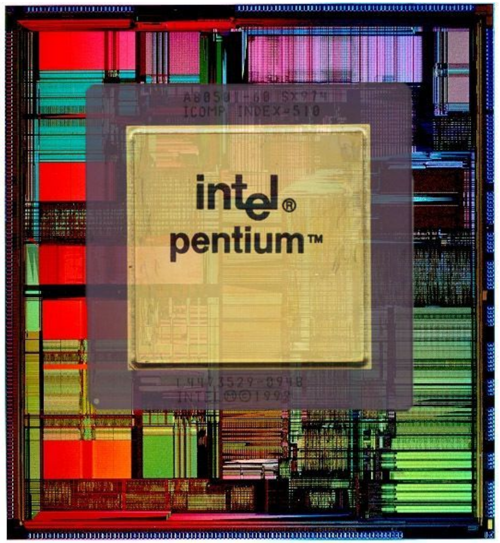
80486

- 1989
- 32-bit
- 4GB (4GB) memory
- 100 Mhz
- 5 stages
- Data width 32 bits
- Address width 32 bits
- 168 pins
- 1 um process
- 1,180,235 transistors / 173 mm^2



Pentium (P5)

- 1993
- 32-bit
- 4GB (4GB) memory
- 200 Mhz
- 5 stages
- Data width 32 bits
- Address width 32 bits
- 273 pins
- 350 nm process
- 3,100,000 transistors / 294 mm^2



Pentium description

The advanced Dynamic Execution engine is a deep, out-of-order, speculative execution engine.

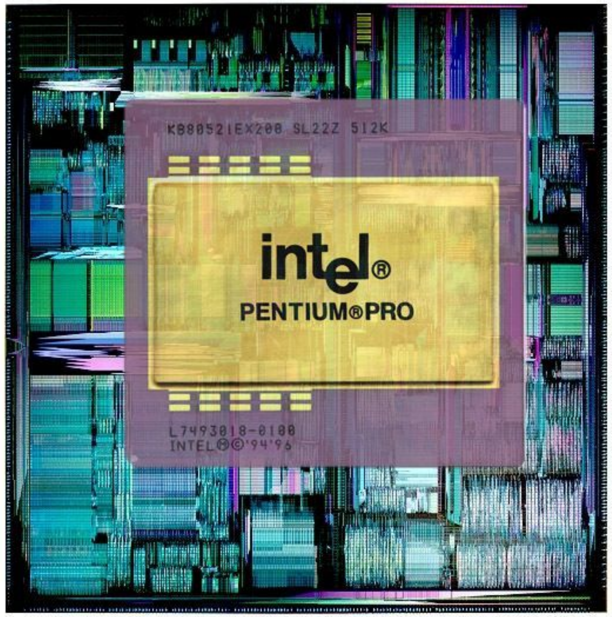
- deep
- out-of-order
- speculative execution

ILP — superscalar, multiple functional units
(later)

branch prediction

Pentium Pro (P6)

- 1995
- 32-bit
- 4GB (4GB) memory
- 450 Mhz
- 14 stages
- Data width 64 bits
- Address width 32 bits
- 387 pins
- 350 nm process
- 5,500,000 transistors / 307 mm^2



Pentium 4

- 2000
- 32-bit
- 64GB (4GB) memory
- 2 Ghz
- 20 stages
- Data width 64 bits
- Address width 36 bits
- 423 pins
- 180 nm process
- 42,000,000 transistors / 217 mm^2

Core 2 Duo

- 2006
- 64-bit
- 16-EB* GB memory
- 3 Ghz
- 12 stages / 2 cores
- Data width 64 bits
- Address width 64* bits

* not really

- 423 pins
- 65 nm process
- 291,000,000 transistors / 143 mm^2

How big is a 64-bit address space?

- $2^{60} \sim= 10^{18}$
- Quintillion
- 2^{64} = 18,446,744,073,709,551,616
- 128GB DIMM (2^{37} bytes for about $960)
- $2^{64} / 2^{37} = 2^{27}$ = 134 million 128GB DIMMs = $129 billion

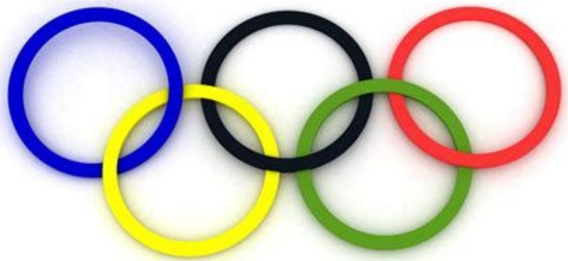
2 x 128GB DDR4 -- $1,922.95 (free shipping)

How big is 64 bits?

2^0	10^0	one	1 second
2^{10}	10^3	kilo thousand	17 minutes

How big is 64 bits?

● 2^{0}	10^{0}	one	1 second
● 2^{10}	10^{3}	kilo thousand	17 minutes
● 2^{20}	10^{6}	mega million	12 days



How big is 64 bits?

- 2^0 10^0 one 1 second
- 2^{10} 10^3 kilo thousand 17 minutes
- 2^{20} 10^6 mega million 12 days
- 2^{30} 10^9 giga billion 32 years

How big is 64 bits?

2^{0}	10^{0}	one	1 second
2^{10}	10^{3}	kilo thousand	17 minutes
2^{20}	10^{6}	mega million	12 days
2^{30}	10^{9}	giga billion	32 years
2^{40}	10^{12}	tera trillion	317 centuries

Christ's birth	**0.006×2^{40}**
Stonehenge	**0.013×2^{40}**
Last Neanderthal	**1.26×2^{40}**

How big is 64 bits?

2^{0}	10^{0}	one		1 second
2^{10}	10^{3}	kilo	thousand	17 minutes
2^{20}	10^{6}	mega	million	12 days
2^{30}	10^{9}	giga	billion	32 years
2^{40}	10^{12}	tera	trillion	317 centuries
2^{50}	10^{15}	peta	quadrillion	31,700 millenia

- **Humans appeared 0.006 x 2^{50}**
- **Dinosaurs extinct 2.1 x 2^{50}**
- **National debt $25 x 2^{50}**

How big is 64 bits?

2^{0}	10^{0}	one	1 second
2^{10}	10^{3}	kilo thousand	17 minutes
2^{20}	10^{6}	mega million	12 days
2^{30}	10^{9}	giga billion	32 years
2^{40}	10^{12}	tera trillion	317 centuries
2^{50}	10^{15}	peta quadrillion	31,700 millenia
2^{60}	10^{18}	exa quintillion	31,700,000 millenia

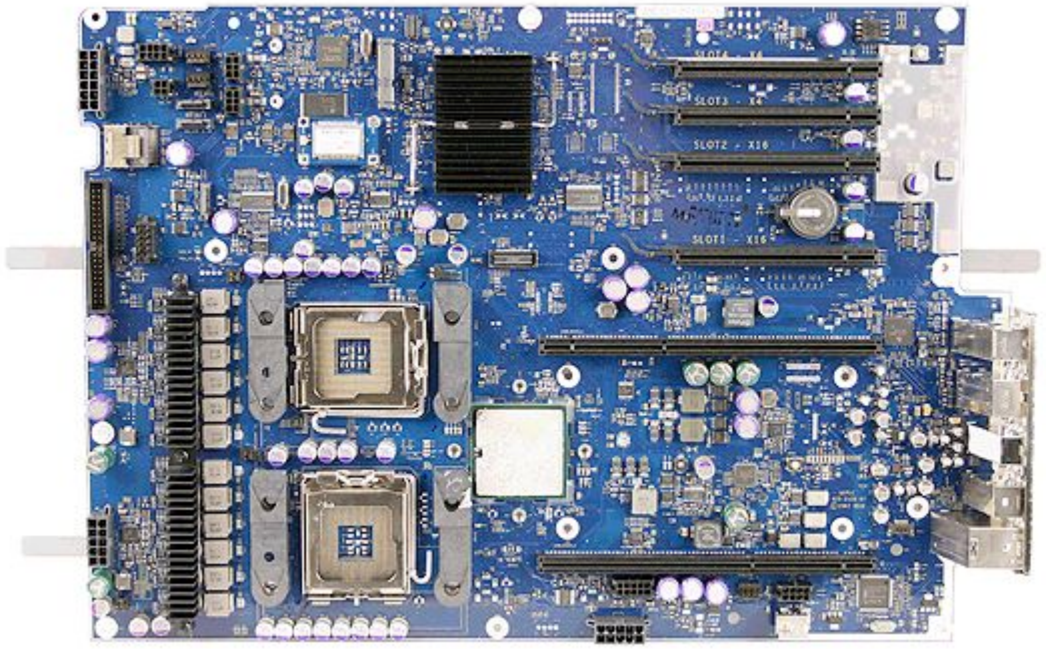
Age of earth	**0.158×2^{60}**
Big bang	**0.435×2^{60}**

How big is 64 bits?

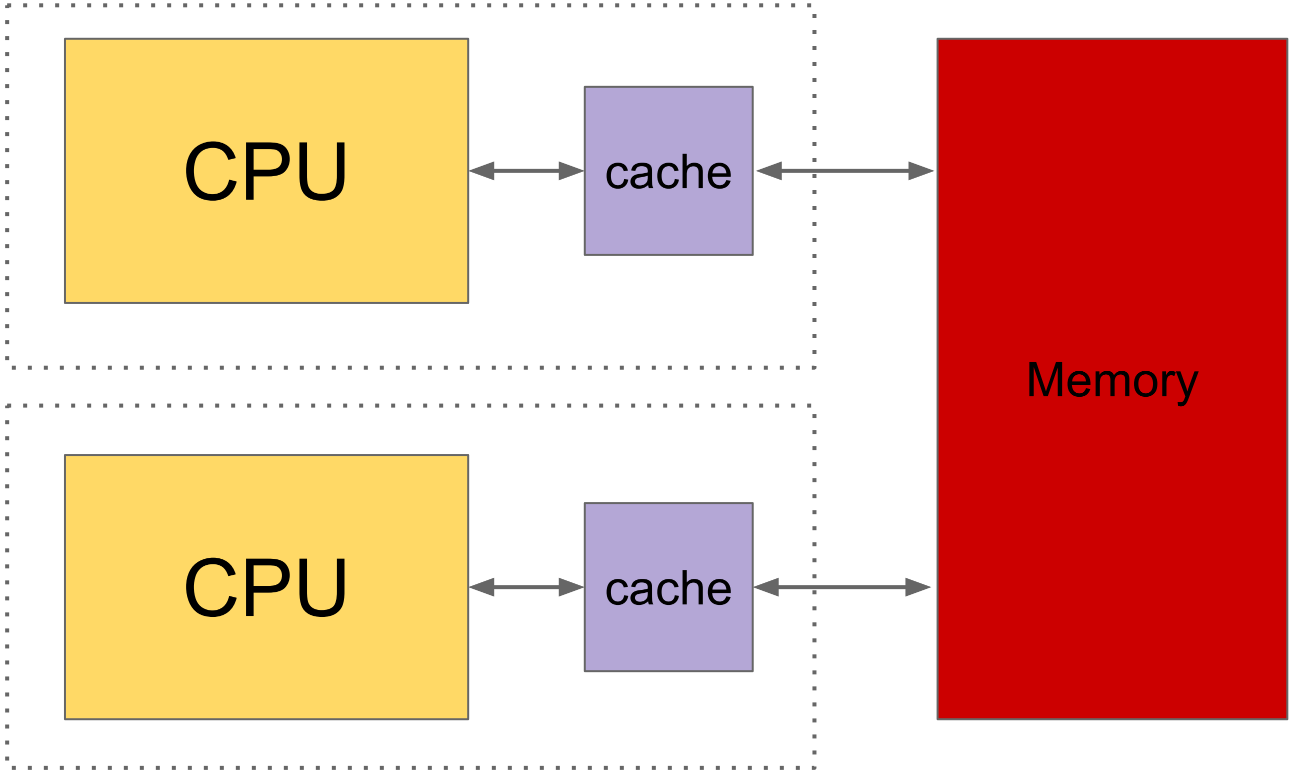
2^{0}	10^{0}	one		1 second
2^{10}	10^{3}	kilo	thousand	17 minutes
2^{20}	10^{6}	mega	million	12 days
2^{30}	10^{9}	giga	billion	32 years
2^{40}	10^{12}	tera	trillion	317 centuries
2^{50}	10^{15}	peta	quadrillion	31,700 millenia
2^{60}	10^{18}	exa	quintillion	31,700,000 millenia
2^{64}	$16x10^{18}$			507,000,000 millenia

Monikers					Description
64-bit	32-bit	16-bit	8 high bits of lower 16 bits	8-bit	
RAX	EAX	AX	AH	AL	Accumulator
RBX	EBX	BX	BH	BL	Base
RCX	ECX	CX	CH	CL	Counter
RDX	EDX	DX	DH	DL	Data (commonly extends the A register)
RSI	ESI	SI	N/A	SIL	Source index for string operations
RDI	EDI	DI	N/A	DIL	Destination index for string operations
RSP	ESP	SP	N/A	SPL	Stack Pointer
RBP	EBP	BP	N/A	BPL	Base Pointer (meant for stack frames)
R8	R8D	R8W	N/A	R8B	General purpose
R9	R9D	R9W	N/A	R9B	General purpose
R10	R10D	R10W	N/A	R10B	General purpose
R11	R11D	R11W	N/A	R11B	General purpose
R12	R12D	R12W	N/A	R12B	General purpose
R13	R13D	R13W	N/A	R13B	General purpose
R14	R14D	R14W	N/A	R14B	General purpose
R15	R15D	R15W	N/A	R15B	General purpose

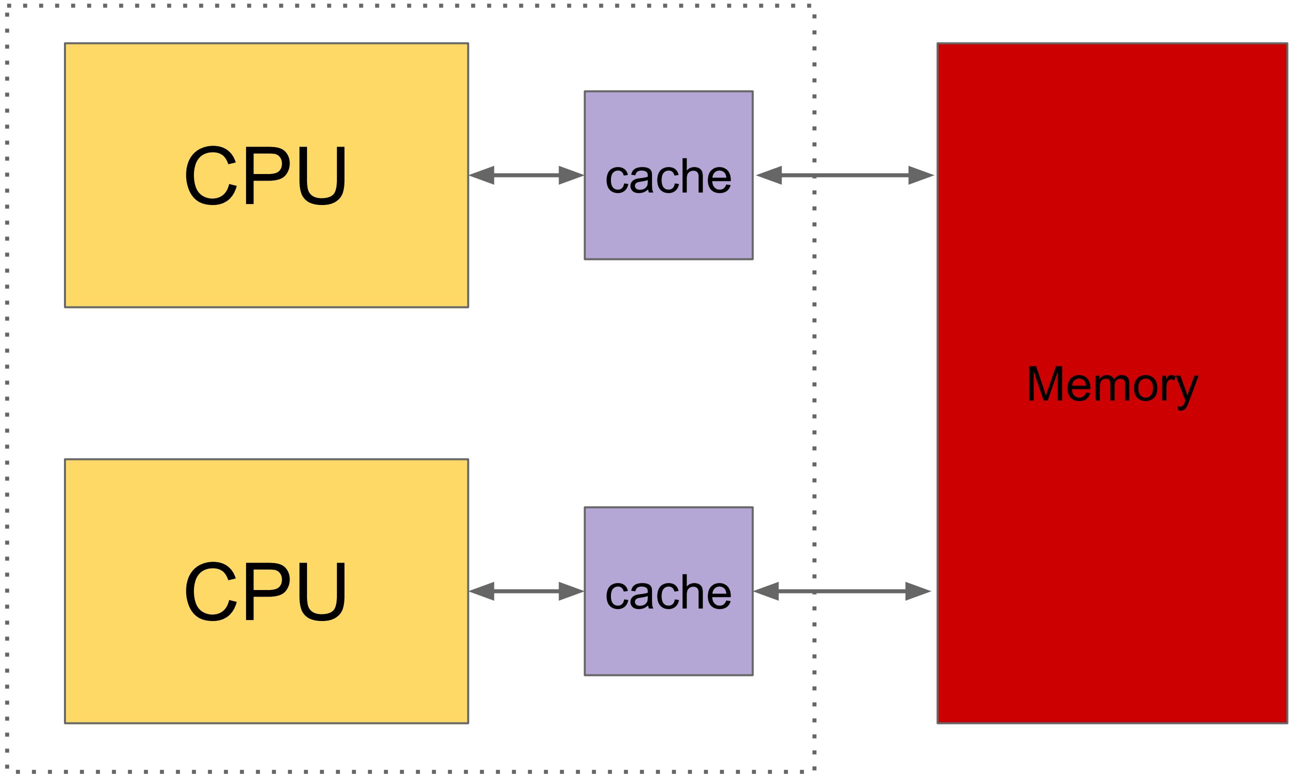
Symmetric multiprocessing (SMP)



Symmetric multiprocessing (SMP)



Multiple cores

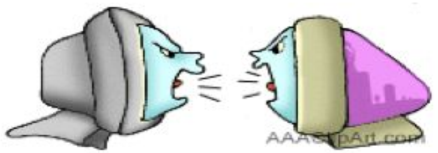


Multiple cores — advantages

- Need to do something with extra transistors
- Parallel speedup
    - Limited single-threaded (application) improvement
- Reduced cost and overhead vs SMP

CISC vs RISC

- **Lots of function**
- **Complicated decode**
- **Slow**



CISC - 1960 ... 1990
Microcoding

RISC - 1990
Hardwired

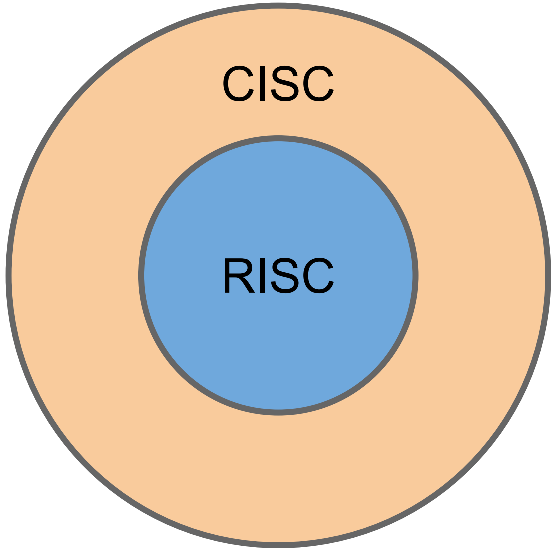
- **Lean hardware**
- **Simple decode**
- **Fast**

- **Many ways to access data (address modes)**
- **Machine Instructions**
  - **complicated and fancy**
  - **variable size**
  - **variable execution time**

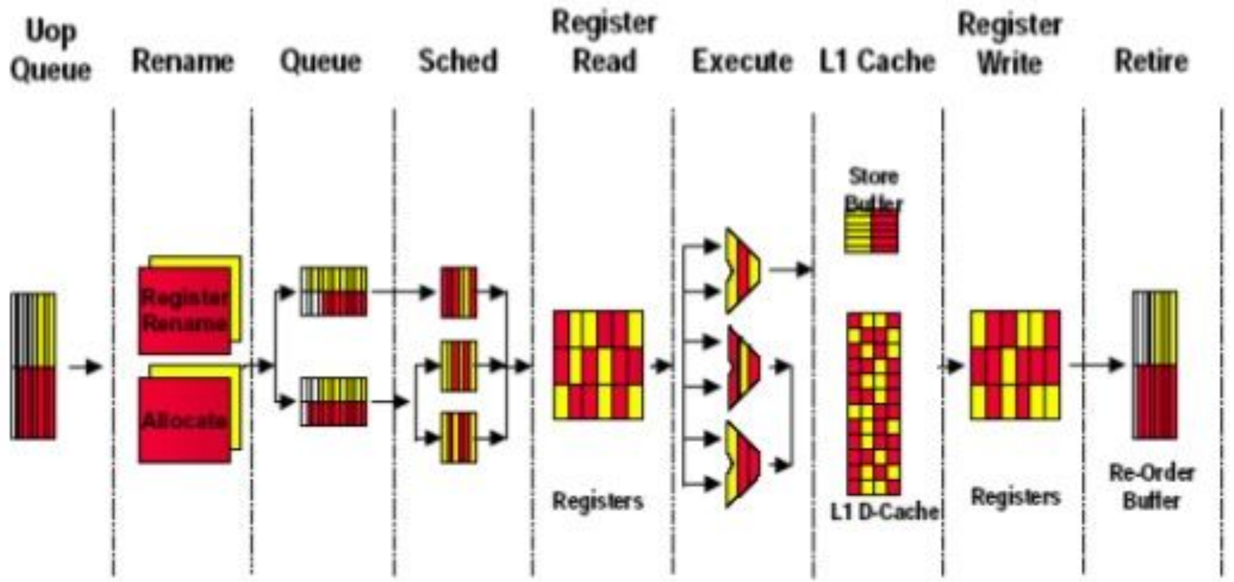
- **Only simple load and stores from memory**
- **Machine Instructions**
  - **only basic operations**
  - **fixed size**
  - **fixed execution times**

RISC core

- RISC more efficient
  - Load/store architecture
  - Easier to pipeline
  - Can extract more ILP
- Problem
  - Must maintain backwards compatibility
- Solution
  - *micro* operations (uops)
  - Translate x86 ISA into uops
  - Similar to microcode but hardcoded



Out of order


Uop Queue
Rename
Queue
Sched
Register Read
Execute
L1 Cache
Register Write
Retire
Register Rename
Allocate
Store Buffer
Registers
L1 D-Cache
Registers
Re-Order Buffer

MELTDOWN
SPECTRE

References

- https://www.pcgamer.com/a-brief-history-of-cpus-31-awesome-years-of-x86/
- https://www.tomshardware.com/picturestory/710-history-of-intel-cpus.html
- https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectures
- https://en.wikipedia.org/wiki/Comparison_of_Intel_processors
- https://en.wikipedia.org/wiki/CPU_socket
- http://www.theinfolist.com/php/SummaryGet.php?FindGo=Transistor%20Count#Microprocessors
- https://www.slideshare.net/am_sharifian/intel-hyper-threading-technology