A Quick Primer on Computer Architecture

ComS 252 — Iowa State University

Andrew Miner

SI multiplier prefixes — small

- SI: International System of Units

SI multiplier prefixes — small

- SI: International System of Units
  - Shouldn't it be IS? or ISU?

SI multiplier prefixes — small

- SI: International System of Units
  - Shouldn't it be IS? or ISU?
  - SI abbreviation is French: *Système International d'unités*
  - Basically, "the metric system"

SI multiplier prefixes — small

- SI: International System of Units
  - Shouldn't it be IS? or ISU?
  - SI abbreviation is French: *Système International d'unités*
  - Basically, "the metric system"
- Important "small" prefixes in computing:

Prefix	**Sym.**	**Multiplier**
milli	m	10^{-3}
micro	μ	10^{-6}
nano	n	10^{-9}
pico	p	10^{-12}

- Note: all symbols are small letters
- Typical for times, physical sizes
  - 45 nm CPU (45 nanometers: refers to "feature" size on CPU)
  - 15 μs to send a message

SI multiplier prefixes — large

- Important "large" prefixes in computing:

Prefix	Sym.	Multiplier
kilo	k	10^3
mega	M	10^6
giga	G	10^9
tera	T	10^{12}
peta	P	10^{15}
exa	E	10^{18}
zetta	Z	10^{21}

- Note: most symbols are capital letters
- These are used for capacities, and frequencies
  - 1 Tb drive (1 Terabyte: 1 trillion bytes)
  - 2 Ghz clock (2 Gigahertz: 2 billion cycles per second)

Binary notation

- Humans write numbers in *decimal* notation:
  - Ten possibilities for each digit (0, 1, . . . , 8, 9)
  - $507 \quad = \quad 5 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0$
  - How many different numbers can n digits represent?

Binary notation

- Humans write numbers in *decimal* notation:
  - Ten possibilities for each digit (0, 1, . . . , 8, 9)
  - $507 \quad = \quad 5 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0$
  - How many different numbers can n digits represent? 10^n

Binary notation

- Humans write numbers in *decimal* notation:
  - Ten possibilities for each digit (0, 1, . . . , 8, 9)
  - $507 = 5 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0$
  - How many different numbers can n digits represent? 10^n
- No fundamental reason that the "base" needs to be 10
- Computers represent numbers in *binary* notation:
  - Two possibilities for each digit (0, 1)
  - $507_{10} = 1{\cdot}2^8{+}1{\cdot}2^7{+}1{\cdot}2^6{+}1{\cdot}2^5{+}1{\cdot}2^4{+}1{\cdot}2^3{+}0{\cdot}2^2{+}1{\cdot}2^1{+}1{\cdot}2^0$
    $507_{10} = 111111011_2$
  - n binary digits (bits) can represent

Binary notation

- Humans write numbers in *decimal* notation:
  - Ten possibilities for each digit (0, 1, . . . , 8, 9)
  - $507 = 5 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0$
  - How many different numbers can n digits represent? 10^n
- No fundamental reason that the "base" needs to be 10
- Computers represent numbers in *binary* notation:
  - Two possibilities for each digit (0, 1)
  - $507_{10} = 1{\cdot}2^8{+}1{\cdot}2^7{+}1{\cdot}2^6{+}1{\cdot}2^5{+}1{\cdot}2^4{+}1{\cdot}2^3{+}0{\cdot}2^2{+}1{\cdot}2^1{+}1{\cdot}2^0$
    $507_{10} = 111111011_2$
  - n binary digits (bits) can represent 2^n different numbers

Hexadecimal

- Base 16: digits are (0, 1, . . . , 9, a, b, c, d, e, f)
- 1 hexadeximal digit equals four bits
- Easy to convert between hex and binary

$0_{16} = 0000_2$ $1_{16} = 0001_2$ $2_{16} = 0010_2$ $3_{16} = 0011_2$
$4_{16} = 0100_2$ $5_{16} = 0101_2$ $6_{16} = 0110_2$ $7_{16} = 0111_2$
$8_{16} = 1000_2$ $9_{16} = 1001_2$ $a_{16} = 1010_2$ $b_{16} = 1011_2$
$c_{16} = 1100_2$ $d_{16} = 1101_2$ $e_{16} = 1110_2$ $f_{16} = 1111_2$

- $507_{10} = 1,1111,1011_2 = 1fb_{16}$
- Often written as `0x1fb` or `x1fb`
- MAC addresses: 12 hex digits
  - Much easier to write than 48 bits

Octal

- Base 8: digits are (0, 1, . . . , 7)
- 1 octal digit equals three bits — also easy to convert

  $0_8 = 000_2 \quad 1_8 = 001_2 \quad 2_8 = 010_2 \quad 3_8 = 011_2$

  $4_8 = 100_2 \quad 5_8 = 101_2 \quad 6_8 = 110_2 \quad 7_8 = 111_2$
- $507_{10} = 111,111,011_2 = 773_8$
- More obscure than hexadeximal, but still used

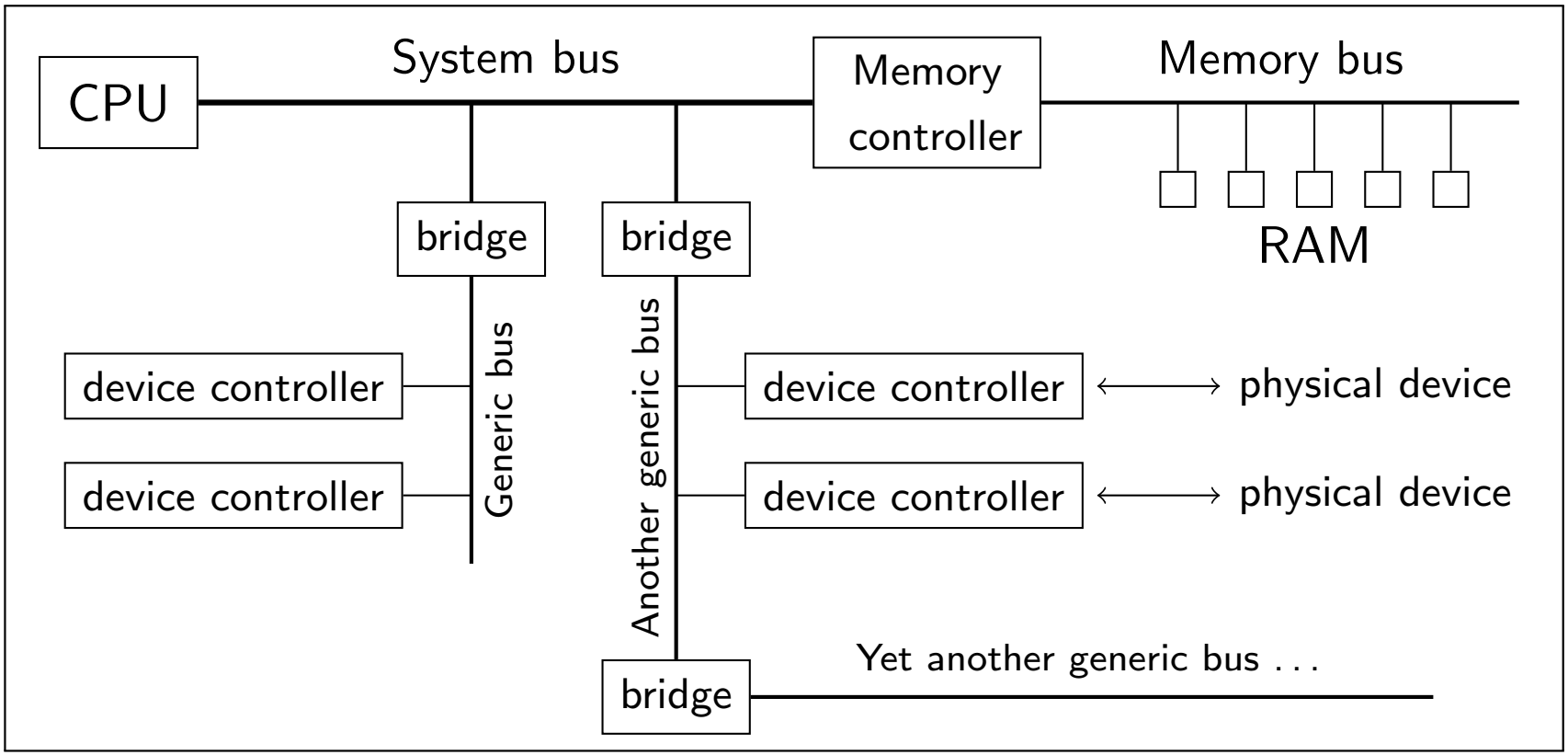
Binary prefixes

- 10^n is a nice round number in decimal notation
- 2^n is a nice round number in binary notation
- In computing, it is often easier to group 2^n things
  - E.g., $2^{10} = 1024$ bytes is more natural than 1000 bytes
- Slightly different prefixes used:

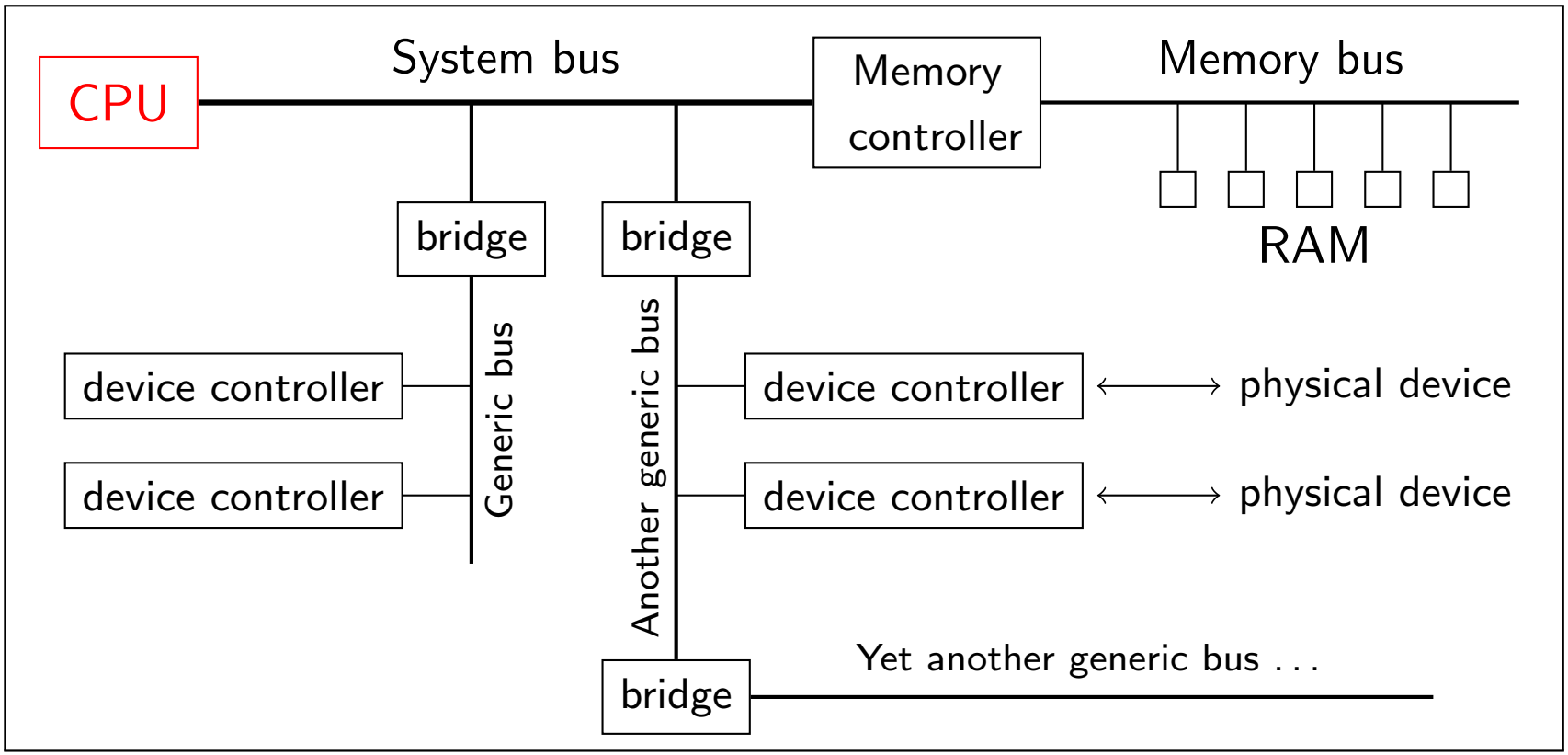
Confusing	**Preferred**	**Sym.**	**Multiplier**
kilo	Kibi	Ki	$2^{10} = 1024^1$
mega	Mebi	Mi	$2^{20} = 1024^2$
giga	Gibi	Gi	$2^{30} = 1024^3$
	Tebi	Ti	$2^{40} = 1024^4$

- Can lead to issues
  - Manufacturer says 1Tb drive (1,000,000,000,000 bytes)
  - Disk utility says 931 Gib drive: $931 \times 2^{30} \approx 10^{12}$ (even less after formatting)

Simple, abstract view of a computer

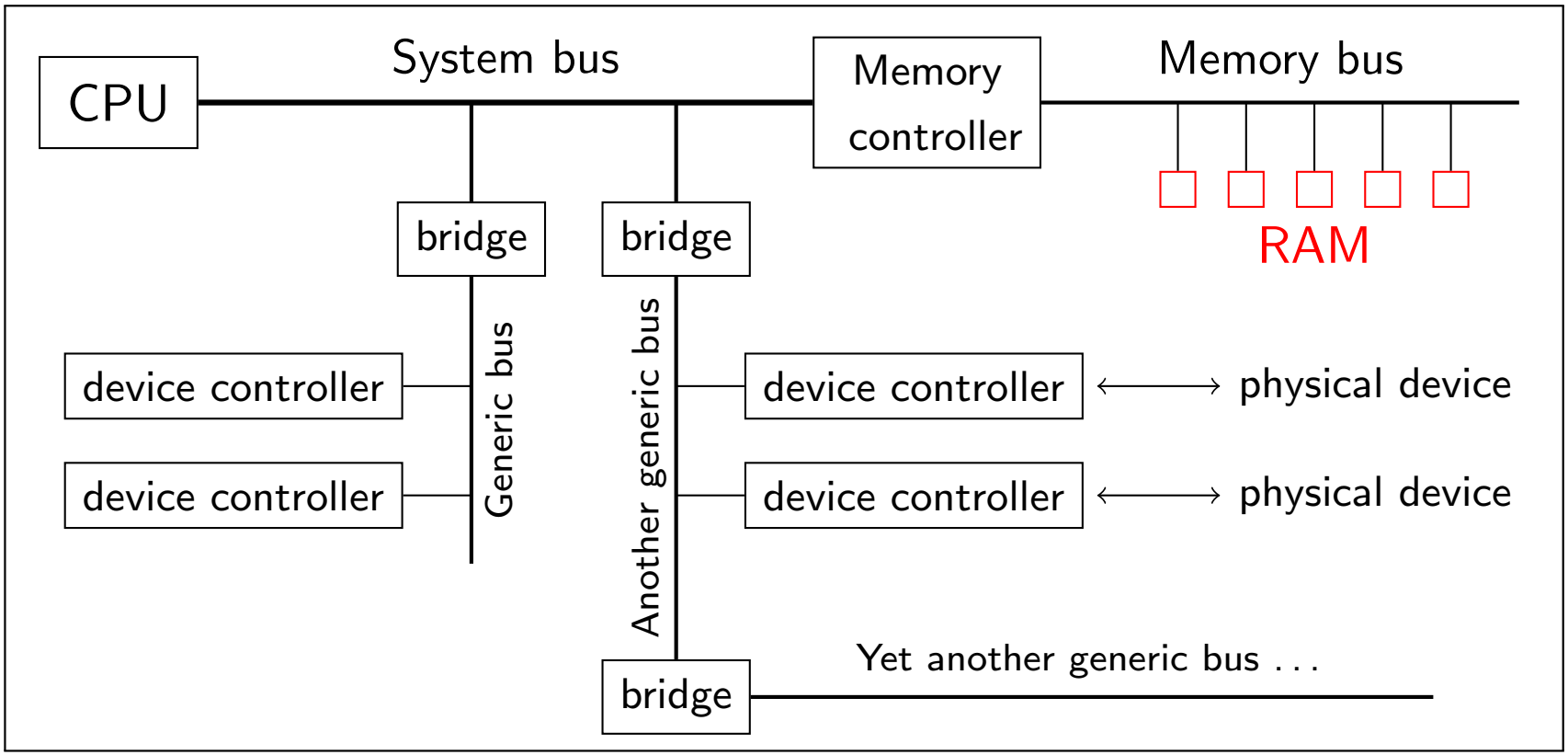


CPU: Central Processing Unit



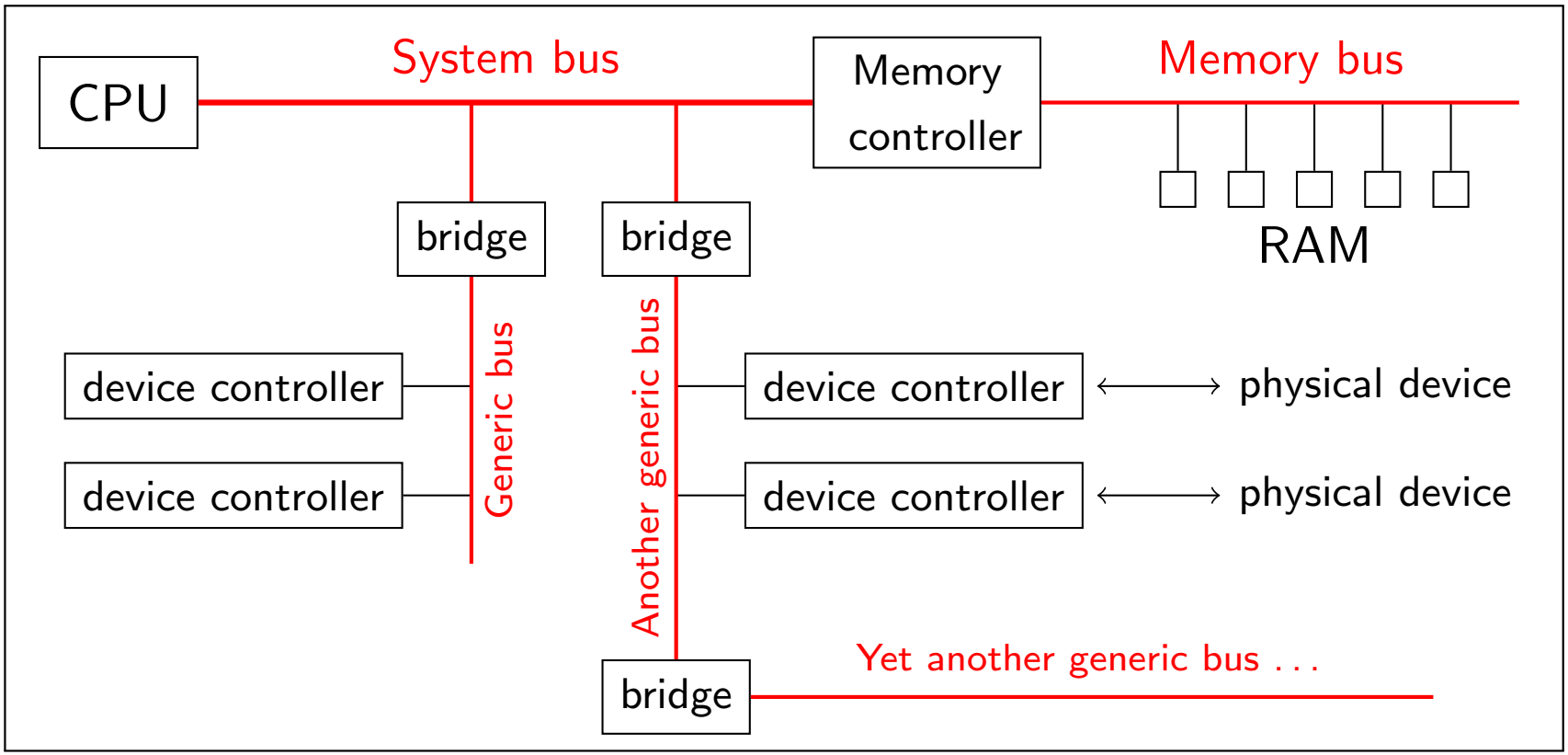
- Executes machine language instructions

RAM: Random Access Memory



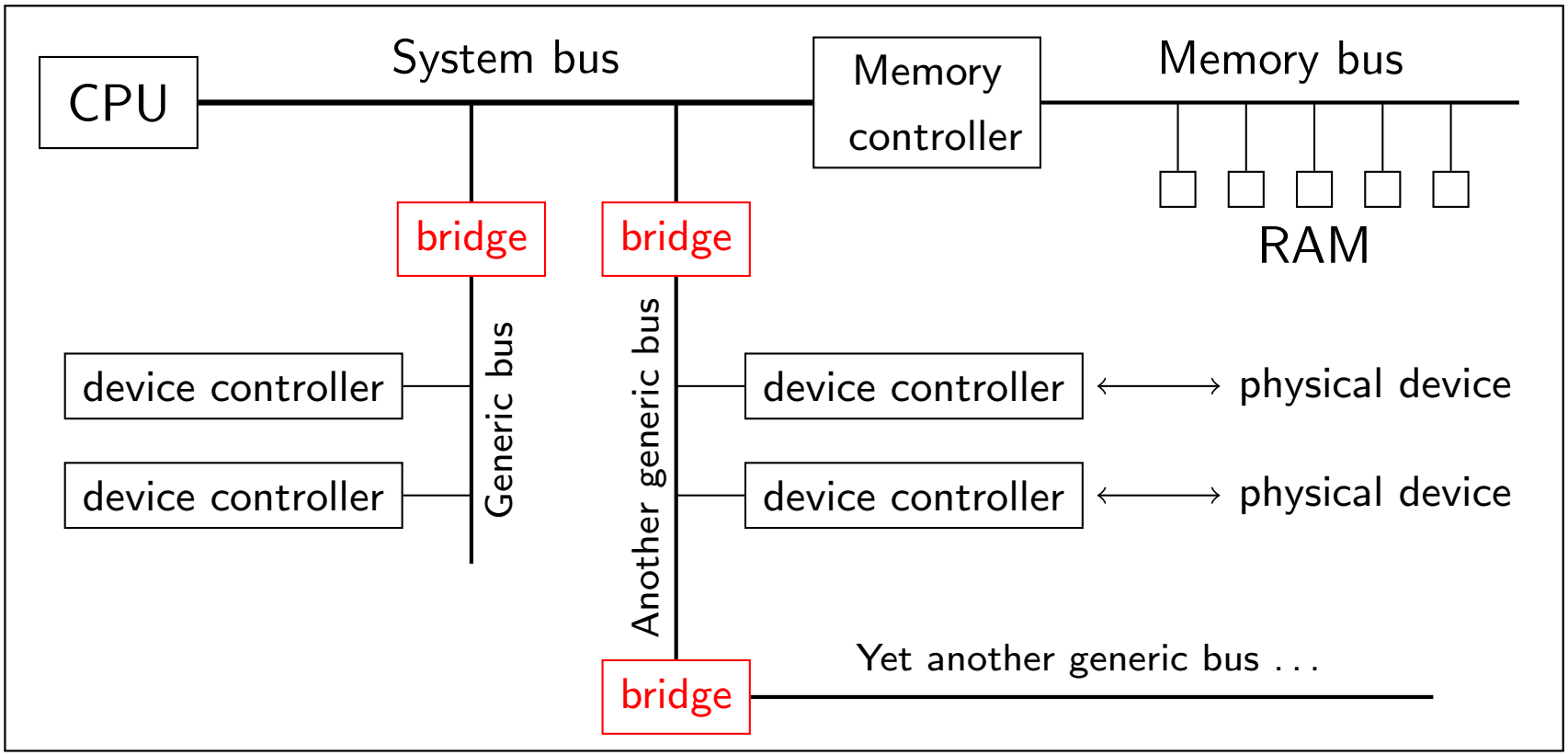
- Stores data
- Unit of storage: *byte* (8 bits)
- Each byte has an integer *adress*

Buses



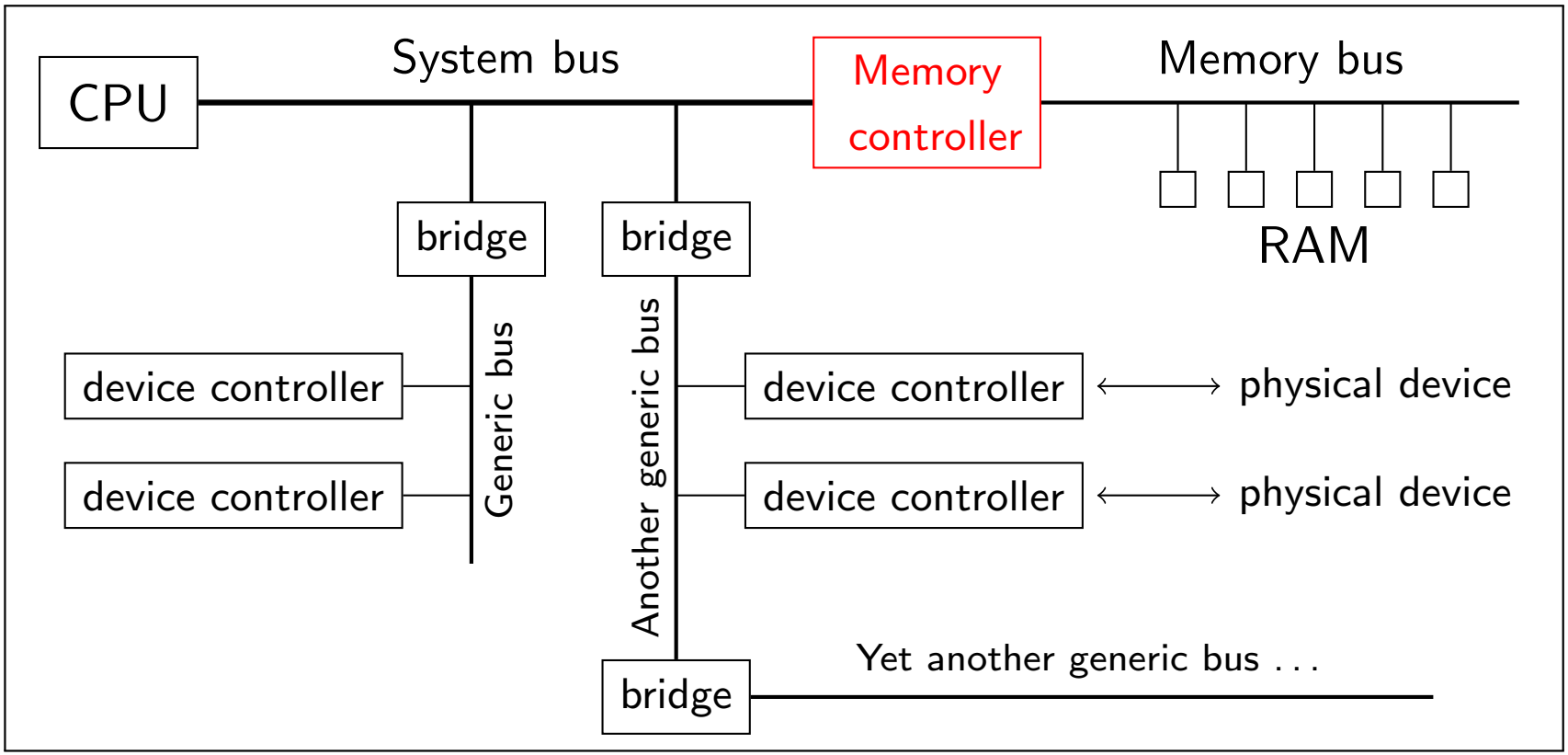
- Wires: allow data transfer
- Protocol: rules for transferring data

Bridges



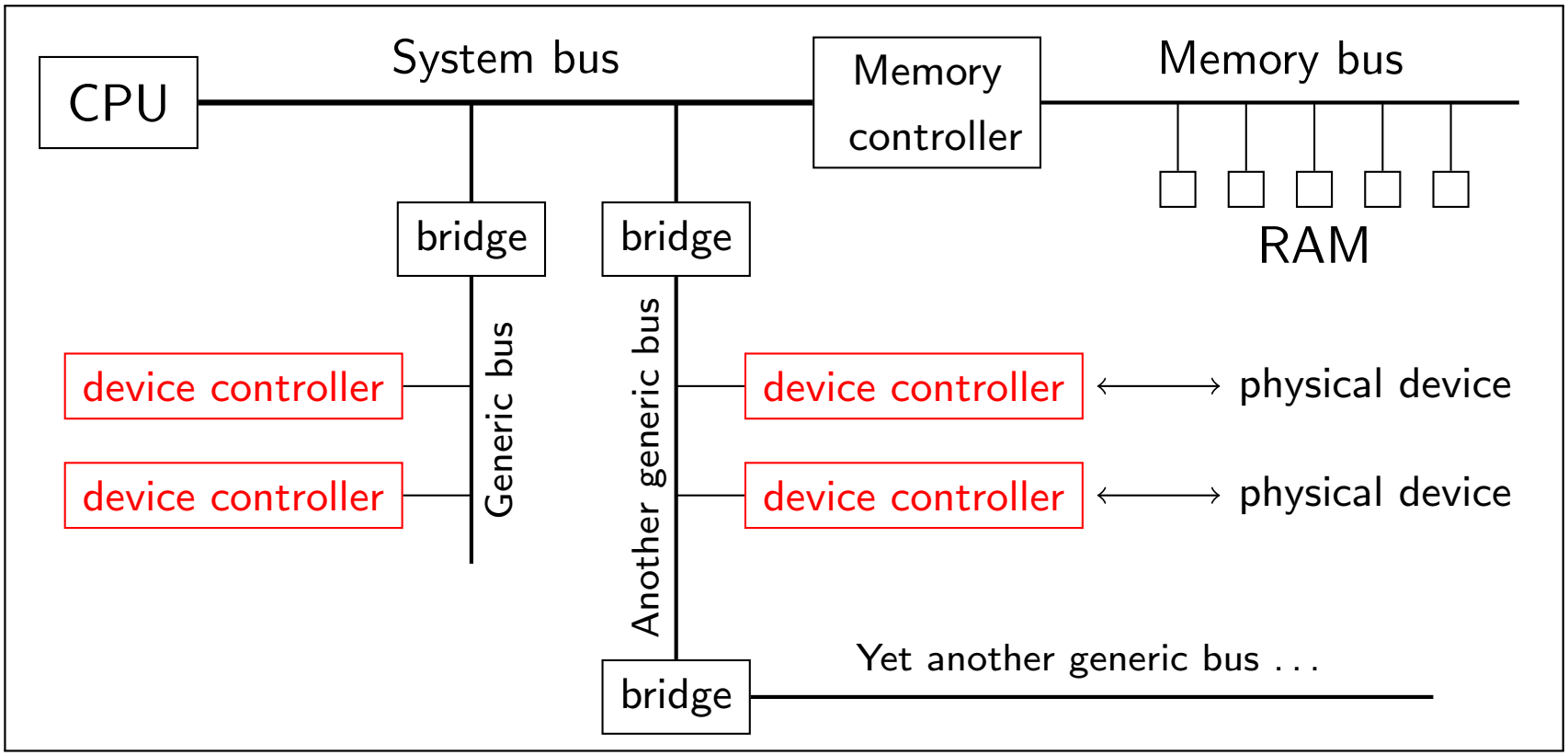
- Convert from one bus type to another

Memory controller



- Specialized bridge for RAM
- Has additional functionality (e.g., for paging)
- Details are "beyond the scope of this class"

Device controllers



- Connect to physical devices

Instructions

CPUs execute instructions in *machine language* (ML)

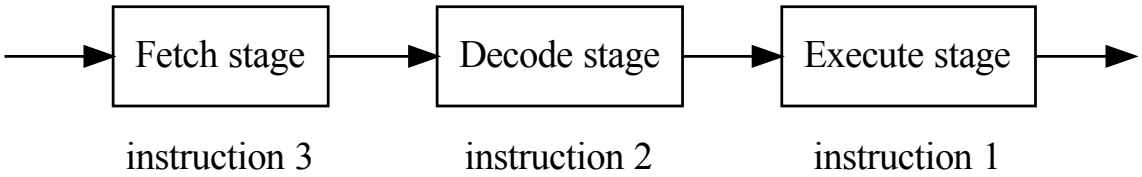
- Fairly simple, e.g.:
  - Copy integer at memory address `0x0c2b4938` into variable `b`
  - Subtract `b` from `a`
  - If `a` is zero, start executing instructions at address `0x003a247c`
- High–level languages (like C) must be translated into ML (That's what a *compiler* does)
- Instruction set: set of instructions CPU can execute
- Family of CPUs: have backward compatible instruction sets
  - E.g., Intel 8086, Intel 80286, Intel 80386
- Different CPUs typically have different instruction sets
  - ML code for Motorola 68000 family will not run on Intel 8086 family

Instruction Cycle

CPUs execute ML code in cycles, e.g.:

1. Fetch the next instruction to execute, from memory
2. Decode the instruction
   - Instructions are encoded into some number of bytes
   - Some instructions have "arguments"
     - E.g., copy into `b` from *where*?
3. Execute the instruction

Cycles are broken into stages, arranged in a pipeline:



Reality: pipelines are more complex than this

- RISC processors have 5 stage pipeline
- Pentium 3: 10 stage pipeline
- Pentium 4: 20 stage pipeline

CPU clock

- CPUs have an internal clock (like a metronome)
  - Stages are synchronized
  - Work within stages is synchronized
  - To learn why, take CprE 281
- Completely different from wall clock
- Clock rate is measured in hertz
- Speed at which a CPU can execute a program depends on
  - Instruction set
  - Pipeline layout
  - Stage design
  - Other details we have not discussed yet...
  - Clock frequency
- Comparing clock frequencies is valid only for similar CPUs
  - 16 Mhz Intel 80386 is slower than 33 Mhz Intel 80386
  - 0.867 Ghz G4 can run faster than 2 Ghz Pentium 4

Moore's Law

Moore's Law

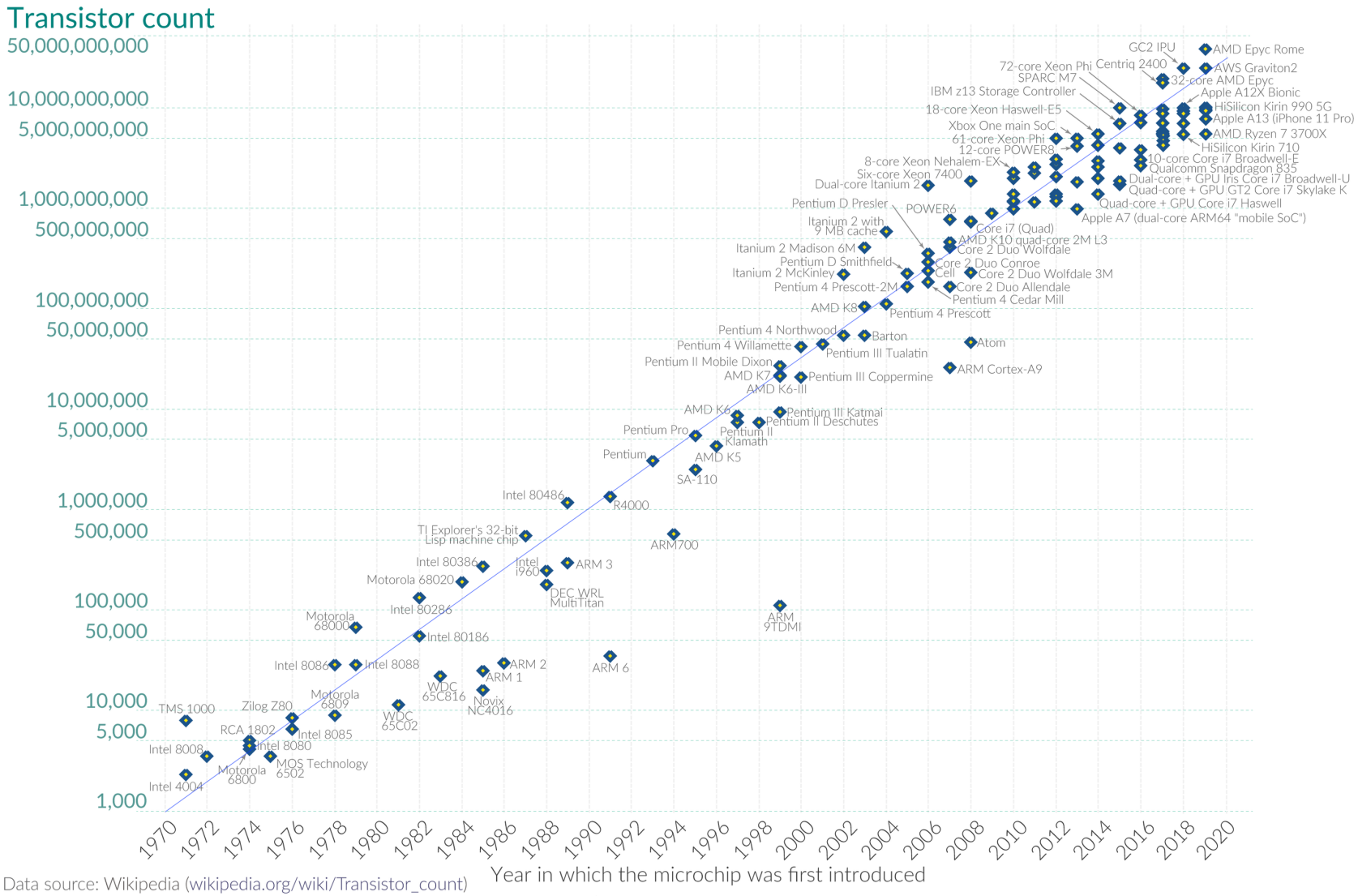
Predicts that the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years

- Not a *physical* law; rather, a "rule of thumb"
- Moore made the original prediction in 1965
  - Originally, doubled *every year*
  - Moore predicted it would hold for 10 years
  - In 1975, Moore altered it to *every two years*
  - Industry has treated Moore's law as a target
- Is the end near for Moore's law?
  - 2017: 5 nm chips announced ($\sim$ 30 billion transistors)
  - 2021: 2 nm chips announced ($\sim$ 50 billion transistors)
    - Human DNA strand is 2.5 nm in diameter

Moore's Law: The number of transistors on microchips doubles every two years

Our World in Data

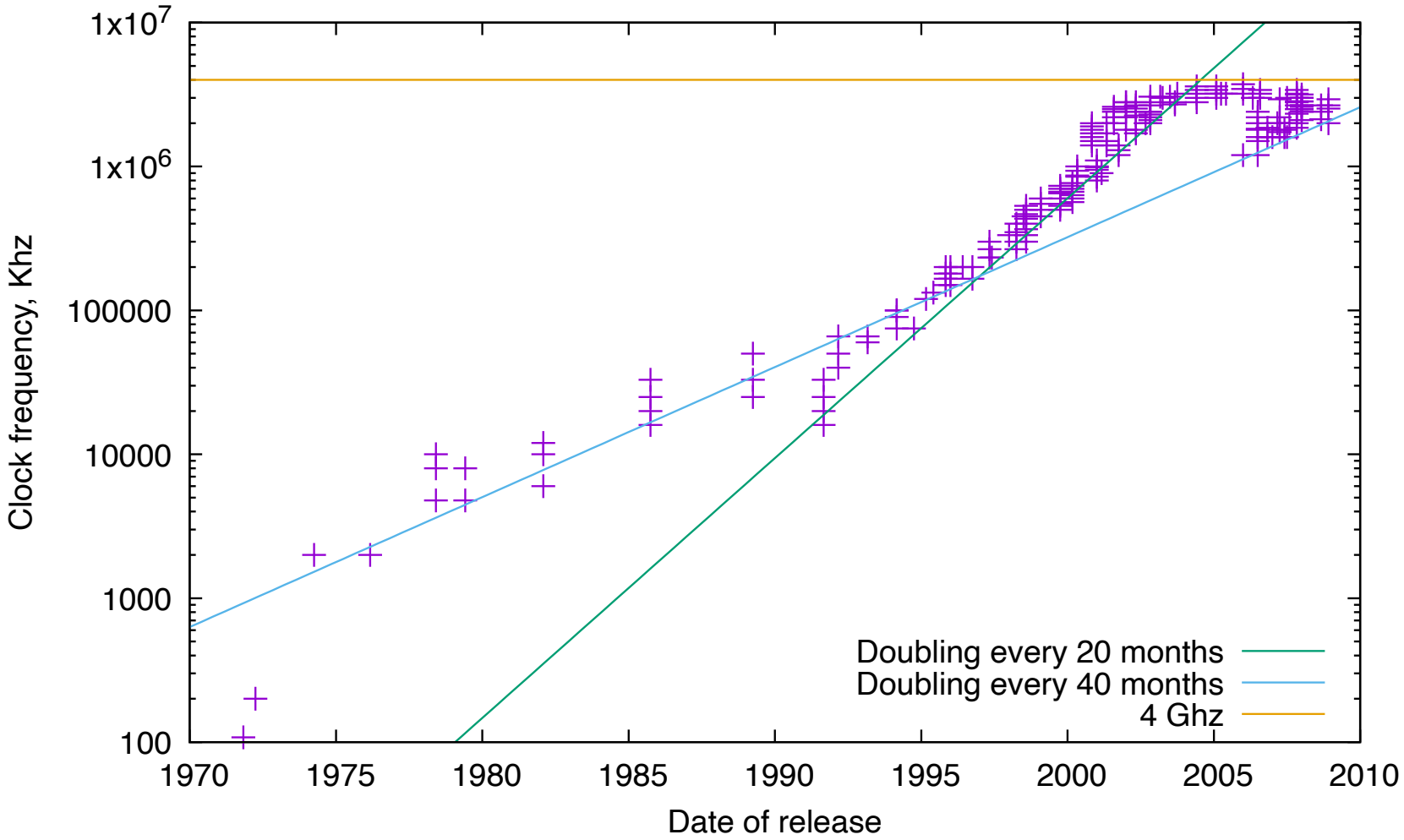
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world's largest problems.

Clock frequencies in some Intel CPUs



The shift to multi–core

- CPU clock frequencies have not increased (much) since 2005
  - Power / Heat is the main issue
- Manufacturers instead started increasing the number of *cores*
  - Effectively, each core is its own processor
- Huge gamble: nobody was asking for multi–core
- How do I keep 8 cores busy? What about 128 cores?
  - Image and video processing are easy to parallelize
  - Most people run several things at once
  - OS's will need smarter schedulers

Random Access Memory

- Stores programs (ML instructions) and data
- Volatile: loses data when power is cut
- Each byte has a unique address
  - Consecutive integers from 0 to $N - 1$
  - E.g., 1 Mb of RAM has addresses 0, 1, 2, ..., $\texttt{fffff}_{16}$
    - $\texttt{fffff}_{16} = 1111\ 1111\ 1111\ 1111\ 1111_2 = 2^{20} - 1 = 1024^2 - 1$
- Memory is divided into *pages* of 2^n bytes
  - Choice for n depends on processor architecture
  - $n = 12$ (typical) gives 4Kb pages
  - Using 2^n means address can be split at bit boundary:

E.g., address `0x8e64a2`: $\underbrace{1000\ 1110\ 0110}_{\text{page number}}\ \underbrace{0100\ 1010\ 0010}_{\text{offset within page}}$

32–bit vs. 64–bit systems

- CPUs use *registers* to store integers
  - 32–bit CPU: (most) registers are 32 bits
  - 64–bit CPU: (most) registers are 64 bits
- So what?

32–bit vs. 64–bit systems

- CPUs use *registers* to store integers
  - 32–bit CPU: (most) registers are 32 bits
  - 64–bit CPU: (most) registers are 64 bits
- Address register size determines amount of addressable RAM

32–bit vs. 64–bit systems

- CPUs use *registers* to store integers
  - 32–bit CPU: (most) registers are 32 bits
  - 64–bit CPU: (most) registers are 64 bits
- Address register size determines amount of addressable RAM
  - 20 bits (old 8086): 2^{20} bytes addressable $= 1$ Mb

32–bit vs. 64–bit systems

- CPUs use *registers* to store integers
  - 32–bit CPU: (most) registers are 32 bits
  - 64–bit CPU: (most) registers are 64 bits
- Address register size determines amount of addressable RAM
  - 20 bits (old 8086): 2^{20} bytes addressable $=$ 1 Mb
  - 32 bits: 2^{32} bytes $=$ 4 Gibi
    - 4 Gb limit on 32–bit machines

32–bit vs. 64–bit systems

- CPUs use *registers* to store integers
  - 32–bit CPU: (most) registers are 32 bits
  - 64–bit CPU: (most) registers are 64 bits
- Address register size determines amount of addressable RAM
  - 20 bits (old 8086): 2^{20} bytes addressable $=$ 1 Mb
  - 32 bits: 2^{32} bytes $=$ 4 Gibi
    - 4 Gb limit on 32–bit machines
  - 48 bits: 2^{48} bytes $=$ 256 Tebi

32–bit vs. 64–bit systems

- CPUs use *registers* to store integers
  - 32–bit CPU: (most) registers are 32 bits
  - 64–bit CPU: (most) registers are 64 bits
- Address register size determines amount of addressable RAM
  - 20 bits (old 8086): 2^{20} bytes addressable = 1 Mb
  - 32 bits: 2^{32} bytes = 4 Gibi
    - 4 Gb limit on 32–bit machines
  - 48 bits: 2^{48} bytes = 256 Tebi
  - 64 bits: 2^{64} bytes = 16 Exabytes (16 billion Gb)

32–bit vs. 64–bit systems

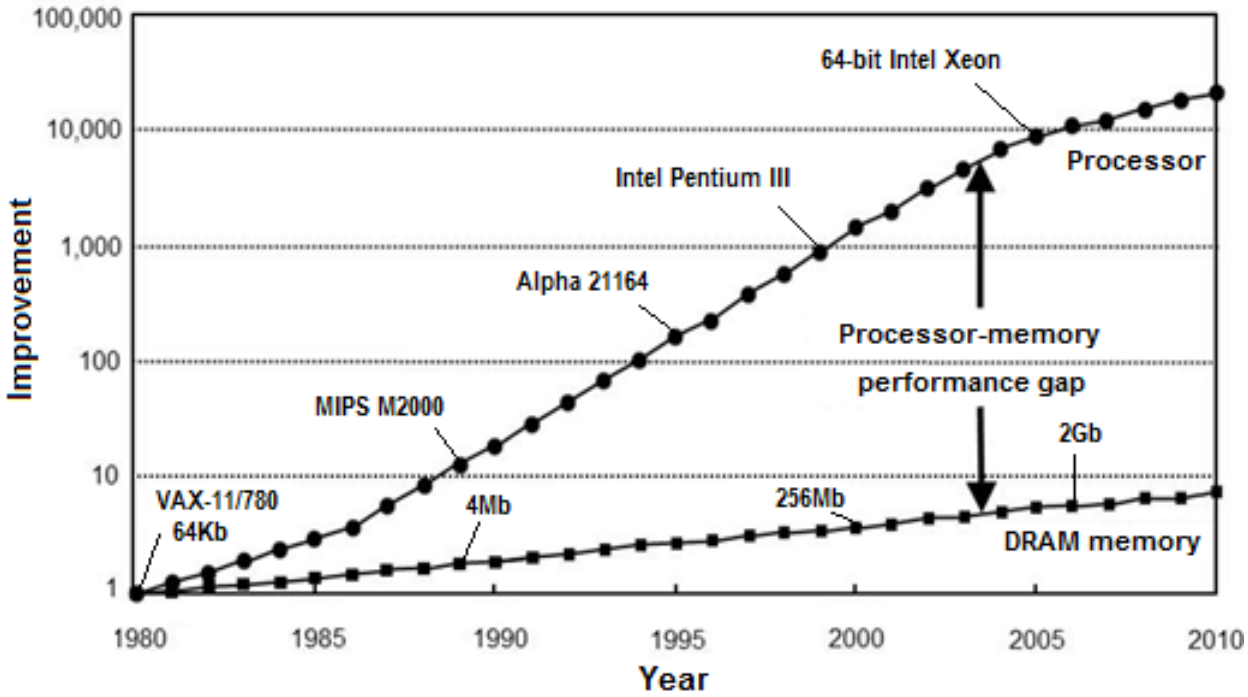
- CPUs use *registers* to store integers
  - 32–bit CPU: (most) registers are 32 bits
  - 64–bit CPU: (most) registers are 64 bits
- Address register size determines amount of addressable RAM
  - 20 bits (old 8086): 2^{20} bytes addressable = 1 Mb
  - 32 bits: 2^{32} bytes = 4 Gibi
    - 4 Gb limit on 32–bit machines
  - 48 bits: 2^{48} bytes = 256 Tebi
  - 64 bits: 2^{64} bytes = 16 Exabytes (16 billion Gb)
- Most systems have switched to 64–bit
  - Not straightforward — need 64–bit OS
  - Similar to change from 16 to 32 bits in early 1990's

Speed

- Memory speed: more important than most people realize
- CPU demands items from memory — often
  - ML instructions
  - Data
- 1Ghz CPU clock: 1 nanosecond per cycle
  - That's a fairly slow CPU
- Typical memory latency: 10 nanoseconds
  - If you have fast memory
- CPU waits tens of cycles for memory
  - Problem: your CPU slowed to 10% of its top speed

Speed Gap

Figure from "A Survey of Different Approaches for Overcoming the Processor-Memory Bottleneck", by Efnusheva et al., IJCSIT Vol 9, No 2, April 2017:



CPU speeds are improving faster than memory speeds

- CPU: around 20% improvement per year since 2004
- Memory: around 8% improvement per year

Memory hierarchy

- We can build extremely fast memory
  - . . . but the cost per byte is very high
- We can build extremely cheap memory
  - . . . but it is very slow

Memory hierarchy

- We can build extremely fast memory
  - . . . but the cost per byte is very high
- We can build extremely cheap memory
  - . . . but it is very slow
- Memory hierarchy: mix a variety of speeds and capacities
  - Faster memory serves as *cache* for slower memory

Memory hierarchy

- We can build extremely fast memory
  - . . . but the cost per byte is very high
- We can build extremely cheap memory
  - . . . but it is very slow
- Memory hierarchy: mix a variety of speeds and capacities
  - Faster memory serves as *cache* for slower memory

Modern memory hierarchy

Cost	**Type**	**Speed**
$$$$$	CPU registers	fast!!!
$$$$	Memory on CPU (L1 Cache)	fast!!
$$$	Memory on CPU (L2 Cache)	fast!
$$	L3 Cache	fast
$	RAM	slow
¢	Disk	slow!!!!!

Standardized buses

- Allow devices to be "portable" to different motherboards
- Not all devices are integrated
- Common for extras, like
  - Modems
  - Network Interfaces
  - Sound
  - Game controllers
  - Custom devices
- Also common for high–performance items
  - Graphics
  - Drives

General–purpose bus types: internal

- ISA bus (Industry Standard Architecture) 1984
  - Obsolete with Pentium
  - Included in some later motherboards
- PCI bus (Peripheral Component Interconnect) 1993
  - Introduced around Pentium
  - Better support for "plug–and–play"
- PCMCIA
  - Like PCI, but for laptops
  - Hot swappable
    - Can add devices while running
    - Can remove devices while running
- PCI-X (PCI eXtended) 1998
- PCIe (PCI express) 2004

General–purpose bus types: external

- USB 1.0 (Universal Serial Bus) 1996
  - Serial: one bit transferred at a time
  - 12 Mbit/s max transmission rate
- USB 2.0 2000
  - 480 Mbit/s max transmission rate
- USB 3.0 2008
  - 5 Gbit/s max transmission rate
  - USB 3.1, 3.2 have faster transmission rates
- USB 4 2019
  - Requires USB-C connectors
  - Carries other protocols (USB 3.2, DisplayPort, PCIe)

- FireWire 1995 – 2008
- Thunderbolt 2011

Graphics bus types

- VESA Local Bus 1992 — 1993
  - Video Electronics Standards Association
  - Closely tied to Intel 80486 CPU memory bus design
  - ISA bus was becoming bottleneck for graphics
  - Short-term solution: add an extra slot to an ISA slot
    - Could use ordinary ISA card and ignore extra
  - Obsolete with PCI
- AGP (Accelerated Graphics Port) 1997 — 2004
  - Introduced when PCI became bottleneck for graphics
  - Obsolete with PCIe

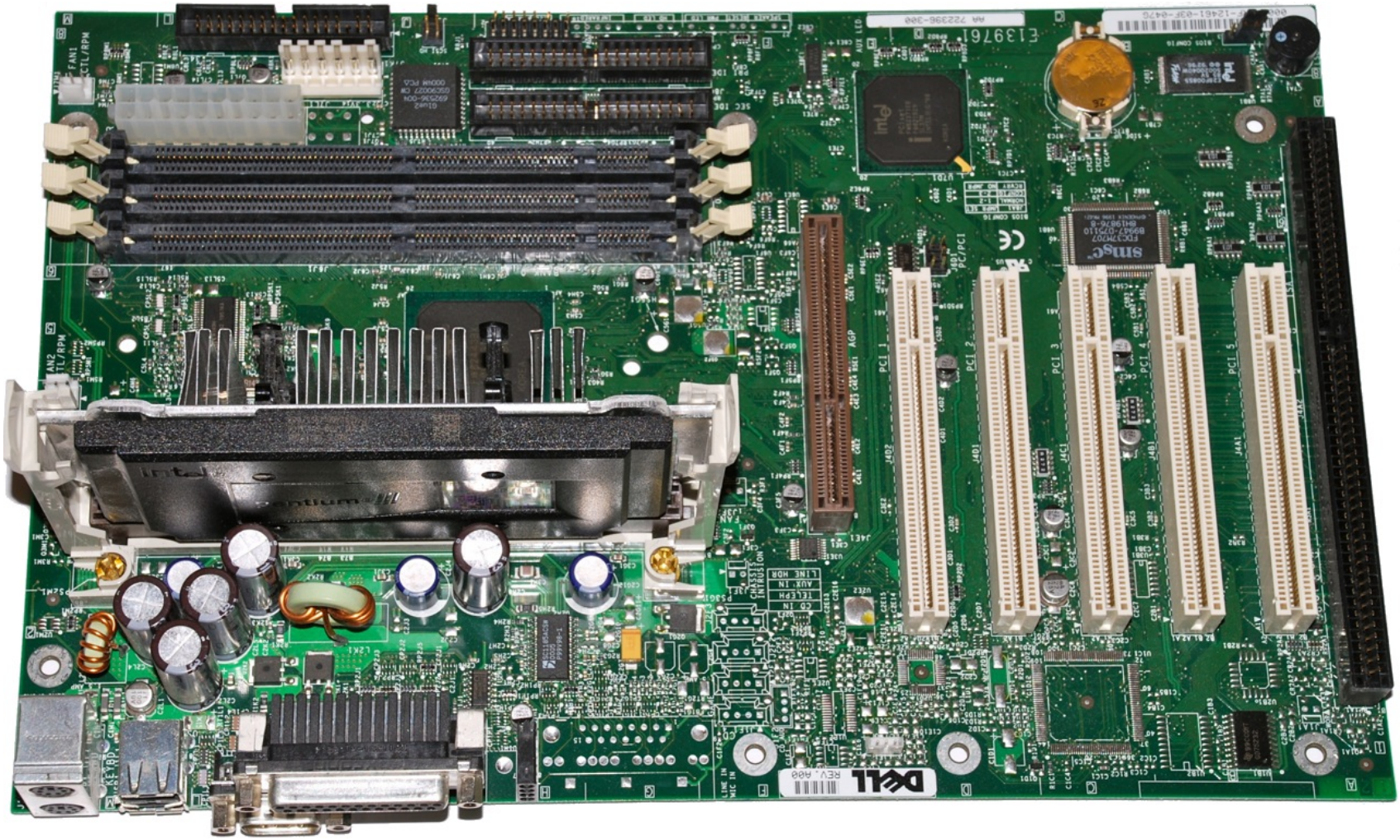
Buses for drives

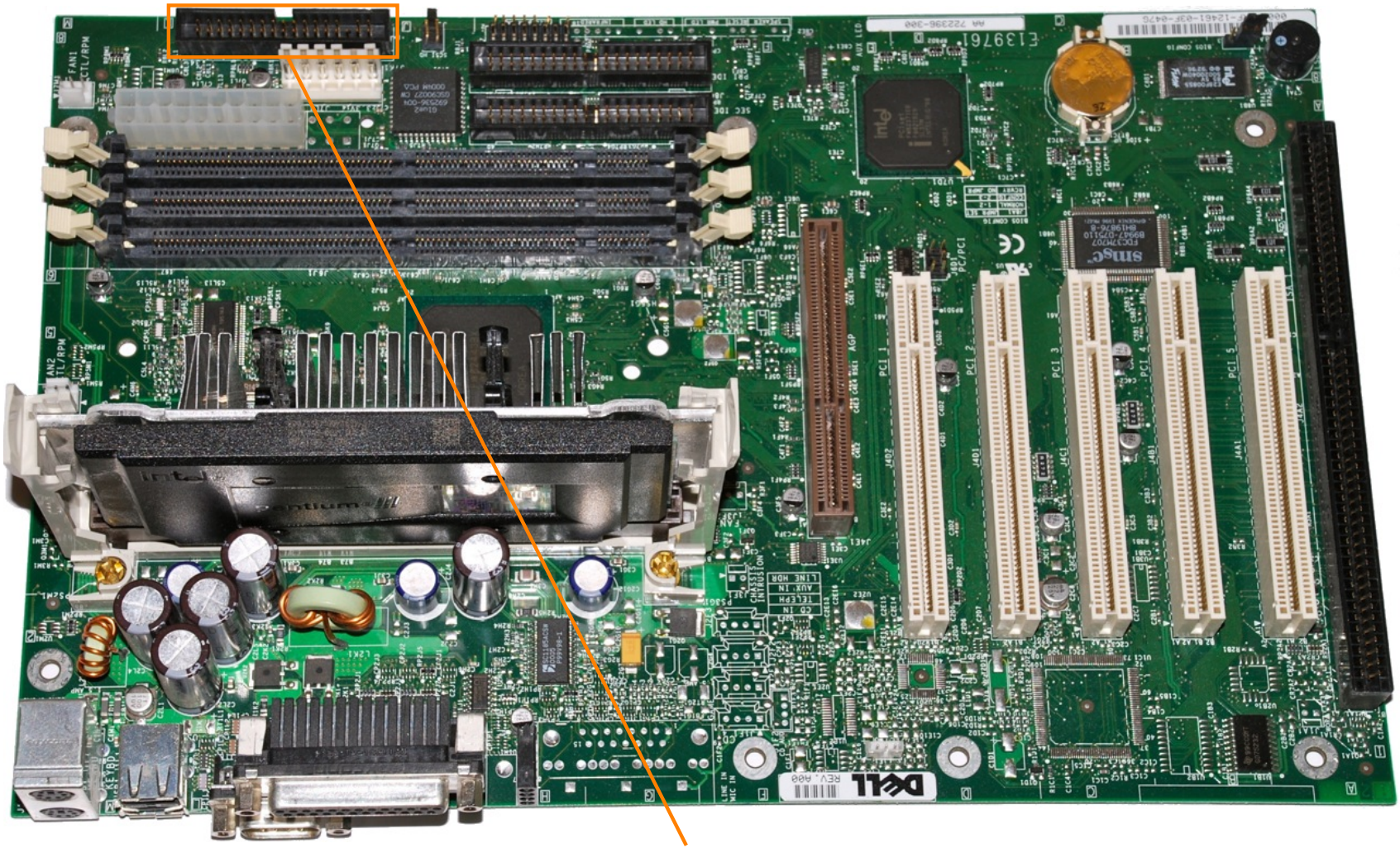
- IDE (Integrated Drive Electronics), same as (parallel) ATA: IBM AT computer Attachment 1986
  - Each IDE interface can control at most 2 drives (with 1 cable)
    - Device 0: "primary" or "master"; normally "C" drive
    - Device 1: "secondary" or "slave"; normally "D" drive
    - Historical "master and slave" terminology has seen controversy
  - Controllers typically integrated into motherboard
  - Uses "ribbon cables" (40 or 80 wires)

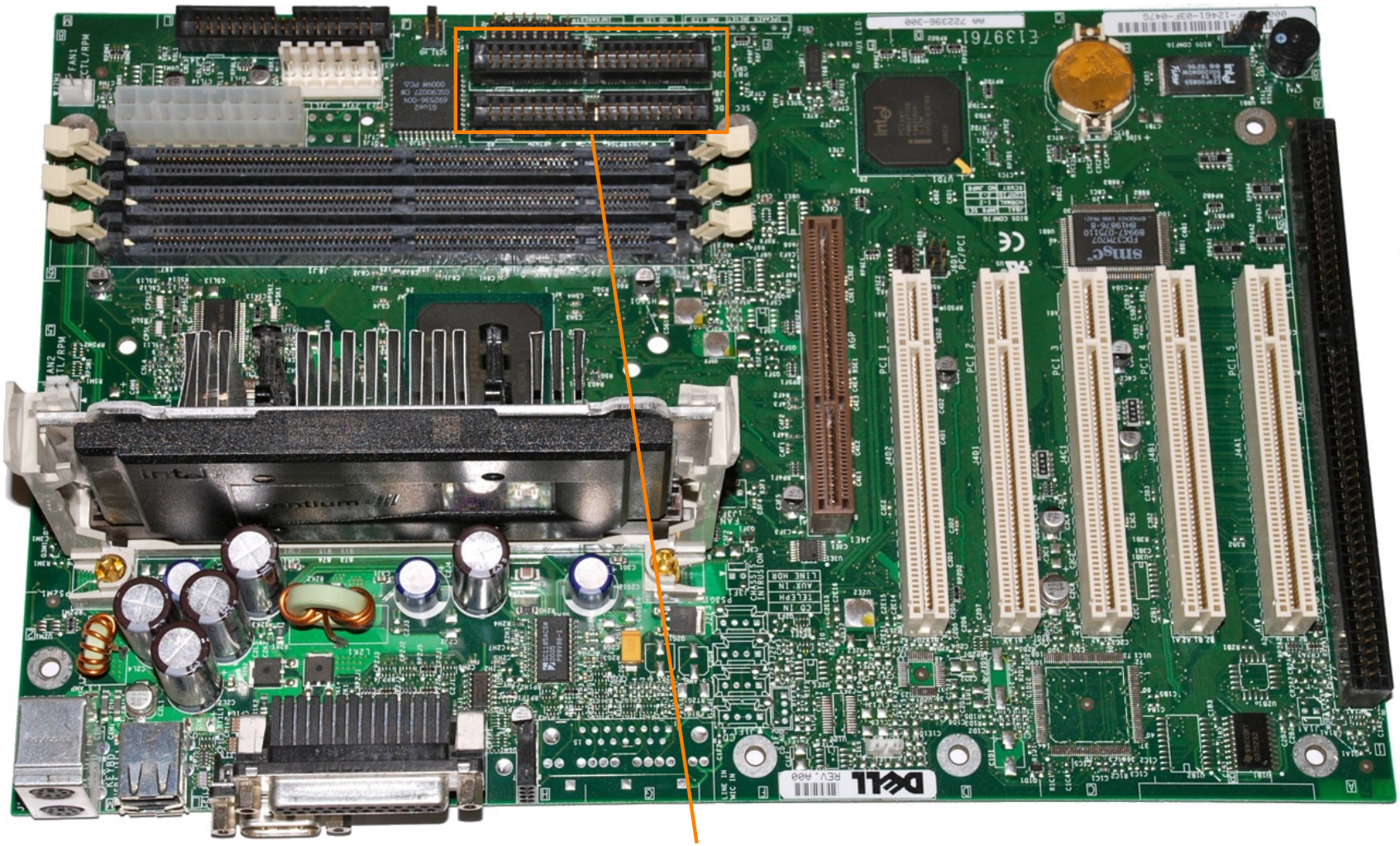
An ancient 40–wire ribbon cable, for floppy drives

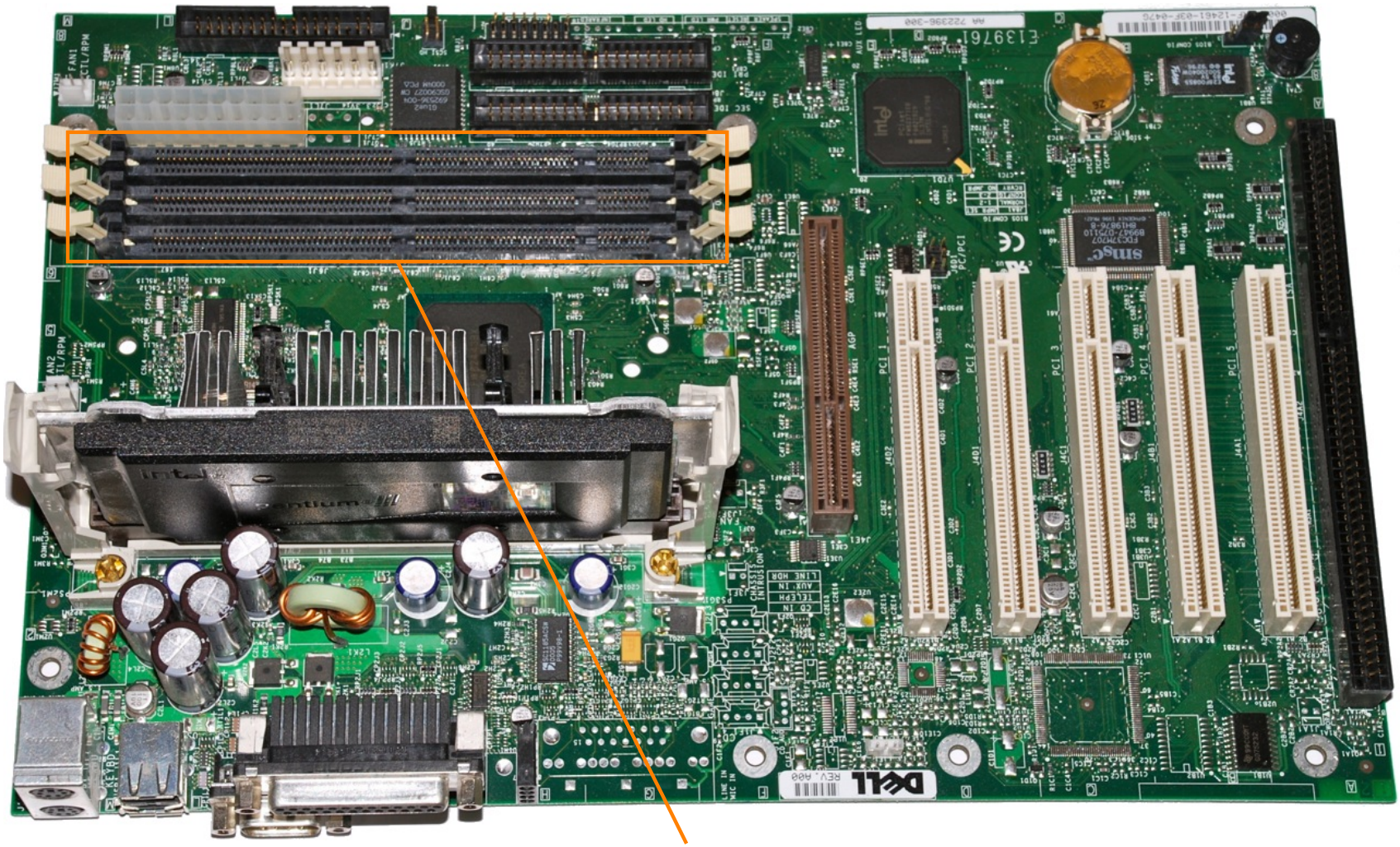
Buses for drives, part 2

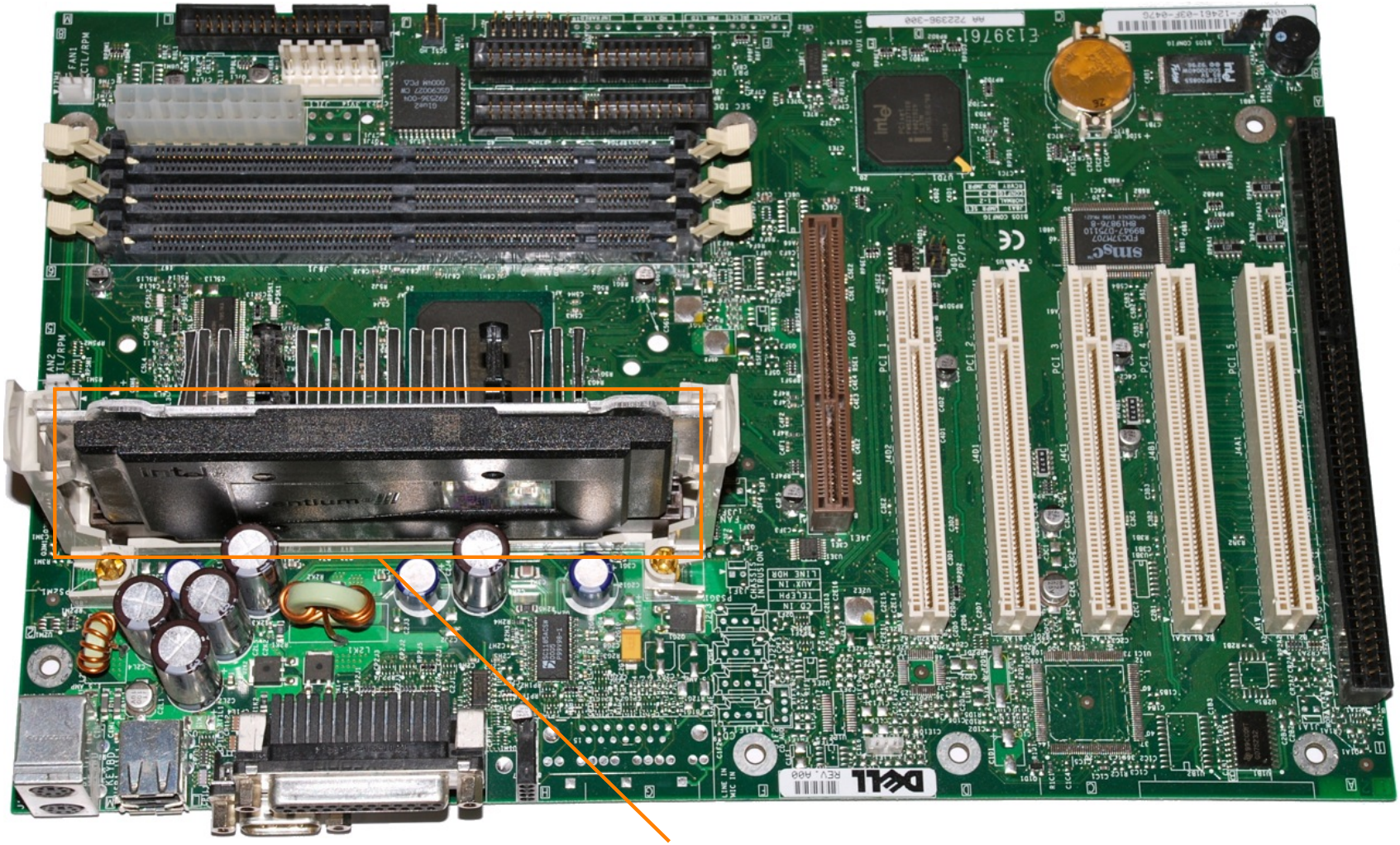
- SATA (Serial ATA) 2003
  - Gradually replaced IDE
  - Cables have 7 wires
  - Each cable connects 1 device
- SCSI (Small Computer System Interface) 1982
  - Original interface was parallel (8, then 16 bits)
- SAS (Serial Attached SCSI) 2004

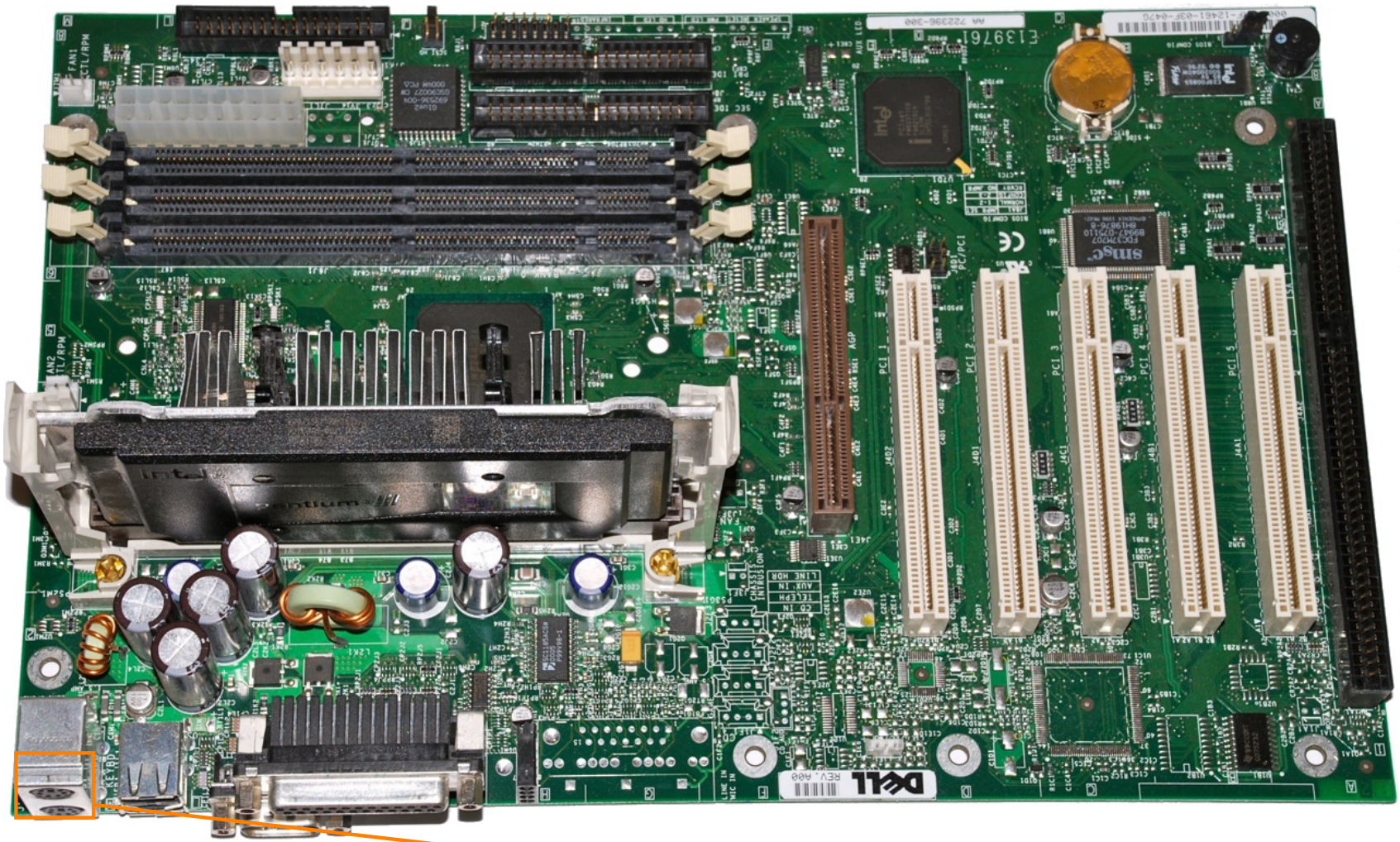

DELL
PCI 1
PCI 2
PCI 3
PCI 4
PCI 5
AGP

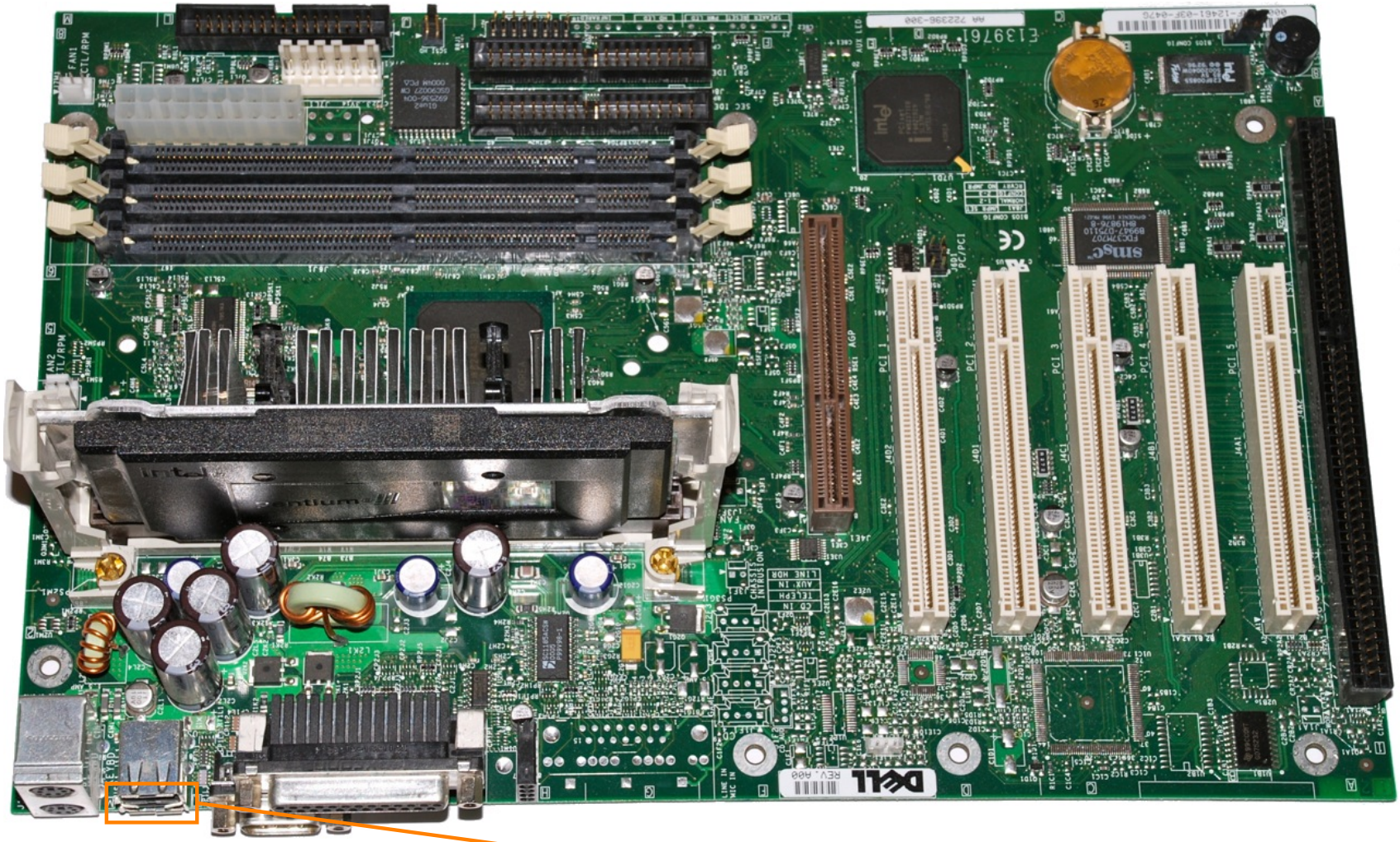

IDE 40-pin port

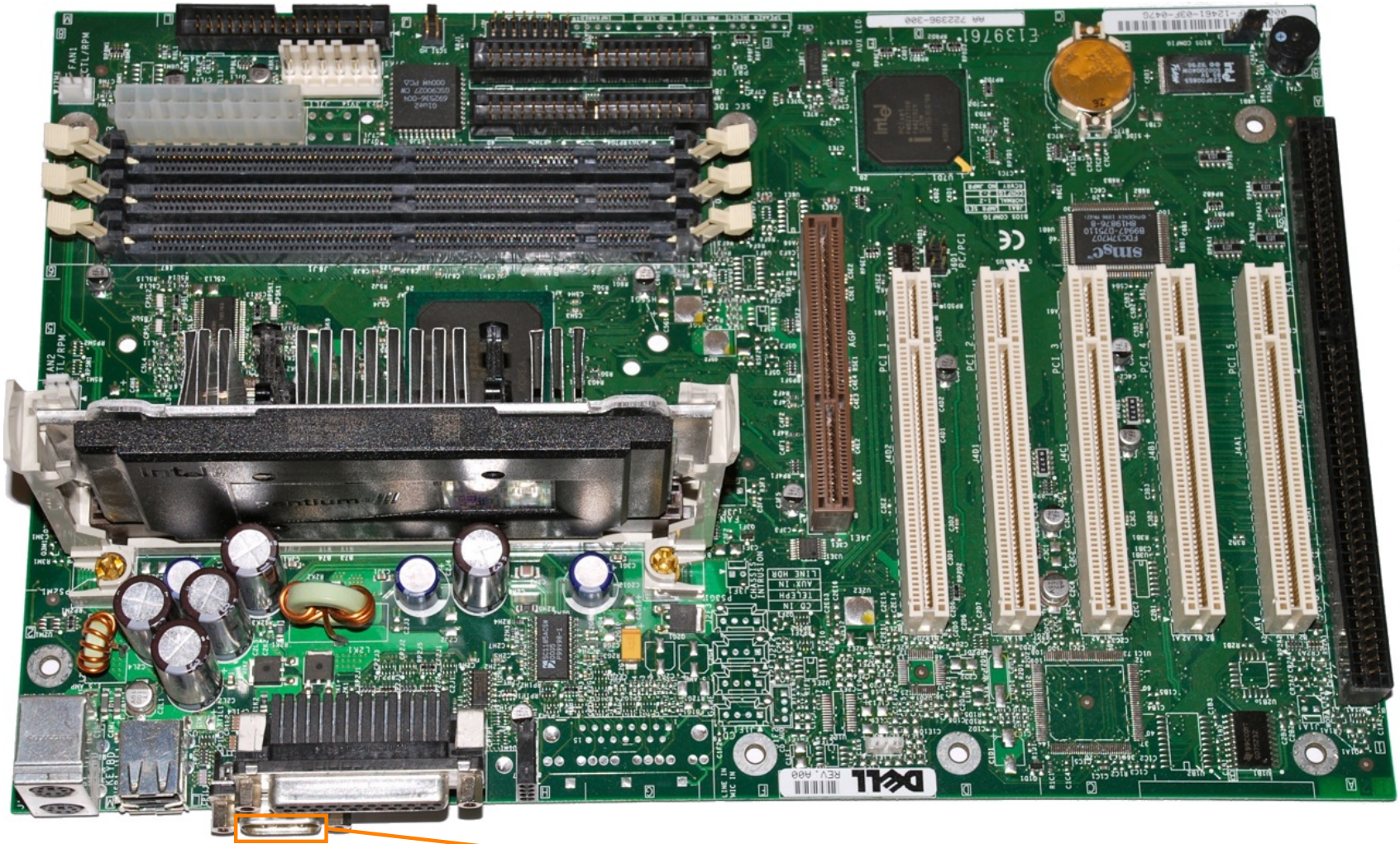

IDE 80-pin ports

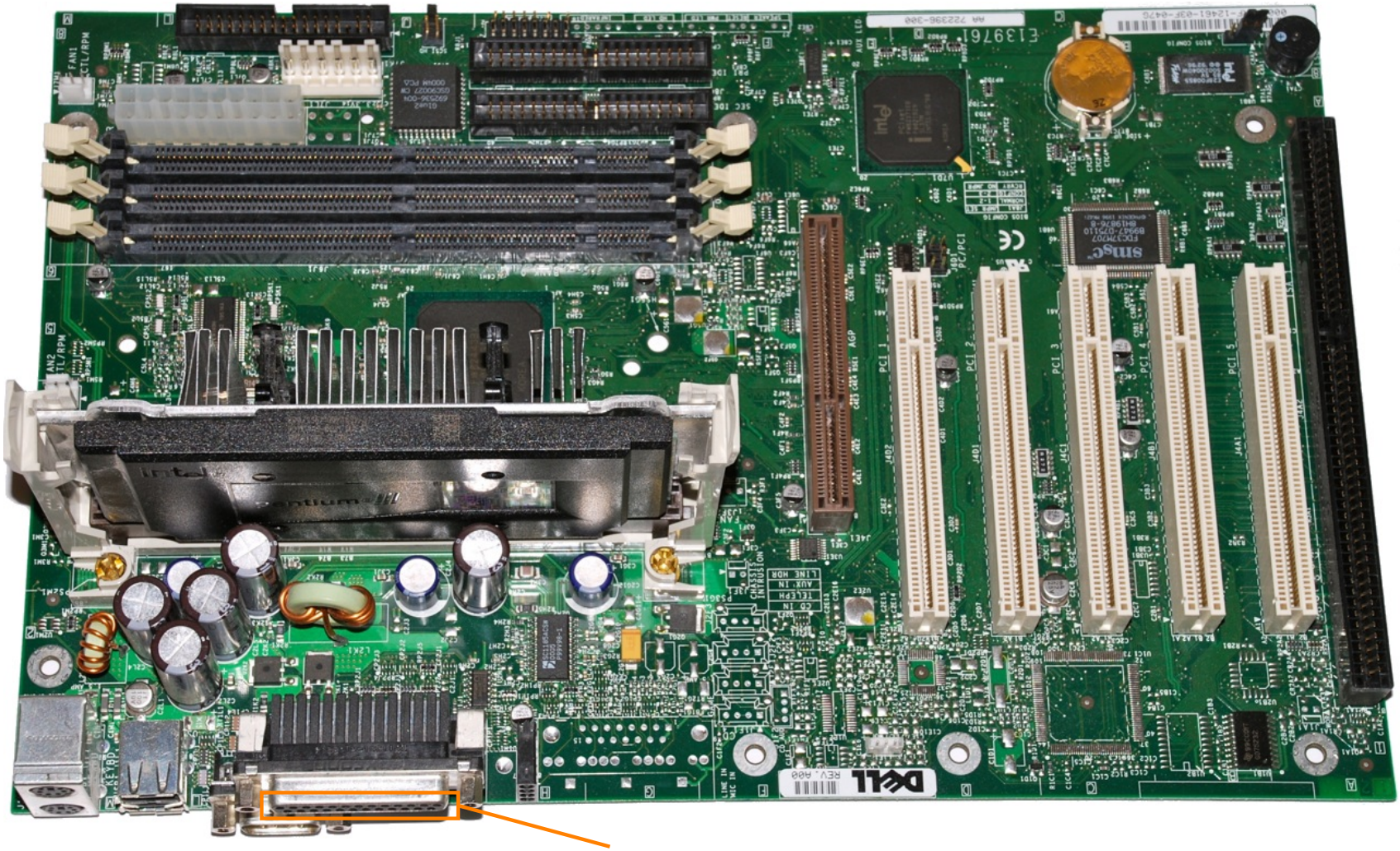

Memory slots

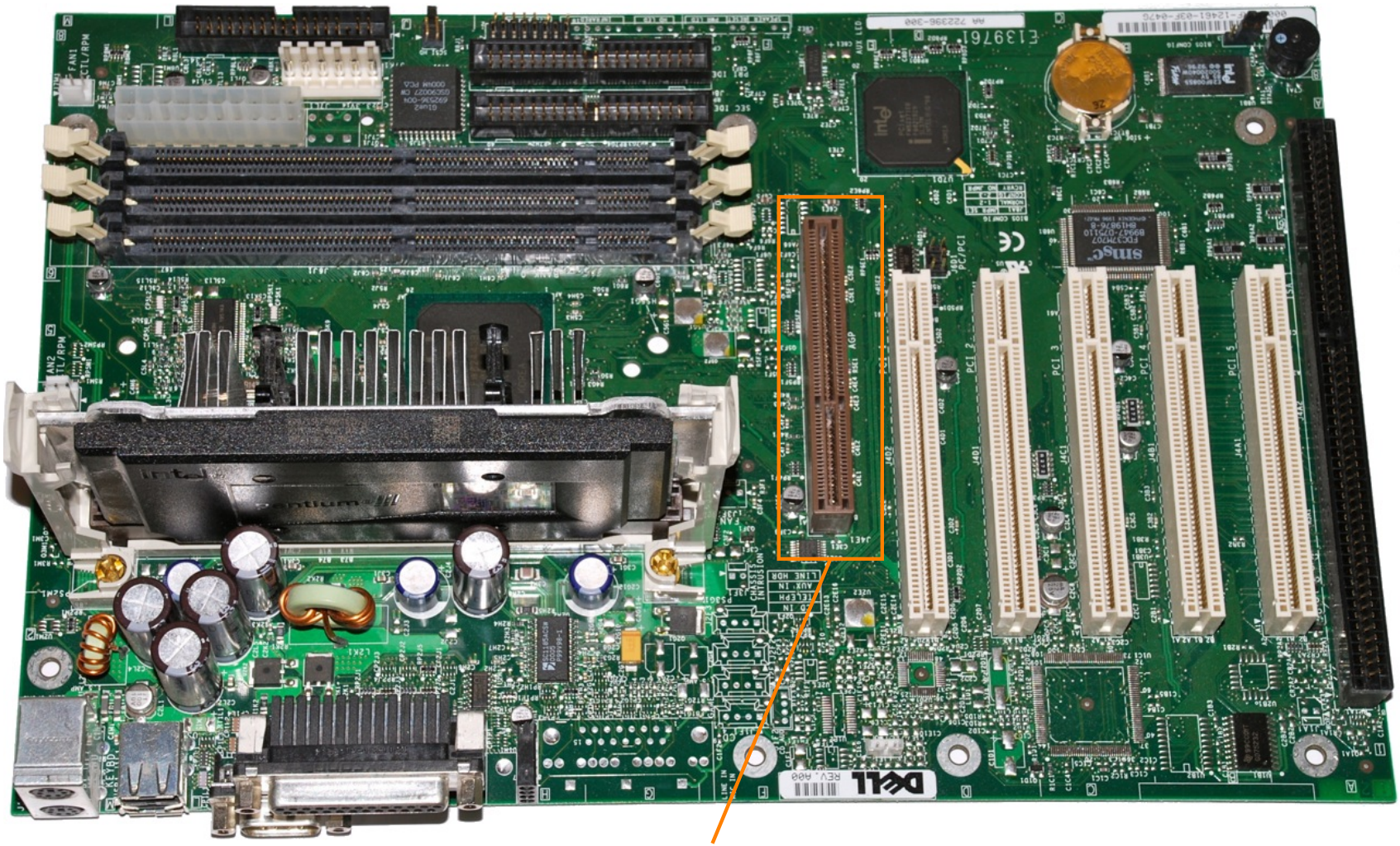

Pentium 3 CPU (vertical)

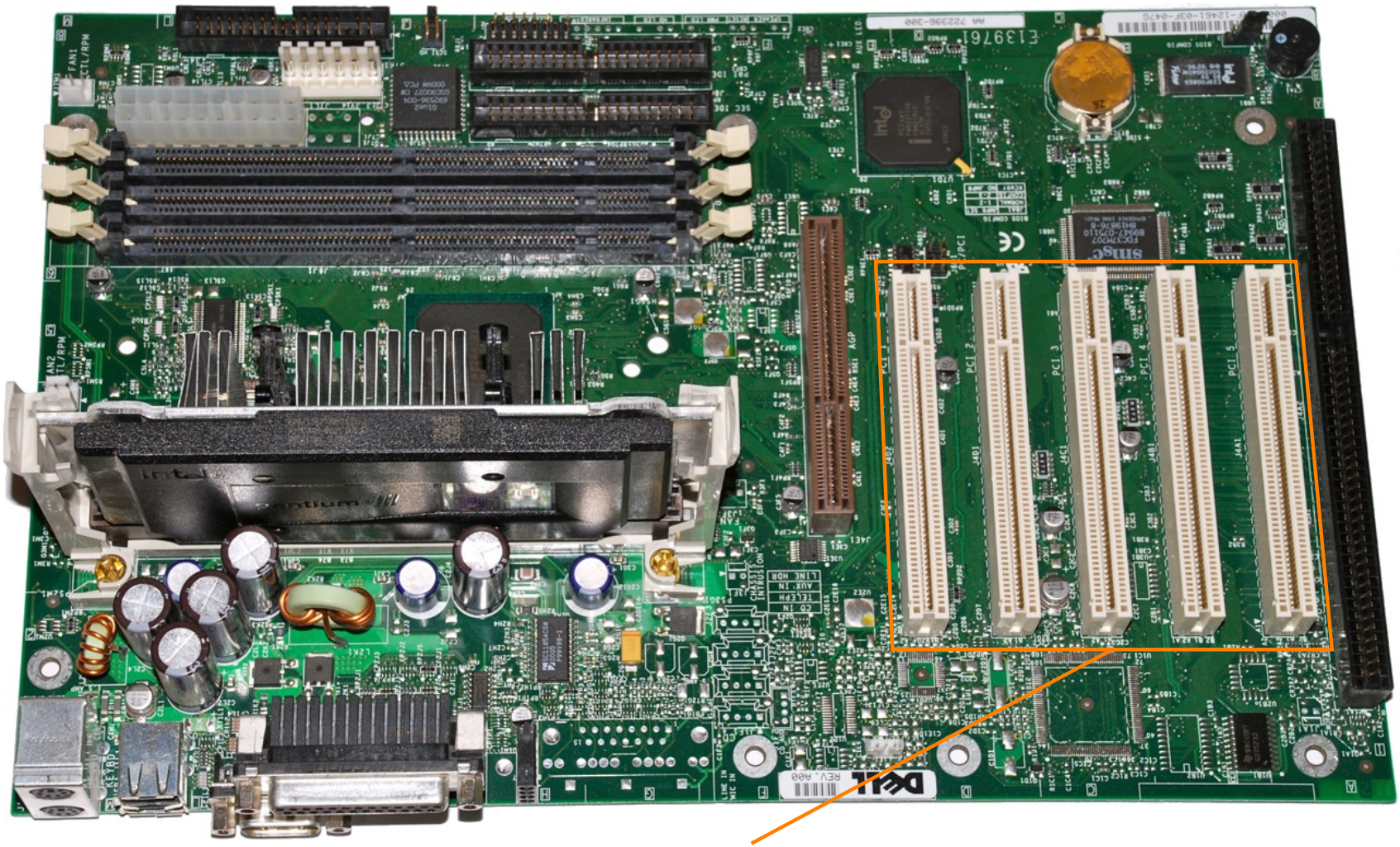

PS/2 keyboard ports

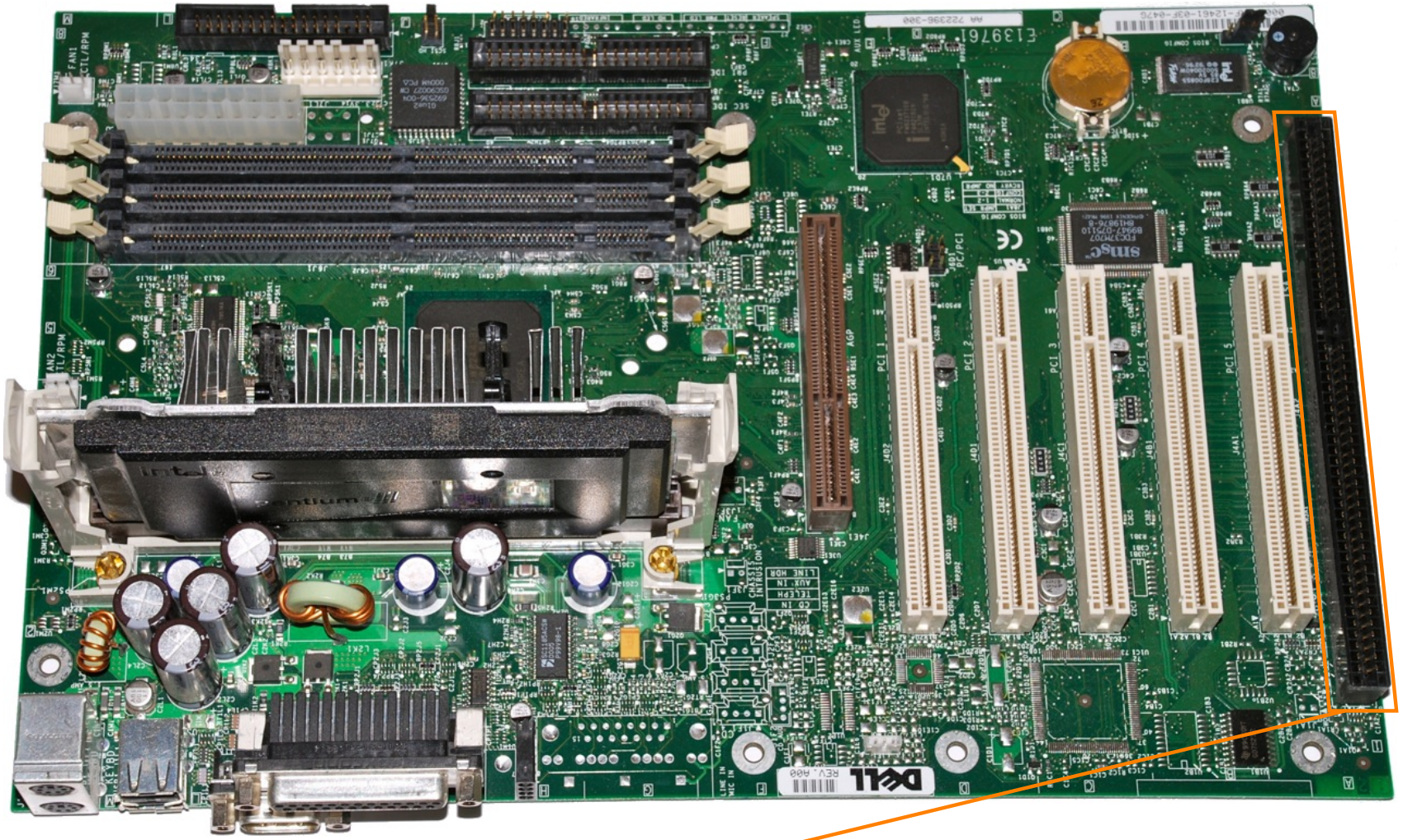

USB ports


Serial port


Parallel port

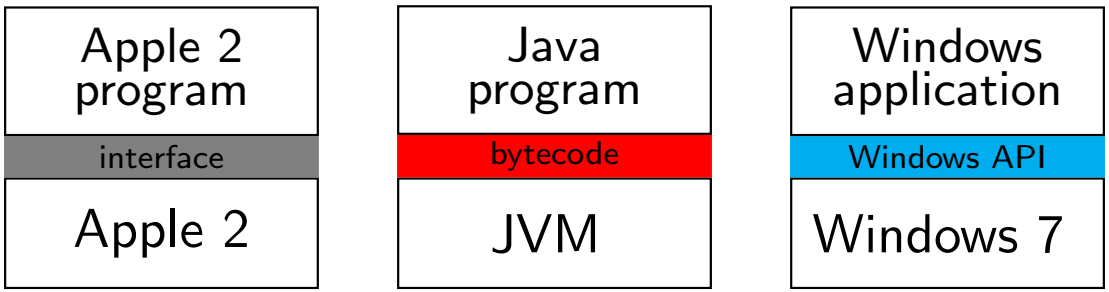

AGP slot


PCI slots


ISA slot

Virtual machines

- All software has an *interface* to an *environment*
- "Environment" can be hardware, software, or both:



- The software can run *anywhere* the interface is provided
- *Virtual machine*: software that provides a machine interface
  - Host refers to the "actual system" beneath the VM
  - Guest refers to anything running on the VM
  - More than one guest can run on a single host
  - *Virtualization*: same hardware; utilizes CPU support
  - *Emulation*: different hardware; uses software to execute

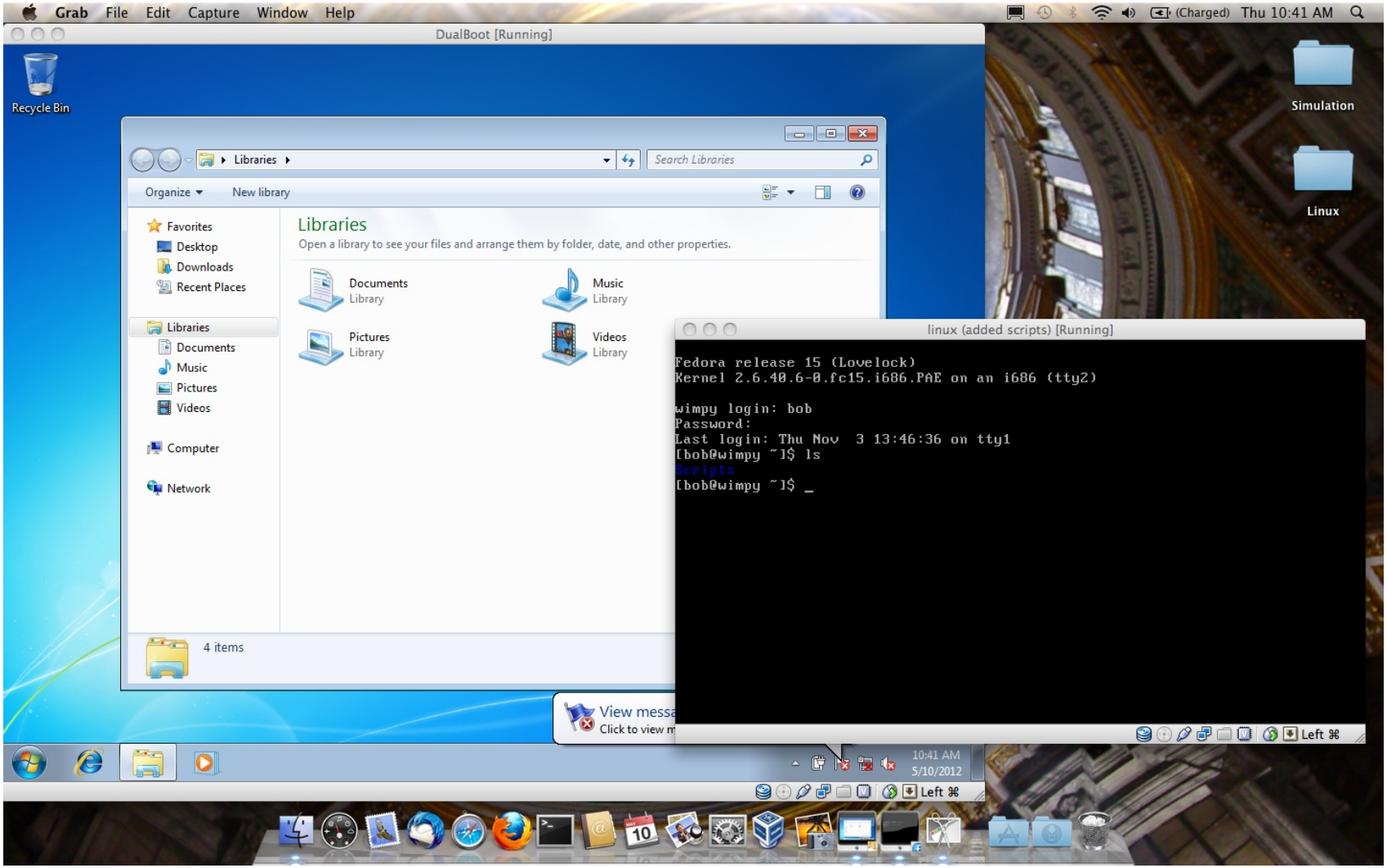
Virtual devices

Virtual machine software must provide *virtual devices*

- Virtual display
  - VM display can appear in a window
- Virtual keyboard, mouse
  - Allows keyboard, mouse to be used in VM
  - Allows keyboard, mouse to be used *outside* VM
- Virtual drives
  - Guest drives are (large) files on host system
  - CD/DVD drives can connect to disk images or actual drives
- Virtual network
  - Allows VMs to connect to host machine's network
  - Allows VMs to connect to host machine
  - Allows VMs to connect to each other
- Other devices
  - Can choose to connect, disconnect USB devices to VM

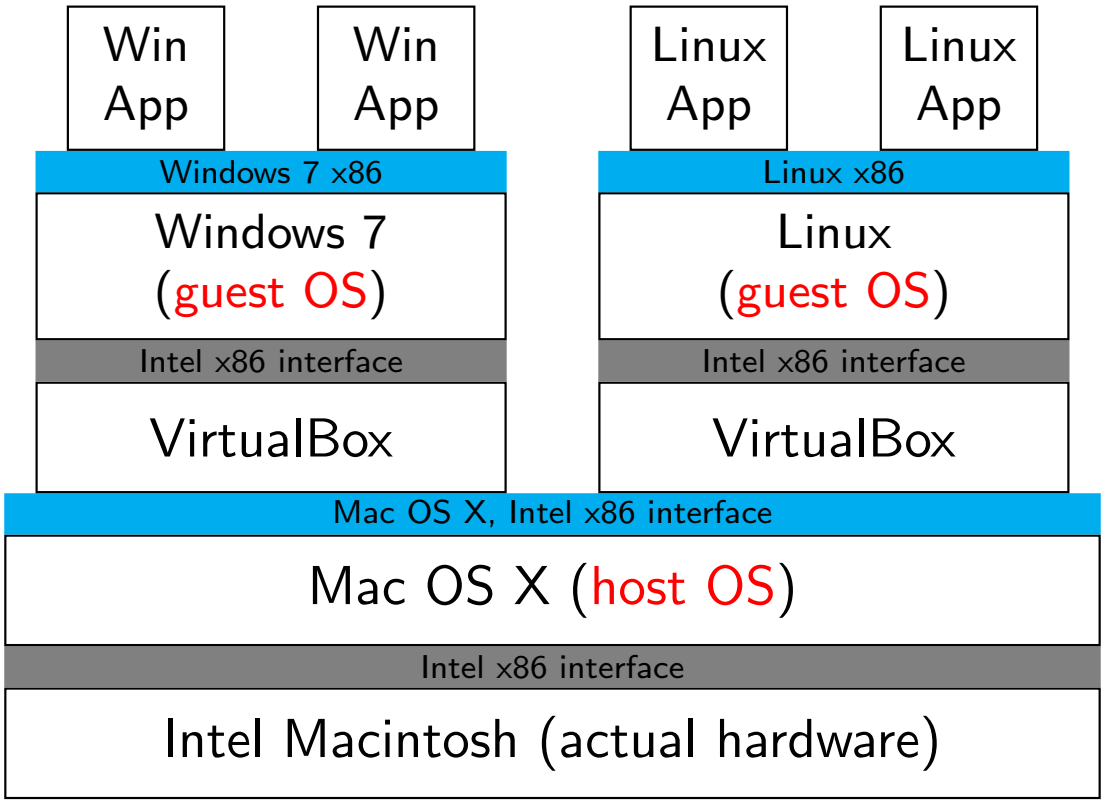
VirtualBox: x86 virtualization software by Oracle

- `http://www.virtualbox.org`
- Homeworks will be designed for this
  - May be possible to import into VMWare — have not tried this
- Open Source: Released under GPL version 2
- Runs on Windows, Linux, Macintosh, Solaris host OS's
- Has support for a variety of guest OS's

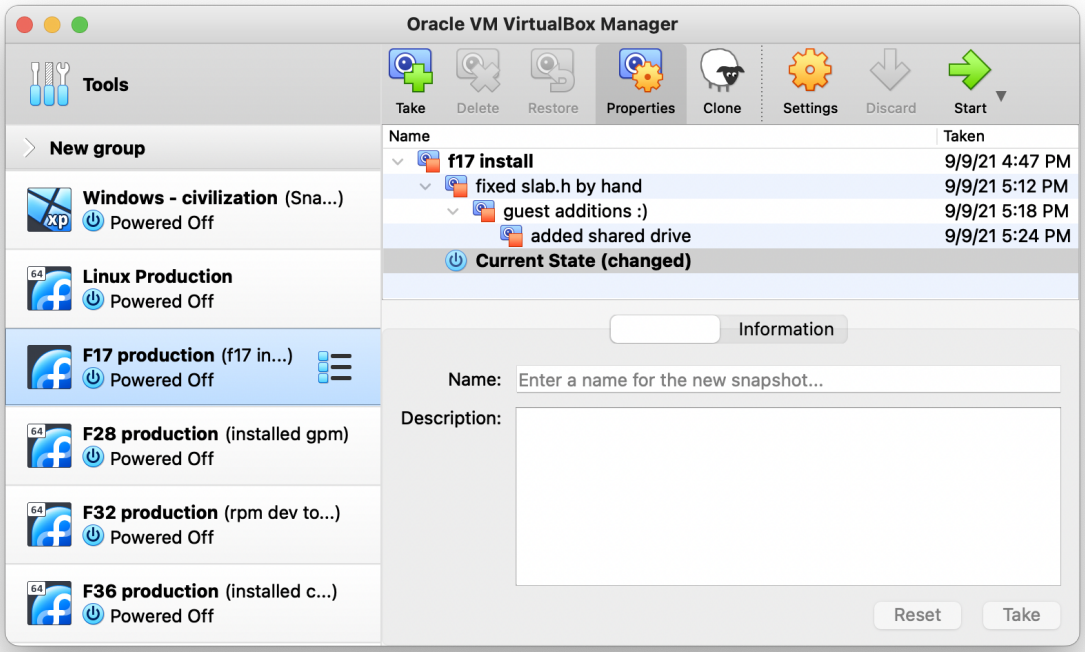


Screen capture showing Windows and Linux VMs running in VirtualBox Mac OS

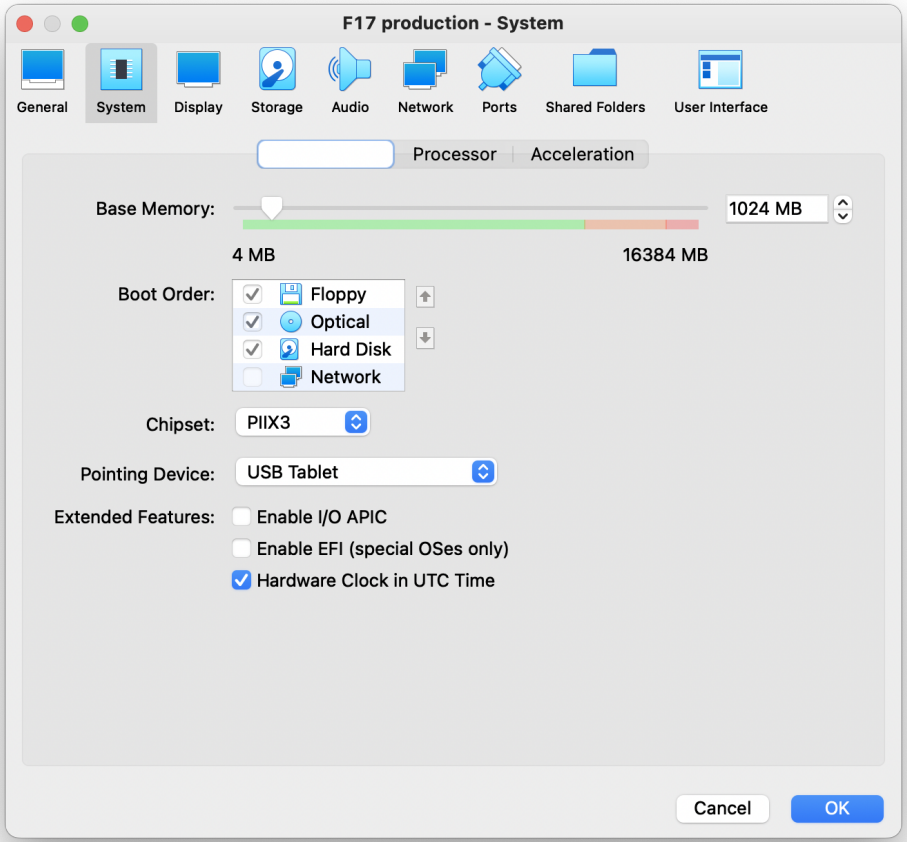
Discussion of screenshot



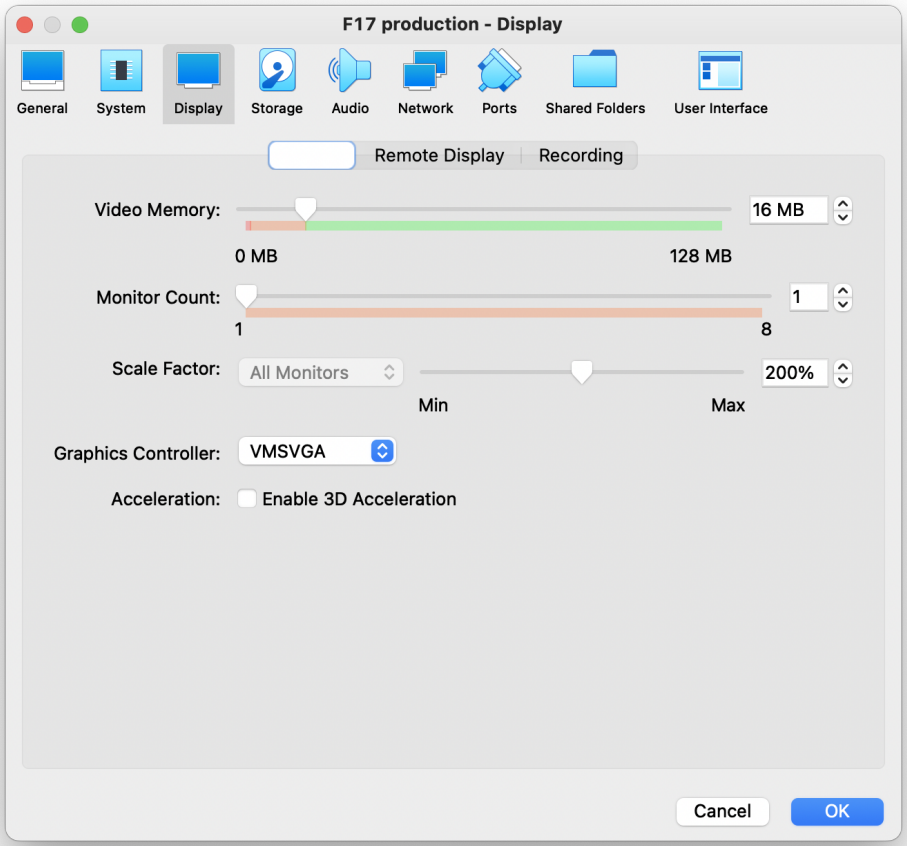
Using VirtualBox



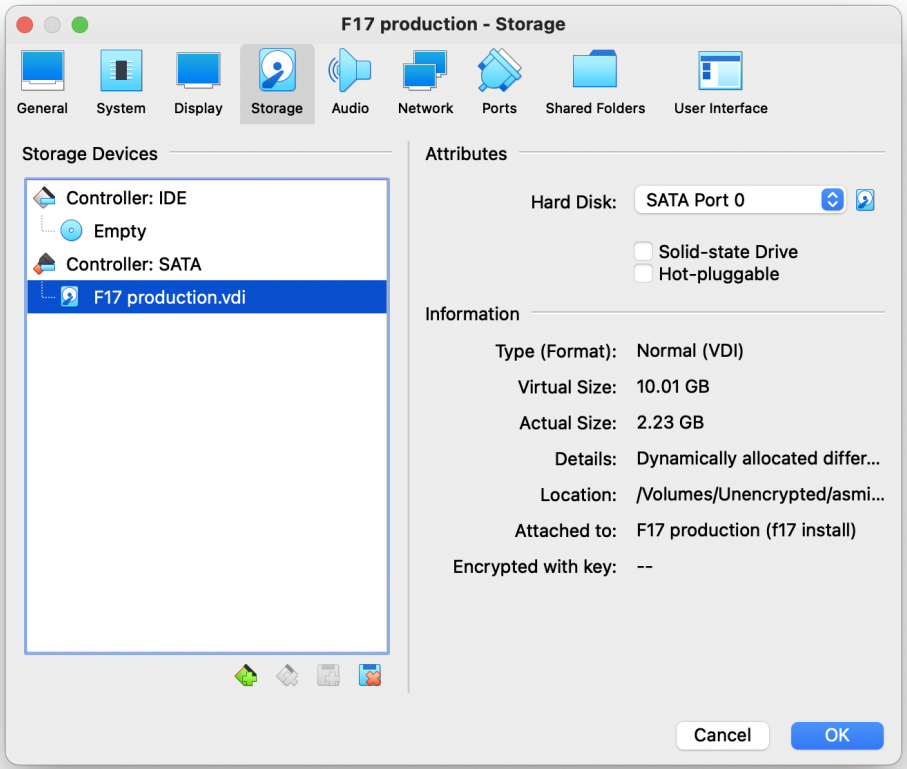
VM Settings: System



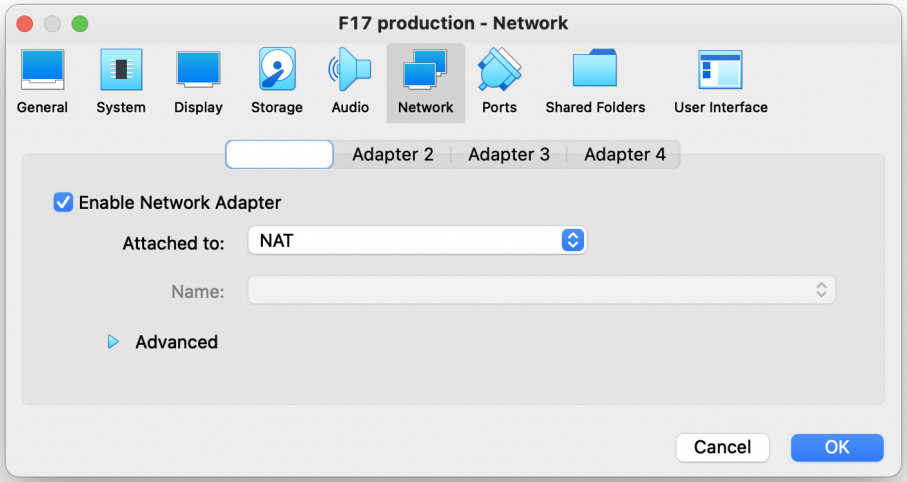
VM Settings: Display



VM Settings: Storage



VM Settings: Network



An appropriate xkcd comic: http://xkcd.com/394

THERE'S BEEN A LOT OF CONFUSION OVER 1024 vs 1000, KBYTE vs KBIT, AND THE CAPITALIZATION FOR EACH.

HERE, AT LAST, IS A SINGLE, DEFINITIVE STANDARD:

SYMBOL	NAME	SIZE	NOTES
kB	KILOBYTE	1024 BYTES OR 1000 BYTES	1000 BYTES DURING LEAP YEARS, 1024 OTHERWISE
KB	KELLY-BOOTLE STANDARD UNIT	1012 BYTES	COMPROMISE BETWEEN 1000 AND 1024 BYTES
KiB	IMAGINARY KILOBYTE	$1024\sqrt{-1}$ BYTES	USED IN QUANTUM COMPUTING
kb	INTEL KILOBYTE	1023.937528 BYTES	CALCULATED ON PENTIUM F.P.U.
Kb	DRIVEMAKER'S KILOBYTE	CURRENTLY 908 BYTES	SHRINKS BY 4 BYTES EACH YEAR FOR MARKETING REASONS
KBa	BAKER'S KILOBYTE	1152 BYTES	9 BITS TO THE BYTE SINCE YOU'RE SUCH A GOOD CUSTOMER

End of lecture