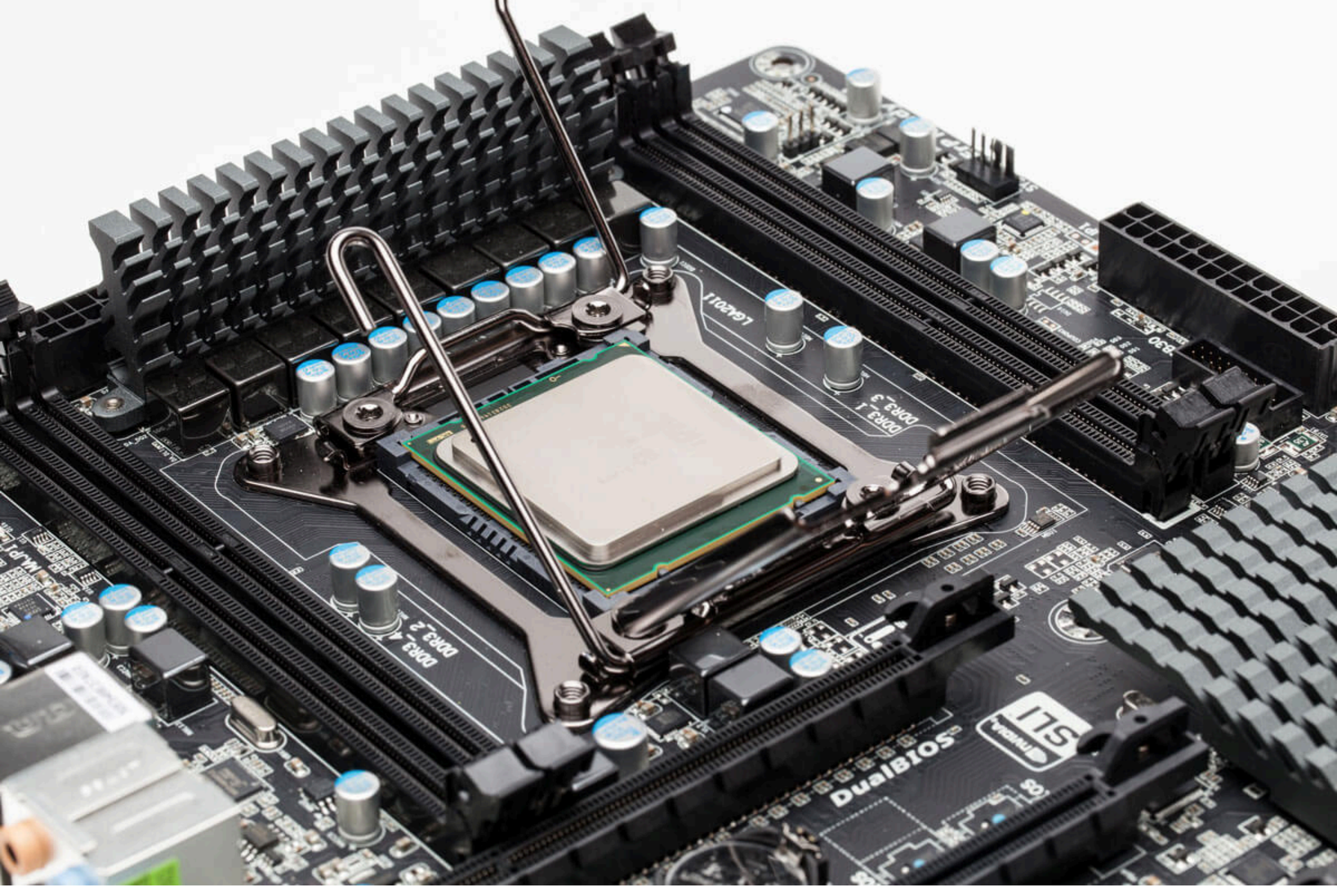


COMP10050 Software Engineering

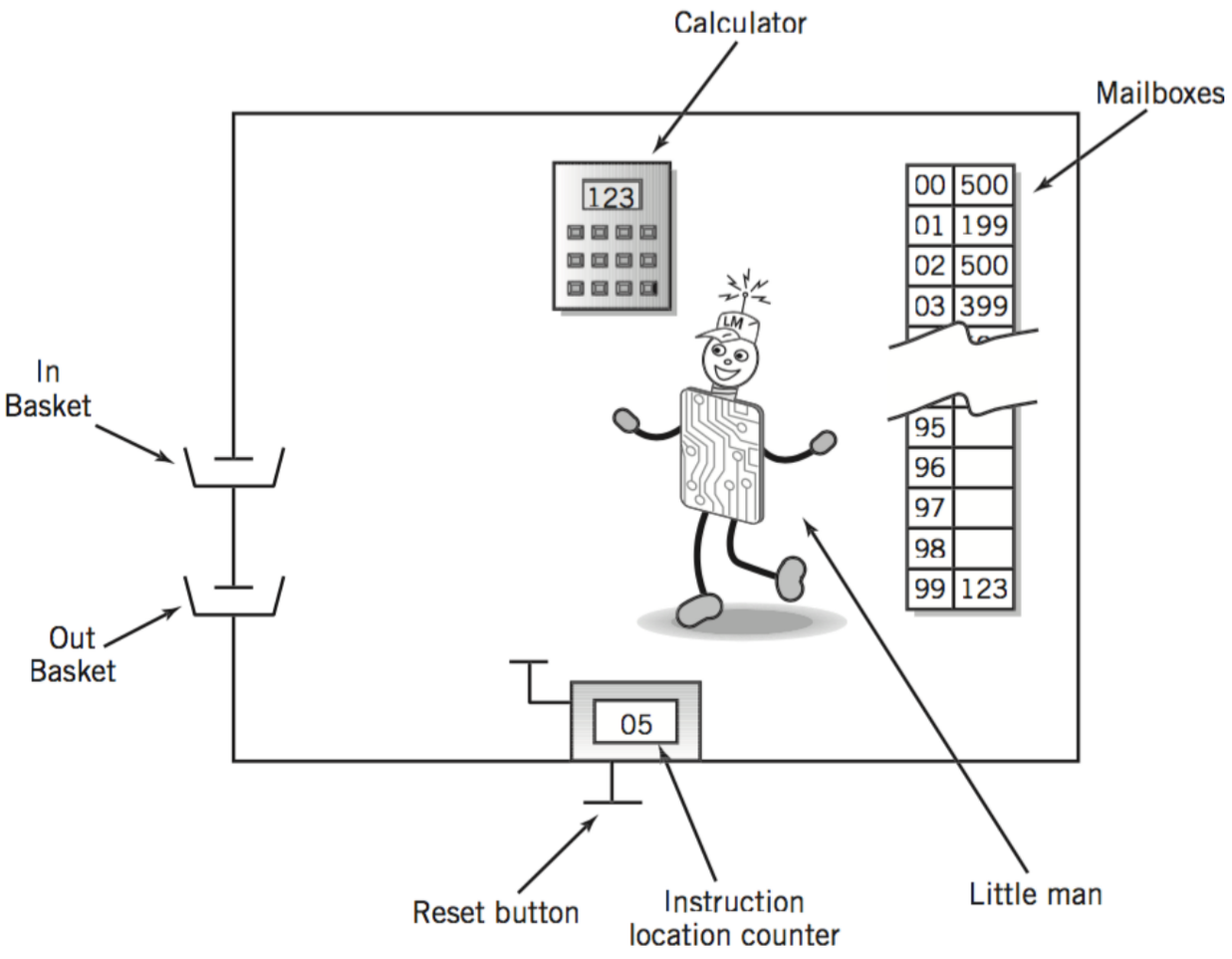
Real CPUs

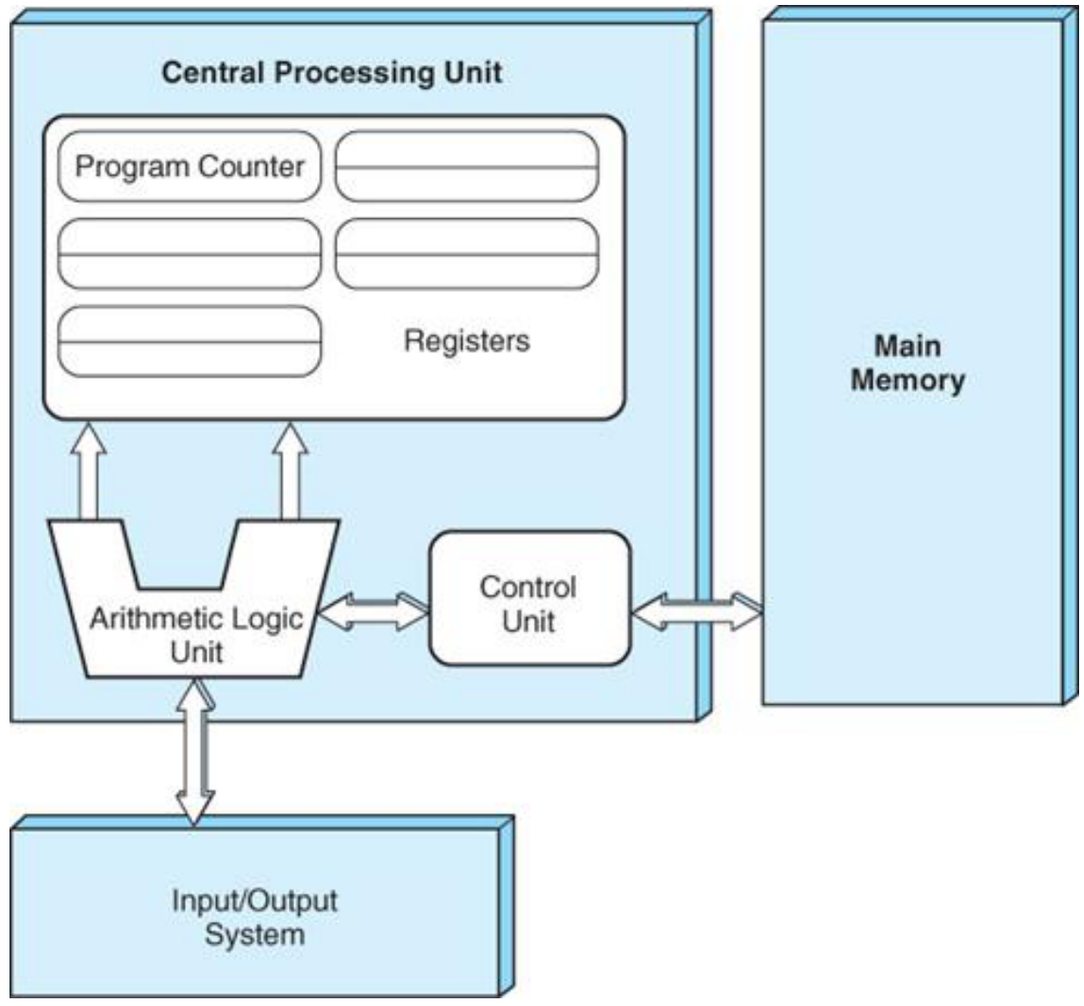
Real CPUs

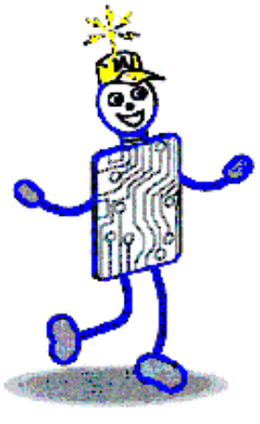
Learning Outcomes - be able to:

- understand relationship between instruction sets & assembly
- interpret simple x86 machine code instructions
- explain the factors that impact on program execution time
- explain the difference between CISC and RISC
- explain the von Neumann bottleneck

Little Man Computer Recap







Overlaps between LMC to processor

Little Man Computer	Real CPUs
The inbox is where the little man receives new paper instructions.	On a real computer, this corresponds to any input device.
Program counter - on completing an instruction, the Little Man looks at the program counter to find the next memory location with a new instruction.	For each instruction the control unit looks at the program counter to find the next instruction.
Little Man	Control unit which interprets and controls the execution of instructions.
Calculator (or accumulator) also keeps one value. This is a kind of “register”, local storage away from the mailbox memory where the little man can store a number.	CPUs have a range of different registers.
The Little Man walking back and forward to the mailboxes	This compares both location bus and data bus. The location bus tells the memory where to pull the data from or write the data to, and the data bus transports the data either way.
The outbox for LMC	On a real computer, this corresponds to any output including displaying a value on a screen, playing a sound etc.
100 mailboxes (00-99) store instructions and data for the Little Man	RAM memory directly corresponds to the mailboxes in LMC
The accumulator / calculator	ALU (arithmetic / logic unit) can perform math and logical operations & registers can be used to store results of arithmetic and logical calculations.

Word Size

Word size of the machine is how many bits the CPU can manipulate in one go.

Can consider it the basic unit of data that moves around the computer.

CPU specifications indicate if you have a 32-bit or 64 bit machine. This tells you that the CPU has a 32-bit or 64-bit word size. It can process 32-bits or 64-bits on a single machine code instruction.

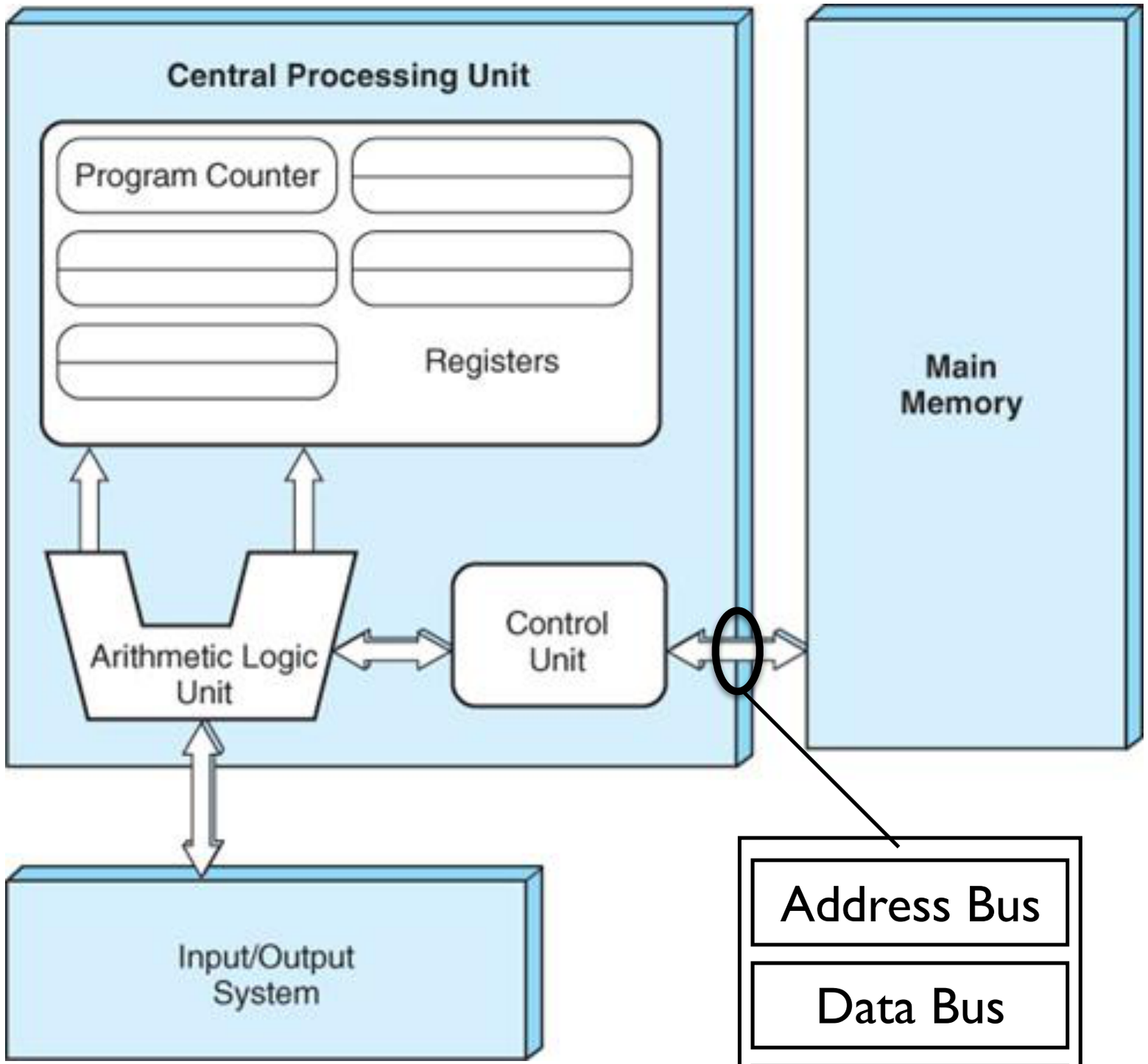
> word size = more data processed quicker

Registers

- Small, permanent storage locations within the CPU used for a particular purpose
- Manipulated directly by the Control Unit
- Wired for specific function
- Size in bits or bytes (not in MB like memory)
- Can hold data, an address, or an instruction

- How many registers does the LMC have?
- What are the registers in the LMC?
  - *Program Counter Register (PC)*
    - Also called instruction pointer (IP)
  - *Instruction Register (IR)*
    - Stores instruction fetched from memory
  - *Accumulator Register*
    - Stores results of arithmetic & logical calculations

Busses

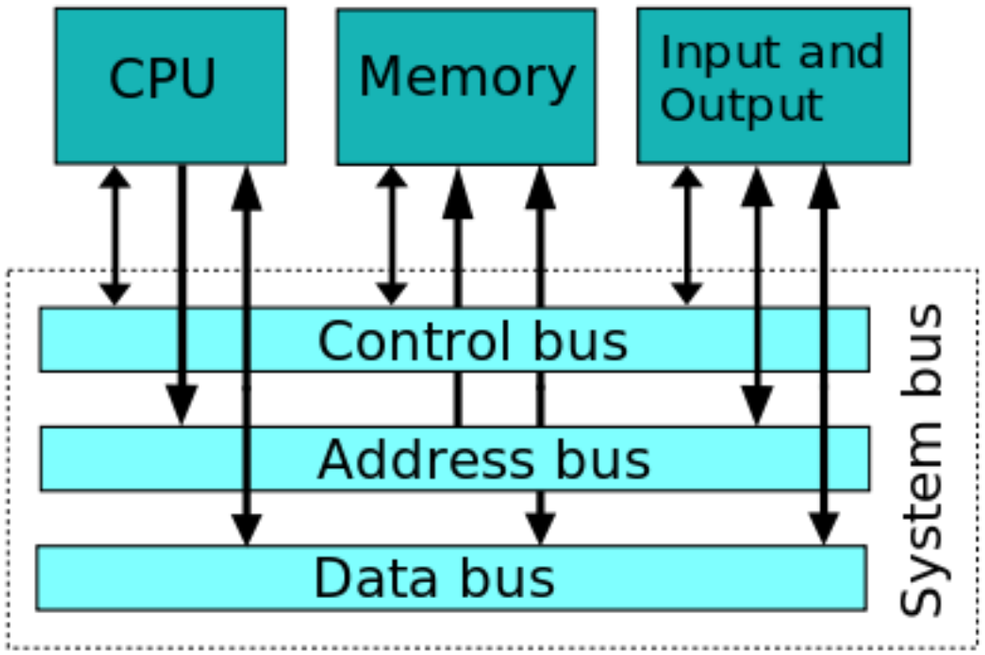


Bus width

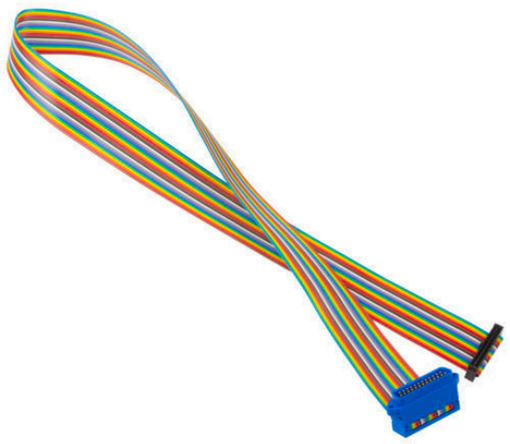
Address bus
- size of addressable memory

Data bus
- how much data can be fetched in one go

Buses



- The physical connection that makes it possible to transfer data from one location in the computer system to another
- Group of electrical or optical conductors for carrying signals from one location to another
- 4 kinds of signals
  1. Data
  2. Addressing
  3. Control signals
  4. Power (sometimes)



Data bus / Address bus

1. The width of the address bus determines the amount of memory addressable **by the CPU**
2. The width of the data bus determines the **amount of data** that can be fetched in a single cycle

LMC
What is the width of the address bus?
What is the width of the data bus?

An Intel 80386SX processor has a 16-bit data bus and a 24-bit address bus. How much memory can it address?

An Intel 80386DX processor has a 32-bit data bus and a 32-bit address bus.
How many bytes can it return in a single fetch?

Instruction Set Architectures (ISA)

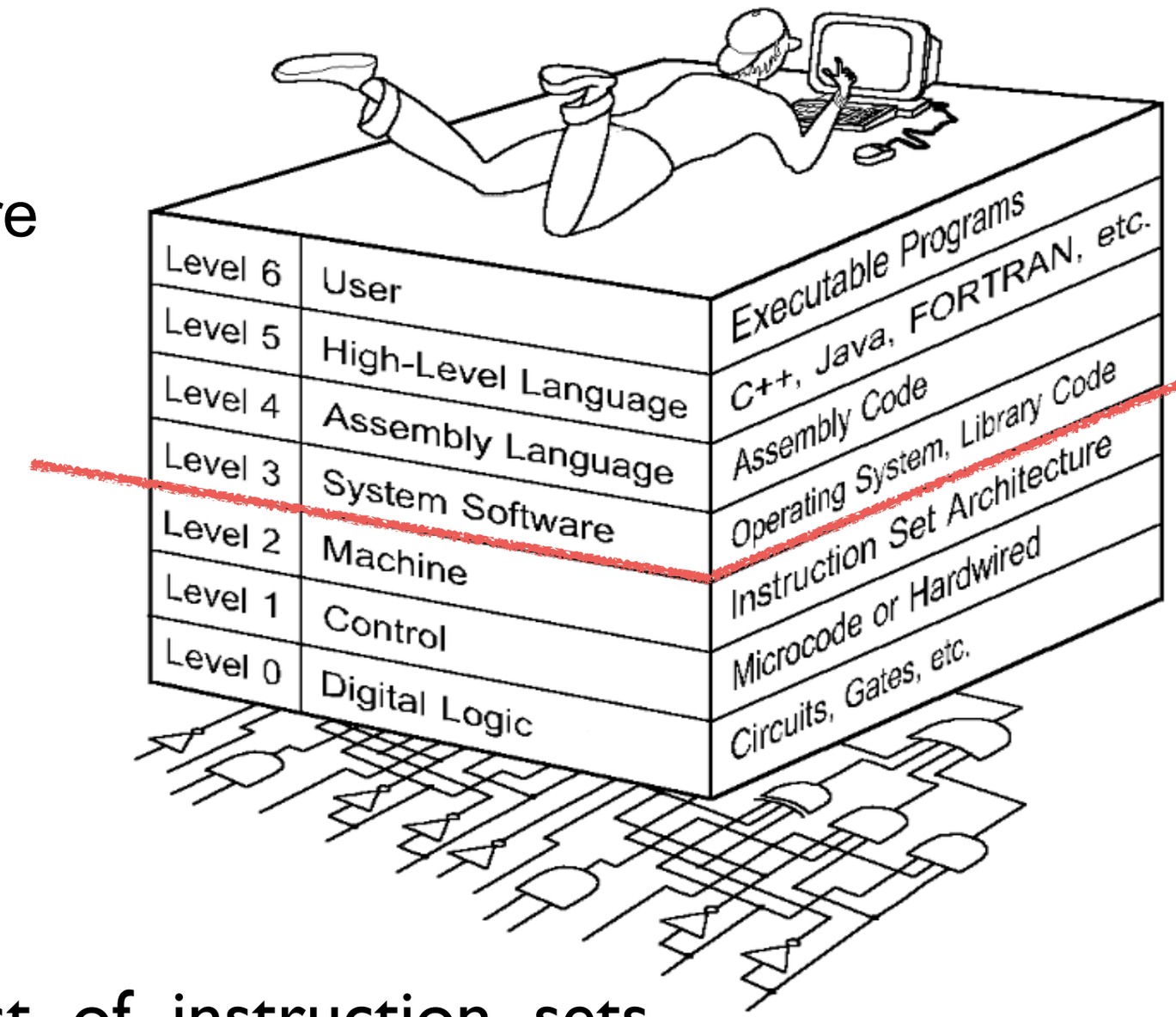
A specification of the set of op-codes and the native commands implemented by a particular processor

- Data handling and memory operations
- Arithmetic and logic operations
- Control flow operations

Allows communication hardware to software

Examples:

- x86
- Mips
- ARM
- Alpha
- AMD64 / x86-64



https://en.wikipedia.org/wiki/List_of_instruction_sets

Assembly Language

- Family of languages
- Specific to a CPU
- 1 to 1 correspondence between assembly language instruction and binary (machine) language instruction
- *Mnemonics* (short character sequence) represent instructions
- Used when programmer needs precise control over hardware, e.g., device drivers

Instruction Format

LMC

simple zero or one-operand instructions

Arithmetic	1xx	ADD
	2xx	SUB
Data Movement	3xx	STORE
	5xx	LOAD
Input/Output	901	INPUT
	902	Output
Machine Control (coffee break)	000	STOP COB

Instruction Format - e.g. x86

A machine code instruction is structured as a sequence of sub-fields. The first field is the **Op-code** (Operation-code) and subsequent fields refer to addresses or registers. The type of instructions executed by a processor is termed its **Instruction Set**.

- OPCODE: task
- Source OPERAND(s)
- Result OPERAND Location of data (register, memory)
  - Explicit: included in instruction
  - Implicit: default assumed

Machine code Instruction (Binary format)

- Not very readable – mnemonics used e.g., ADD X, Y, Sum (Assembly language)

Sample x86 machine code

Machine code bytes

```
B8 22 11 00 FF
01 CA
31 F6
53
8B 5C 24 04
8D 34 48
39 3C
72 EB
C3
```

Assembly language

```
foo:
movl $0xFF001122, %eax
addl %ecx, %edx
xorl %esi, %esi
pushl %ebx
movl 4(%esp), %ebx
leal (%eax,%ecx,2), %esi
cmpl %eax, %ebx
jnae foo
retl
```

LMC has 1 register for arithmetic - the Accumulator

x86 has 8 general purpose registers
eax, ecx, edx, ebx, esp, ebp, esi, edi

http://www.cs.virginia.edu/~evans/cs216/guides/x86.html

Some basic x86 instructions

`l` suffix means long int - 32 bit

Assembly example

```
subl %edx, %ebx          means ebx = ebx - edx
andl %esp, %eax          means eax = eax & esp
incl %ecx                means ecx = ecx + 1

movl $0xFF, %esi         means esi = 0xFF
addl $-2, %edi           means edi = edi + (-2)
shrl $3, %edx            means edx = edx >> 3
addl %ecx, %eax          means eax = eax + ecx
```

Guess what this does:

```
addl (%ecx), %eax
```

Recap: CPU Components

- Registers
  - Store data
  - Update of registers controlled by Control Unit.
  - Data can come from or go to memory, ALU or other registers.
- ALU (Arithmetic Logic Unit)
  - Performs operations, usually 2 inputs and 1 output per operation.
  - Controlled by signals from Control Unit.

Control Unit regulates and synchronises all the activity within the processor and any activity between the processor and external devices such as memory and I/O ports.

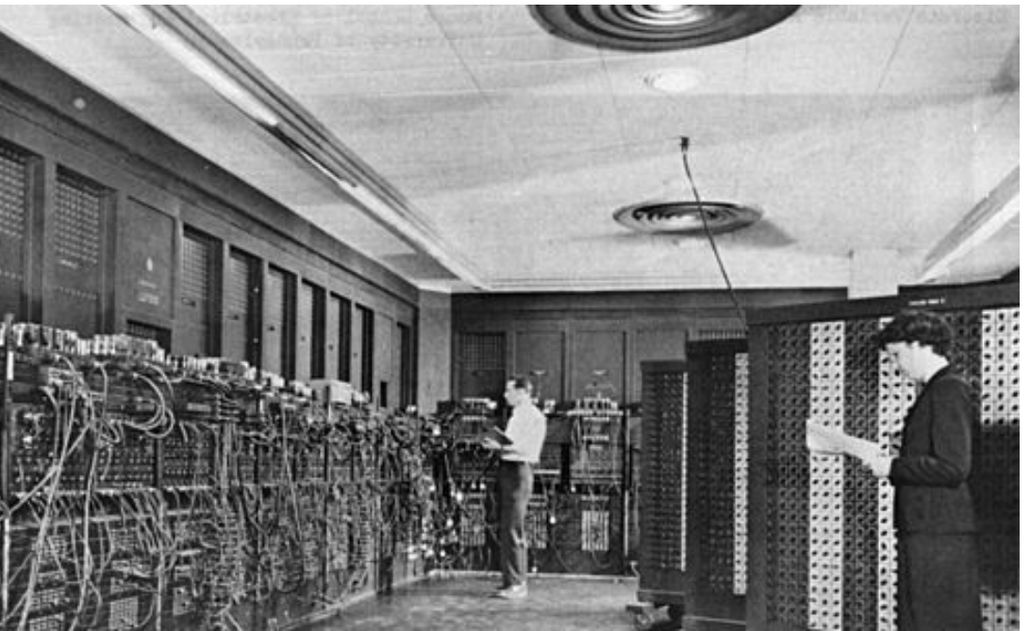
Clock Rate

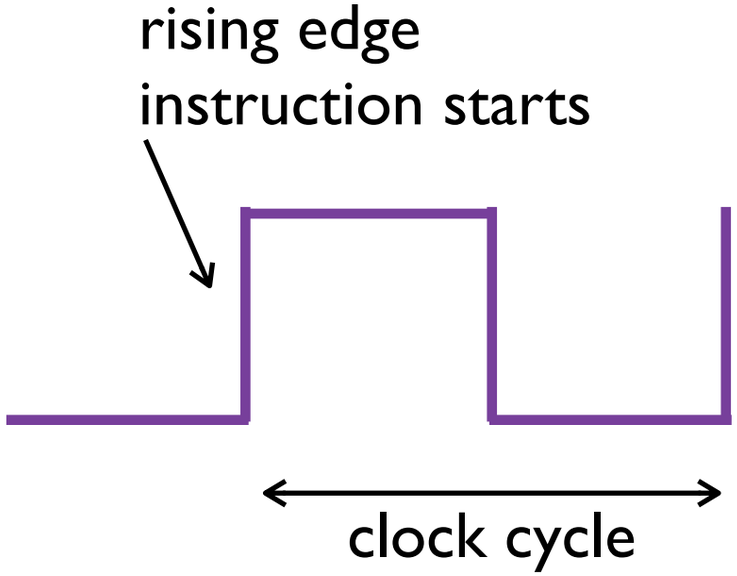
Frequency at which CPU runs

ENIAC: 100kHz, 20 cycles per instruction, 5,000 inst/sec

IBM PC: 1981 4.77MHz

2016 laptop 1GHz - 2GHz





Exercise

What are the execution times for these programs if
- each cycle takes 1 millisecond and
- each instruction takes 2 clock cycles
- except the branch and halt instructions that only take 1

Ignore time spent waiting for I/O

Add two numbers:

```
00 INP 9 01
01 STA 3 99
02 INP 9 01
03 ADD 1 99
04 OUT 9 02
05 HLT 0 00
```

Max two numbers:

```
00 INP 9 01
01 STA 3 11
02 INP 9 01
03 STA 3 12
04 SUB 2 11
05 BRP 8 08
06 LDA 5 11
07 BRA 6 09
08 LDA 5 12
09 OUT 9 02
10 HLT 0 00
11 DAT 0
12 DAT 0
```

Performance Metrics

Runtime

time it takes the program to run

$$\text{time per program} = \text{clock period} \times \text{cycles per instruction} \times \text{instructions per program}$$

900,000 instructions

2 cycles per instruction (on average)

1GHz CPU

=1 x 10^-9 x 2 x 9x10^5

= 18 x 10^-4

= 1.8 milliseconds

ISA Types

What can be done to speed up execution?

$$\text{time per program} = \text{clock period} \times \text{cycles per instruction} \times \text{instructions per program}$$

CISC (Complex Instruction Set Computer)

hardware focus - complete a task in as few lines of assembly as possible.
complex instructions built directly into the hardware
high-level code relatively simple/short

RISC (Reduced Instruction Set Computer)

Software focus - simple instructions that can be executed within one clock cycle

Performance Metrics

$$\text{time per program} = \text{clock period} \times \text{cycles per instruction} \times \text{instructions per program}$$

Reducing any of the terms will reduce runtime

clock period

- 1GHz CPU, 1 billion cycles per second, 1 nano second per cycle

cycles per instruction

- RISC: reduce this, simpler instructions

instructions per program

- CISC: reduce this, complex instructions that do more

ISA Trade-offs

- Trade-offs
  - A simpler instruction set may offer the potential for higher speeds, reduced processor size, and reduced power consumption.
  - More complex set may optimize common operations, improve memory/cache efficiency, or simplify programming.

The CISC approach attempts to **minimize the number of instructions per program**, sacrificing the number of cycles per instruction.

RISC does the opposite, **reducing the cycles per instruction** at the cost of the number of instructions per program.

CISC vs. RISC

MULT 2:3, 5:2

LOAD A, 2:3
LOAD B, 5:2
PROD A, B
STORE 2:3, A

CISC	RISC
Emphasis on hardware	Emphasis on software / compiler takes most of the burden
Instructions can take several cycles	Instructions complete in one cycle
Efficient use of RAM	Heavy use of RAM
Small code sizes	Large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

More here: https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/

ISA Characteristics

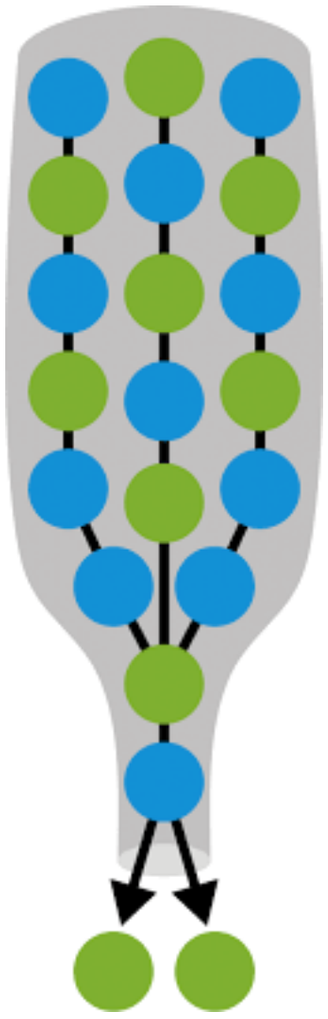
- Bits /register width
  - N-bit architecture: 8, 16, 32, 64-bit
- Operands
  - Max number of operands in a single instruction
  - 3 operands (A: = B+C), 2 operands (A: = B; A:= A+C)
- Endianness
  - Big endian, little endian, or configurable

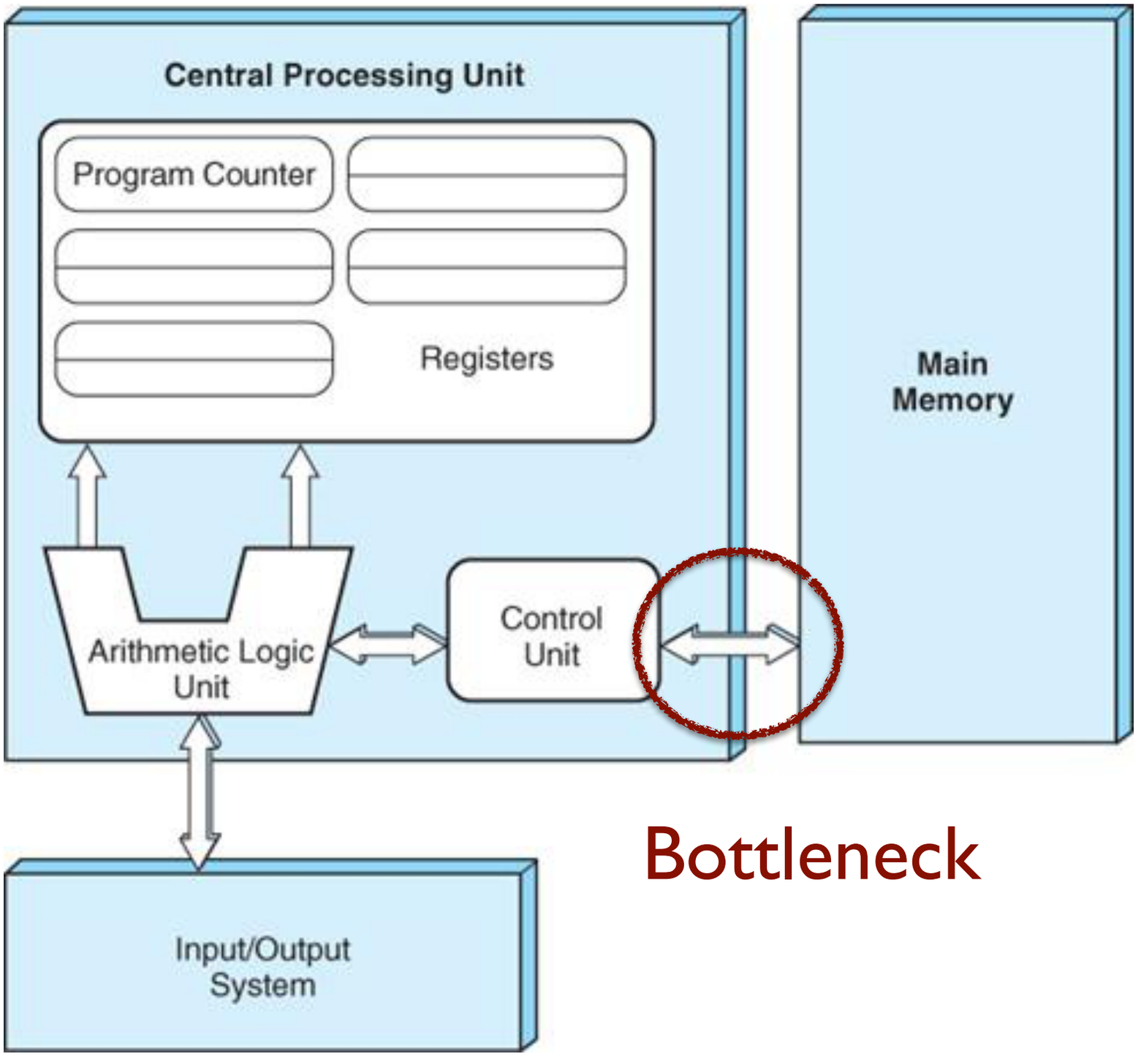
ISA Examples

- RISC
  - MIPS – embedded systems, video game consoles (PS, PS2, PSP, Nintendo64)
  - ARM – small portable devices, Raspberry Pi, smart phones and tablets (64-bit ARM in iPhone, Snapdraggon in Nexus)

- X86, X86-64 – AMD (PS4, Xbox One) and Intel (core i3, i5, i7)
  - Based on CISC - Internal RISC core under the CISC instructions.
  - https://en.wikipedia.org/wiki/X86_instruction_listings

Von Neumann Architecture





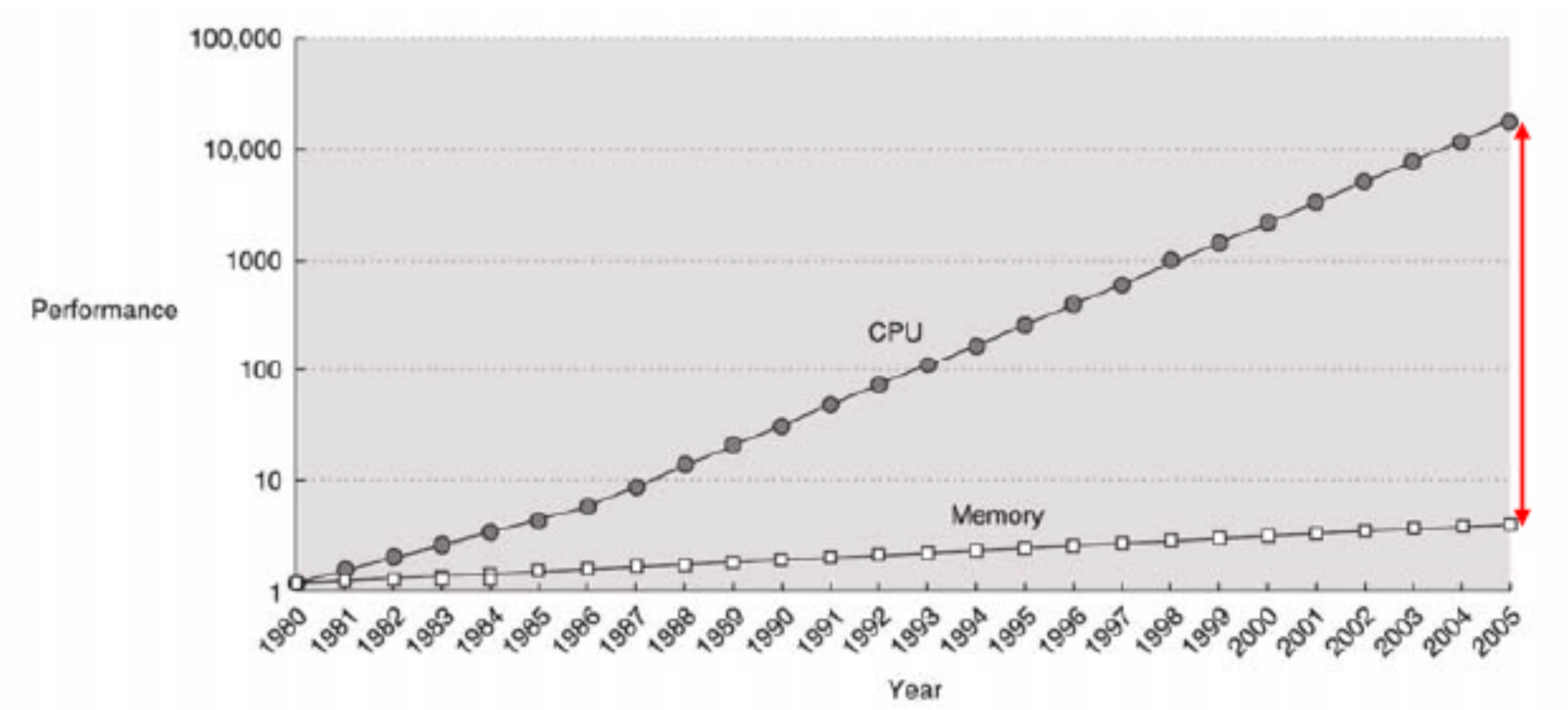
Data & Instructions

Bottleneck

Memory performance gap

CPU speeds increase 25%-30% per year

DRAM speeds increase 2%-11% per year

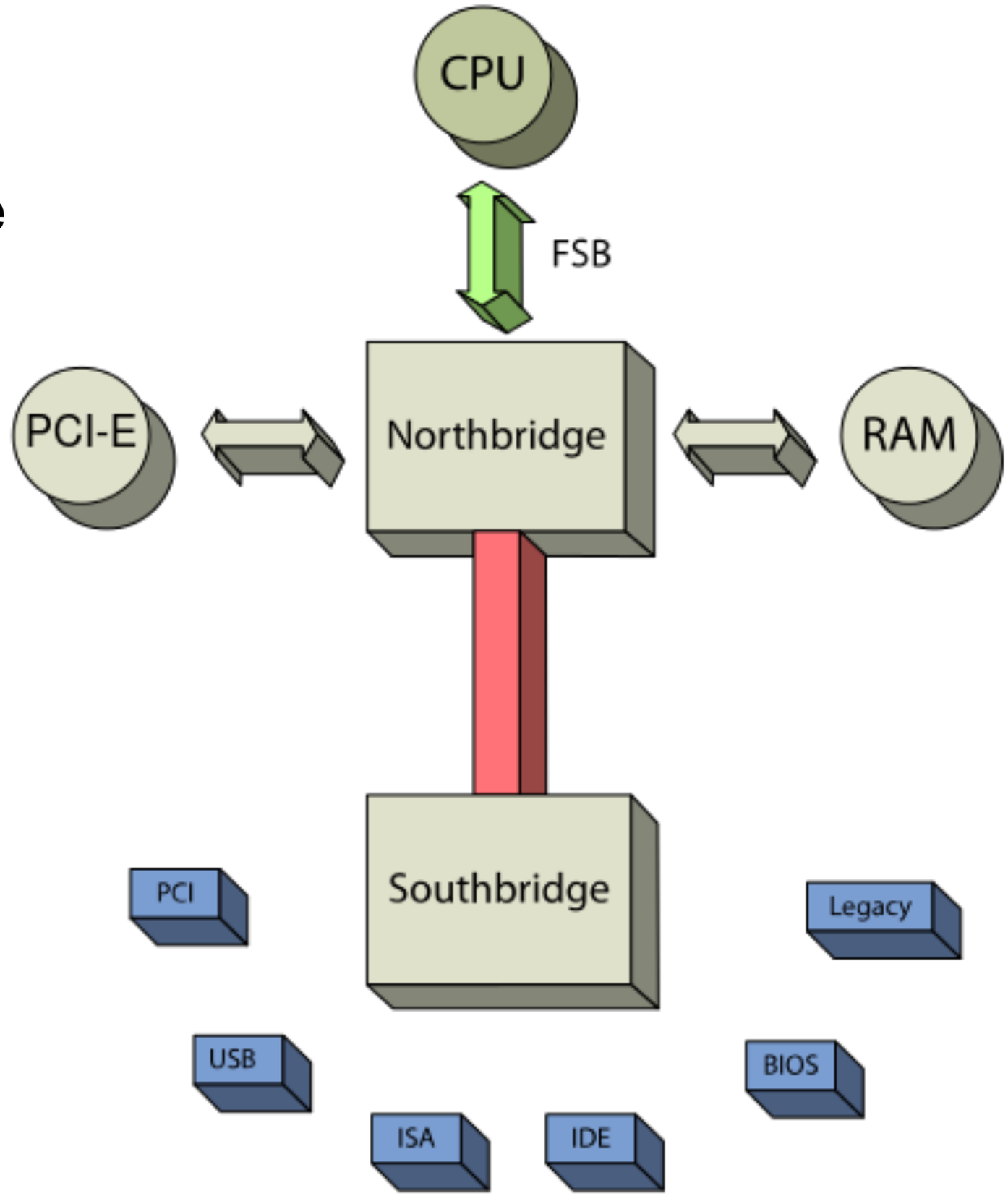


Chipset or System on a Chip

Modern Intel Architecture

3 Components

- CPU
- Northbridge
  - close to CPU
  - rapid access
- Southbridge
  - I/O
- Chipset
  - One chip each
- System on a chip
  - One chip



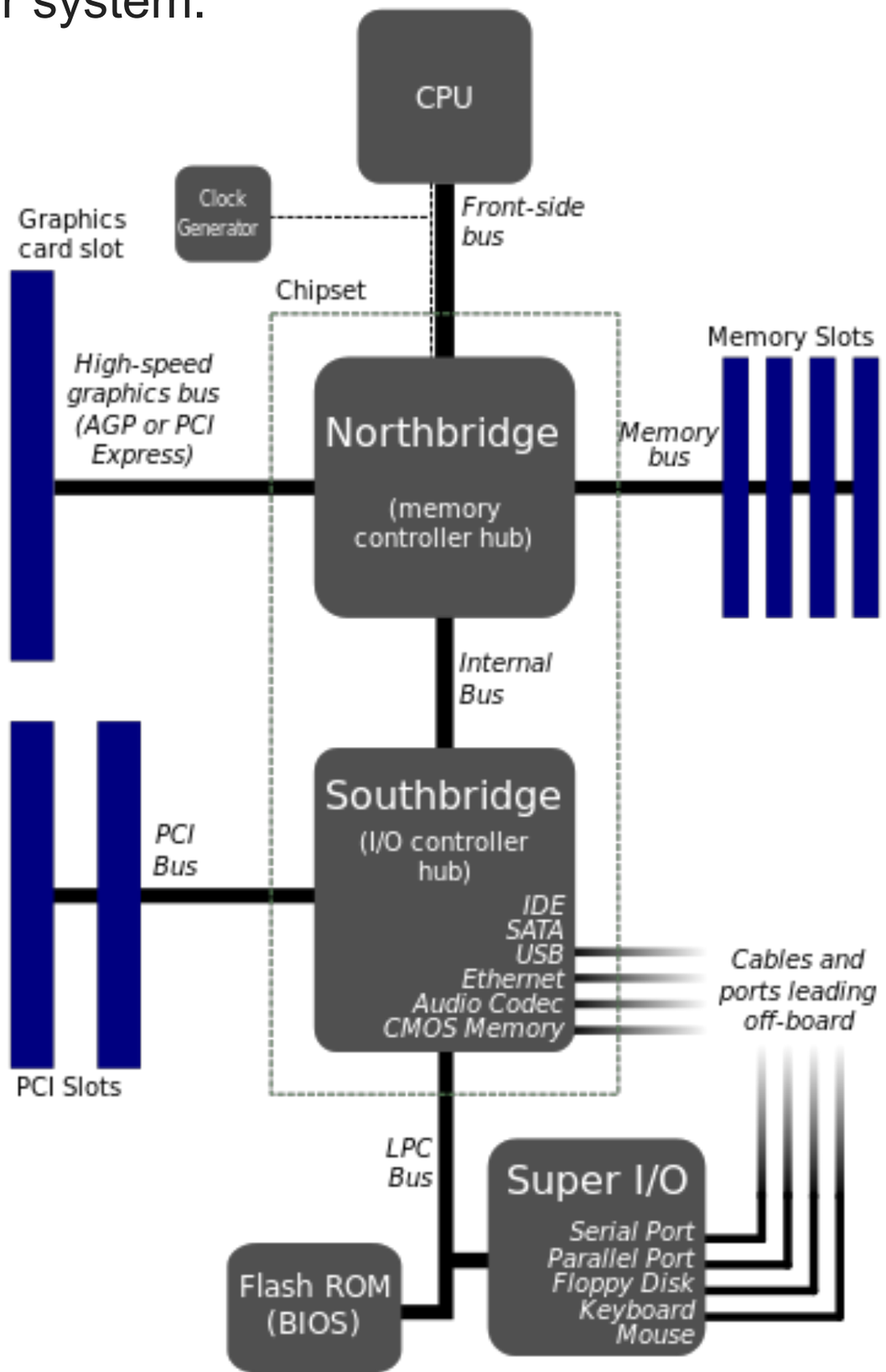
Chipsets

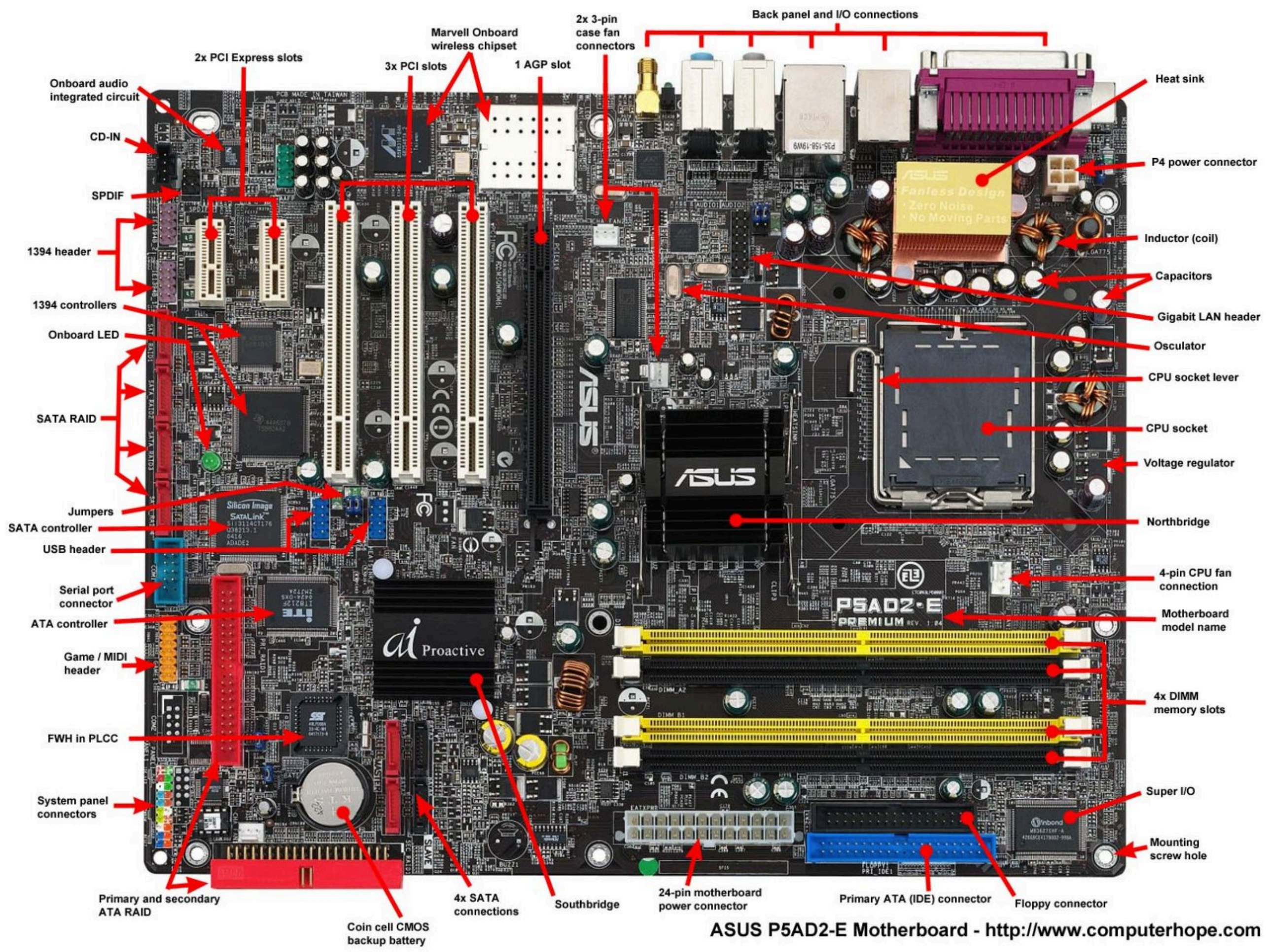
a collection of integrated circuits which are designed to function together as a unit, especially t
perform a particular task within a computer system.

Separate chips

Include Northbridge with CPU

Northbridge, Southbridge and GPU on single chip





ASUS P5AD2-E Motherboard - http://www.computerhope.com

Dell Vostro 1540

Processor: Intel Core i3-370M, 2.4GHz

Memory: 3GB 1,333MHz DDR3 RAM

Graphics: Intel HD Graphics integrated

Hard disk: 320GB

Display: 15.6in 1,366x768, LED-backlit screen

Features: 1.3 megapixel webcam, microphone

Connectivity: 802.11g/n Wi-Fi, Gigabit Ethernet, Bluetooth 3.0

Ports: 3 x USB2, 1 x VGA, 1 x HDMI,

1x 3.5mm headphone output, 1x 3.5mm microphone input, SD/MMC/MS card reader

Dimensions: 376 x 260 x 33mm (WxDxH)

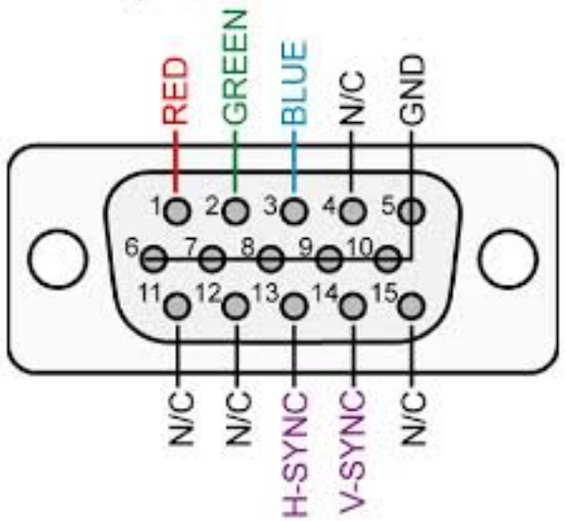
Weight: 2.4kg

Connecting Peripherals

VGA cable

VGA connector

VGA port, view from Wire Side



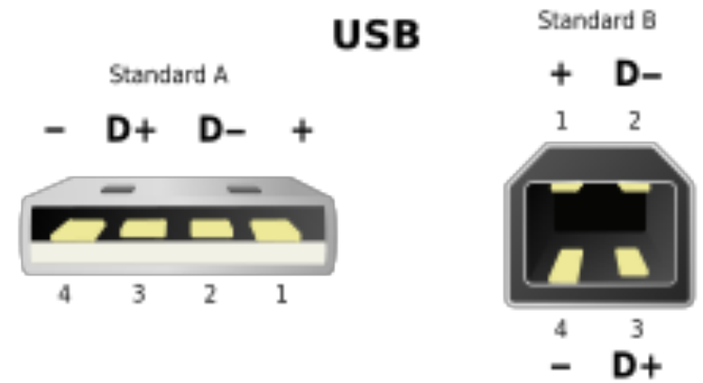
USB

Universal Serial Bus

USB 2.0

< 480 Mb/s (280Mb/sec 35MB/s)

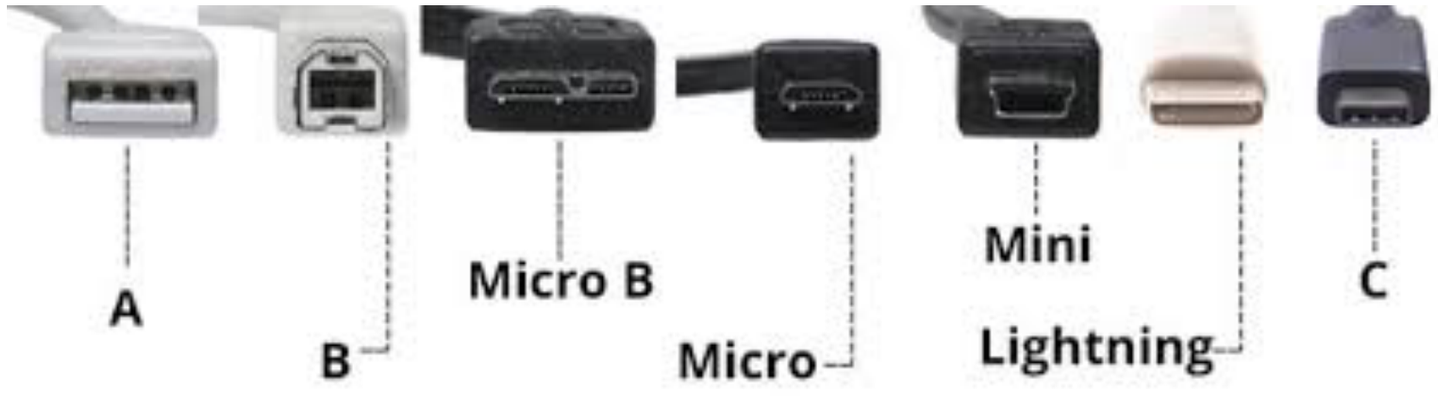
1.5A - 5A charging (where allowed)



USB 3.0

~ 4Gb/s

150mA - 900mA with data, 1.5A without data



Performance metrics

Raw speed

Clock rate

Response time/Execution time

Time between start and completion of event

Bandwith

Bits per second

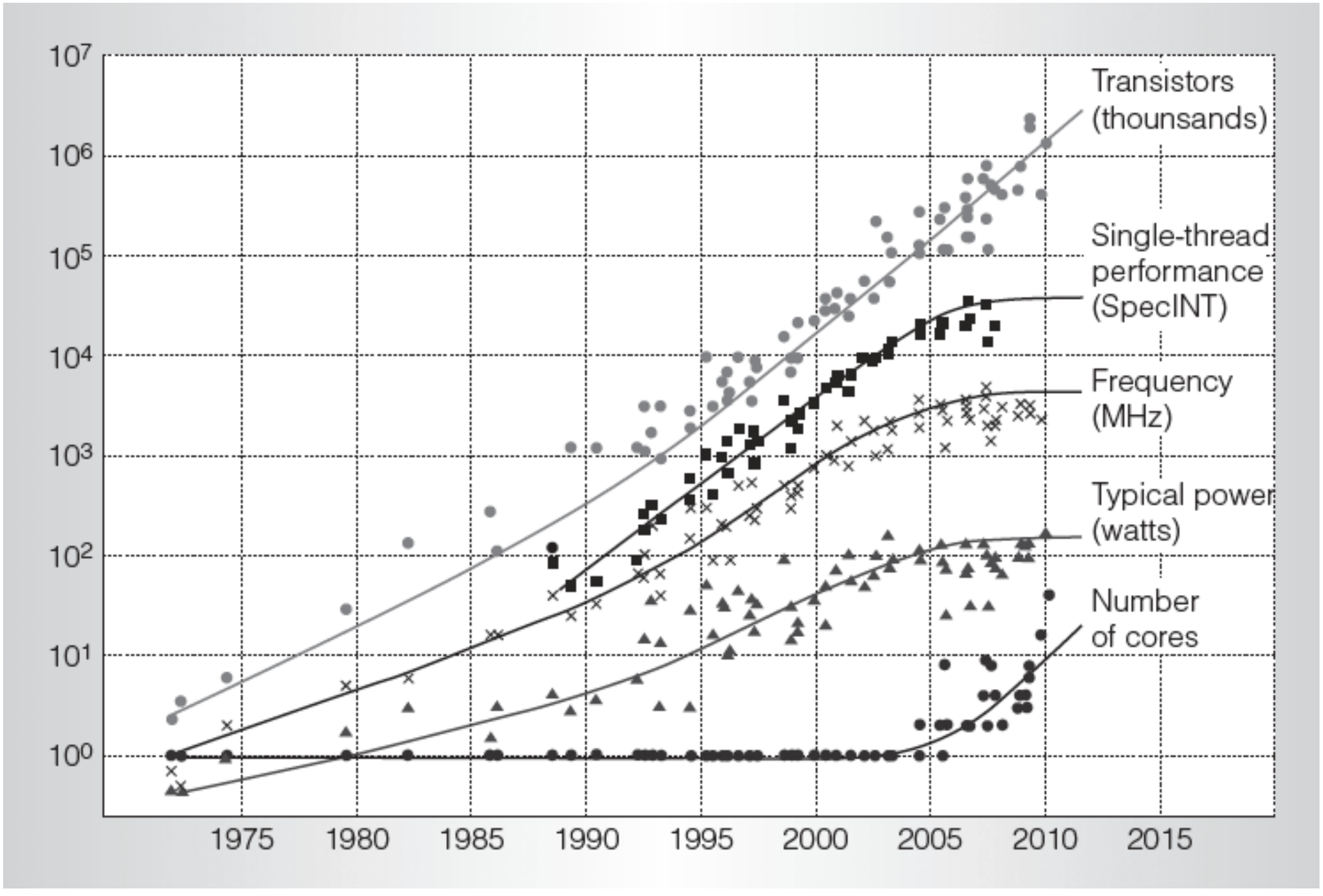
Throughput

Total amount of work done in a given time

Power

Energy per instruction

Key Metrics per CPU



https://www.computer.org/csdl/mags/mi/2010/04/mmi2010040008-abs.html

Real CPUs

understand relationship between instruction sets & assembly

interpret simple x86 machine code instructions

explain the factors that impact on program execution time

explain the difference between CISC and RISC

explain the von Neumann bottleneck