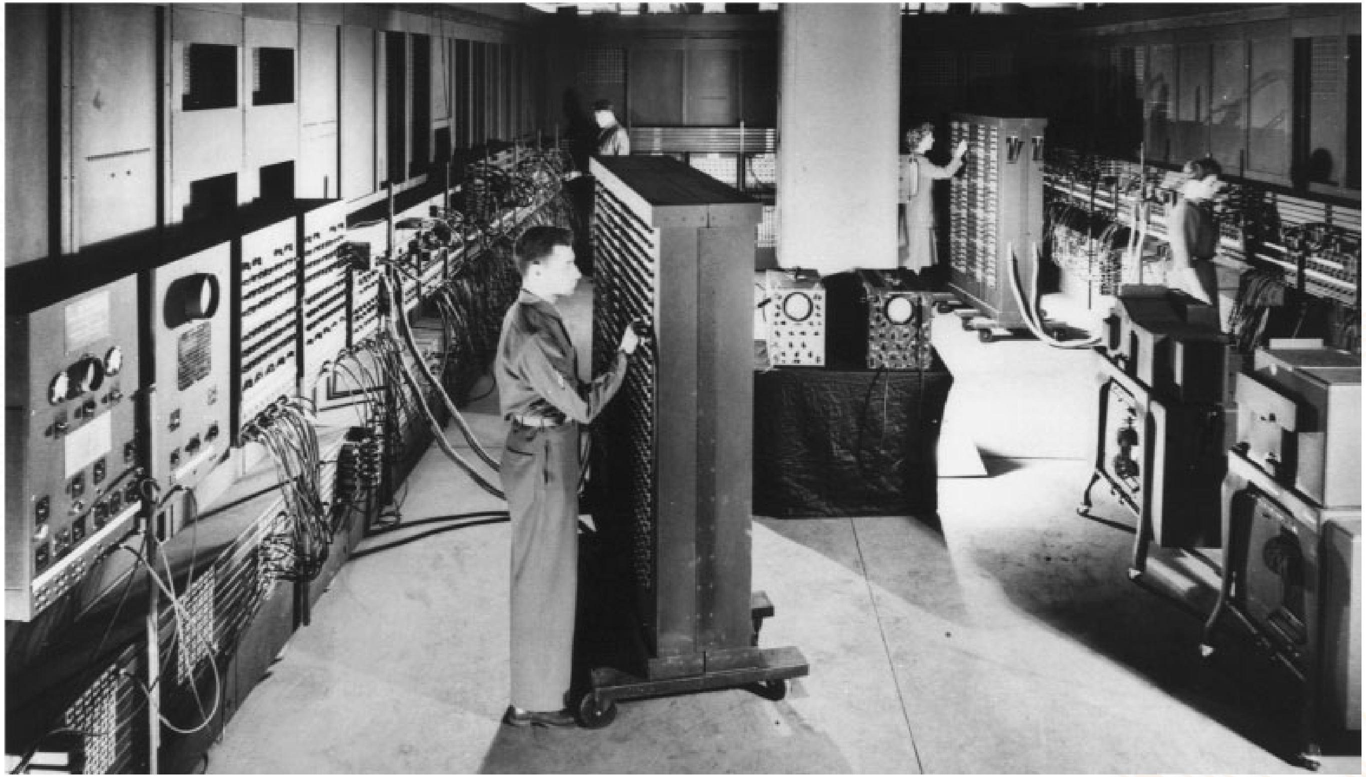
CE/CZ 3001: Advanced Computer Architecture

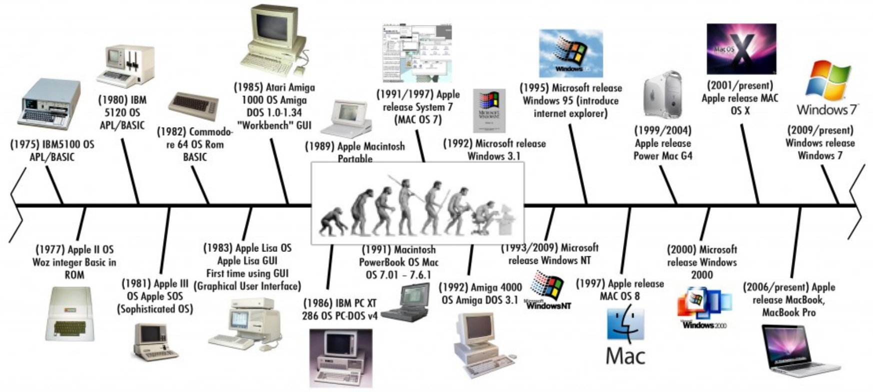
(Module 1: Introduction and Background)

Dr Smitha K. G.
School of Computer Science
And Engineering

Computing devices then…



ENIAC, the world’s first general-purpose electronic computer. (1946)

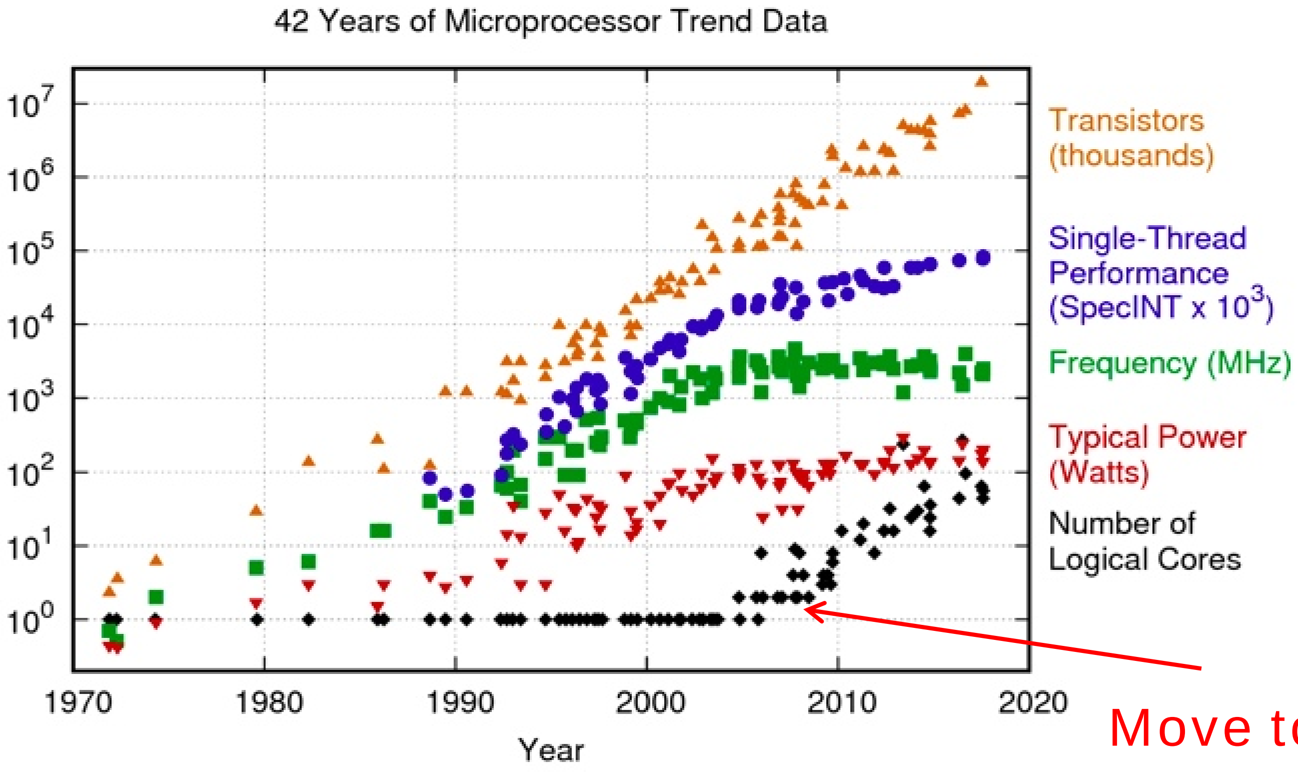

(1975) IBM5100 OS APL/BASIC
(1977) Apple II OS Woz integer Basic in ROM
(1980) IBM 5120 OS APL/BASIC
(1981) Apple III OS Apple SOS (Sophisticated OS)
(1982) Commodore 64 OS Rom BASIC
(1983) Apple Lisa OS Apple Lisa GUI First time using GUI (Graphical User Interface)
(1985) Atari Amiga 1000 OS Amiga DOS 1.0-1.34 "Workbench" GUI
(1986) IBM PC XT 286 OS PC-DOS v4
(1989) Apple Macintosh Portable
(1991) Macintosh PowerBook OS Mac OS 7.01 – 7.6.1
(1991/1997) Apple release System 7 (MAC OS 7)
(1992) Microsoft release Windows 3.1
(1992) Amiga 4000 OS Amiga DOS 3.1
(1993/2009) Microsoft release Windows NT
(1995) Microsoft release Windows 95 (introduce internet explorer)
(1997) Apple release MAC OS 8
Mac
(1999/2004) Apple release Power Mac G4
(2000) Microsoft release Windows 2000
(2001/present) Apple release MAC OS X
(2006/present) Apple release MacBook, MacBook Pro
Windows 7
(2009/present) Windows release Windows 7

Computers lead to the third revolution in our civilization:

Agricultural Age → Industrial Age → Information Age

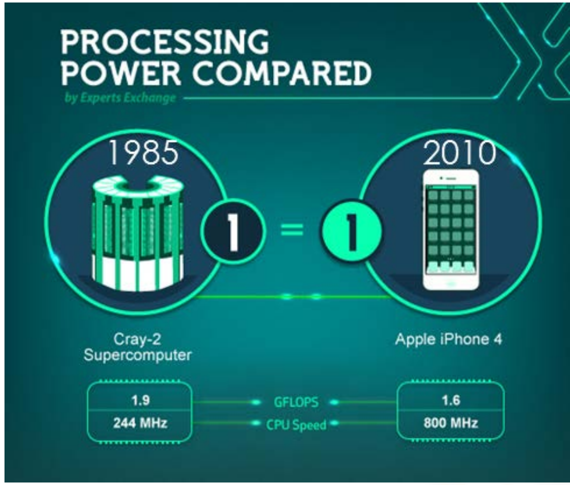
Computing system	Mainframe	Mini computer	Personal computer	Embedded computer
Era	1950s on	1970s on	1980s on	2000s on
Form factor	Multi-cabinet	Multi-board	Single board	Single chip
Resource type	Corporate	Departmental	Family	Personal
Users/system	100s – 1000s	10s – 100s	1s	1/10s
Cost	$ 1 million +	$ 100Ks +	$1Ks – $10Ks	$1s – $100s
Total units	10Ks +	100Ks +	1 billions +	1 Trillions +

Processor Performance



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

https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/



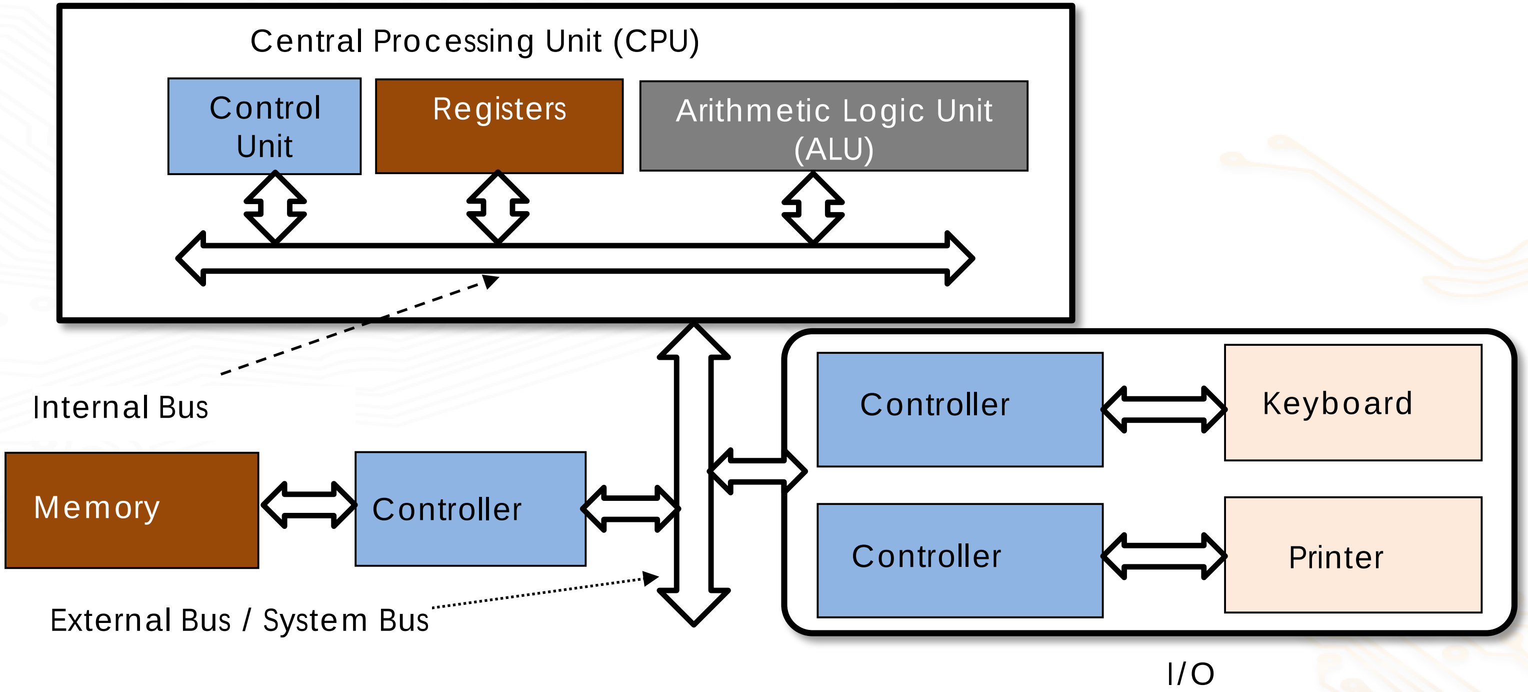
Move to multi-processor

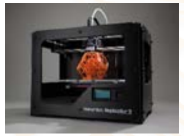
Module 1 Outline

- Review key features of computer architecture.
- Performance metrics and performance enhancement techniques.
- Power dissipation in processors, power metrics, and low-power design techniques.

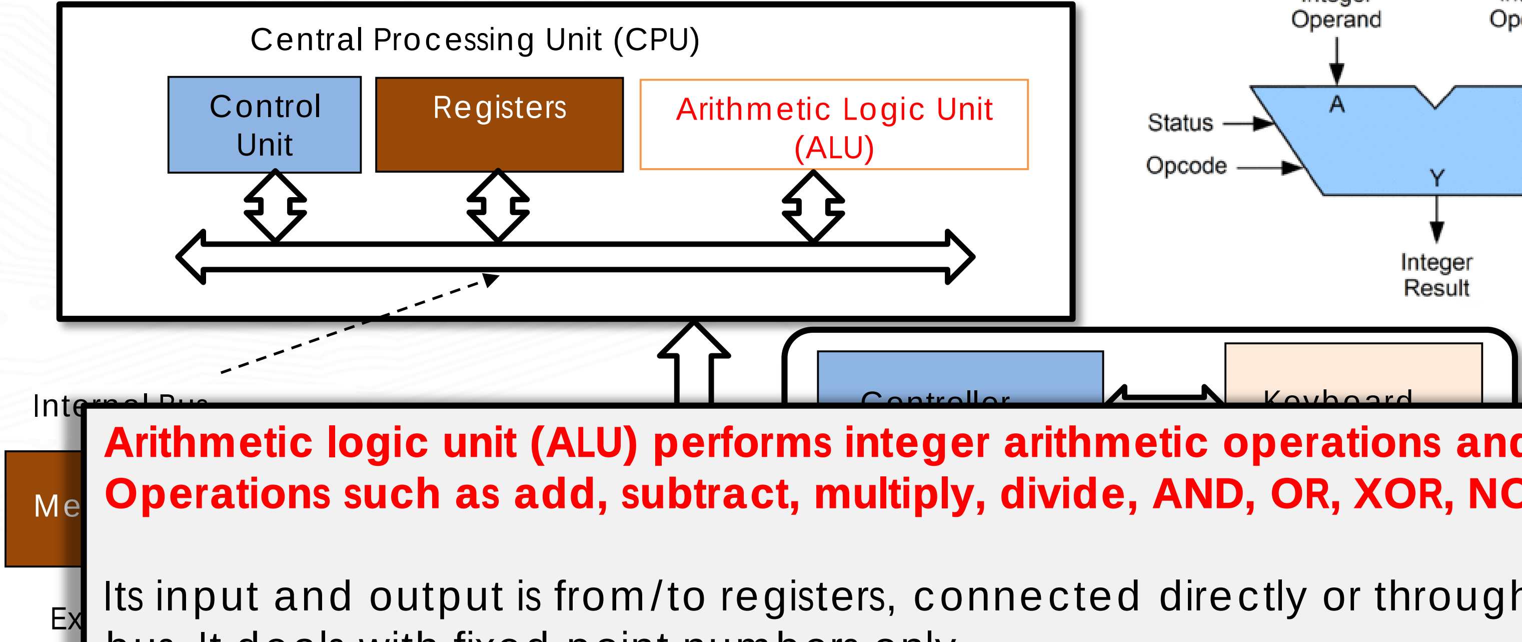
Review key features of computer architecture

5-basic hardware components of a computer

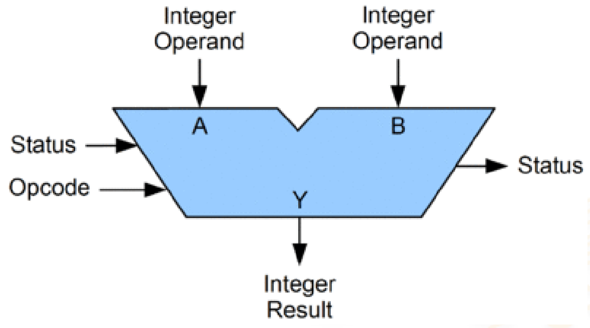




Basic Components: ALU/Computing



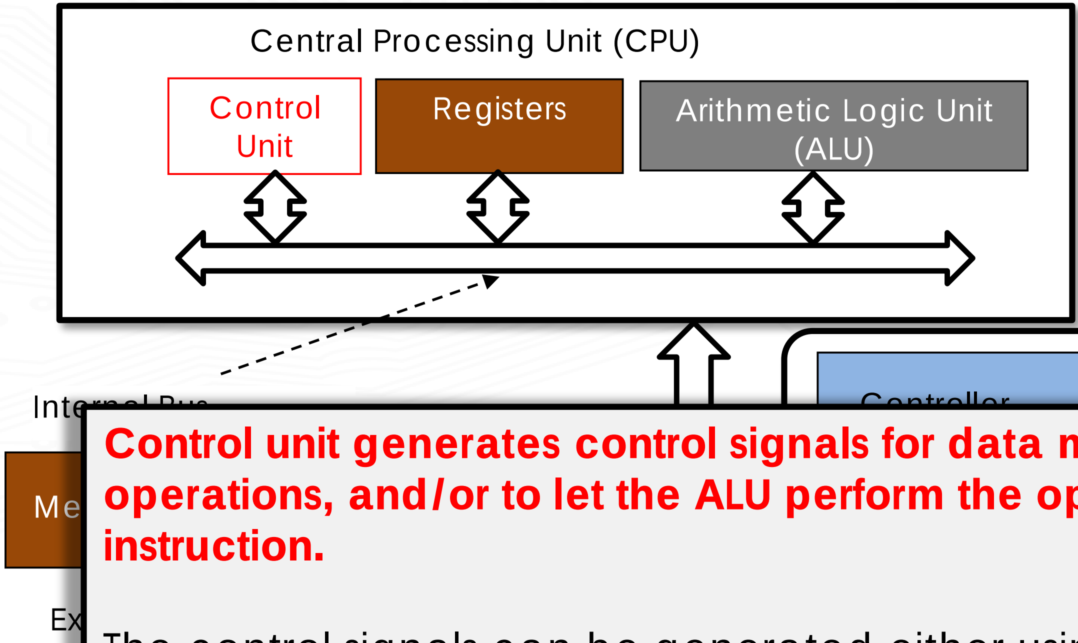
managed by op code

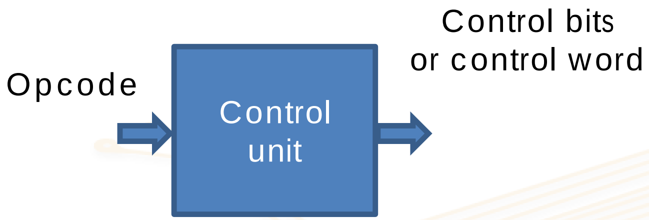


Arithmetic logic unit (ALU) performs integer arithmetic operations and logical Operations such as add, subtract, multiply, divide, AND, OR, XOR, NOT etc.

Its input and output is from/to registers, connected directly or through a fast bus. It deals with fixed point numbers only.

Basic Components: Control

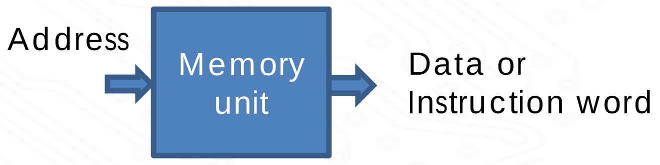


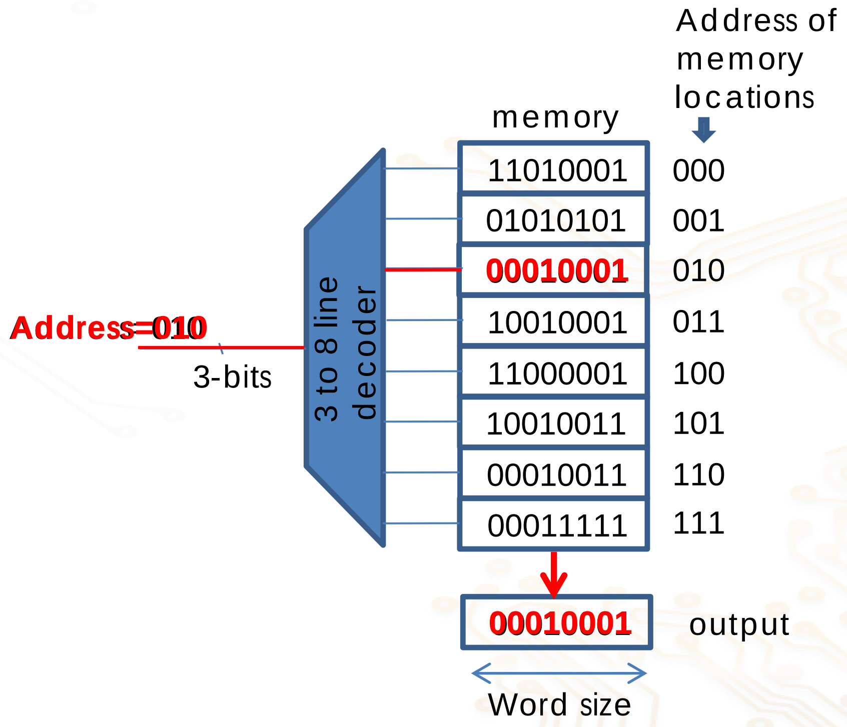


Control unit generates control signals for data movement and data storage operations, and/or to let the ALU perform the operations specified in the instruction.

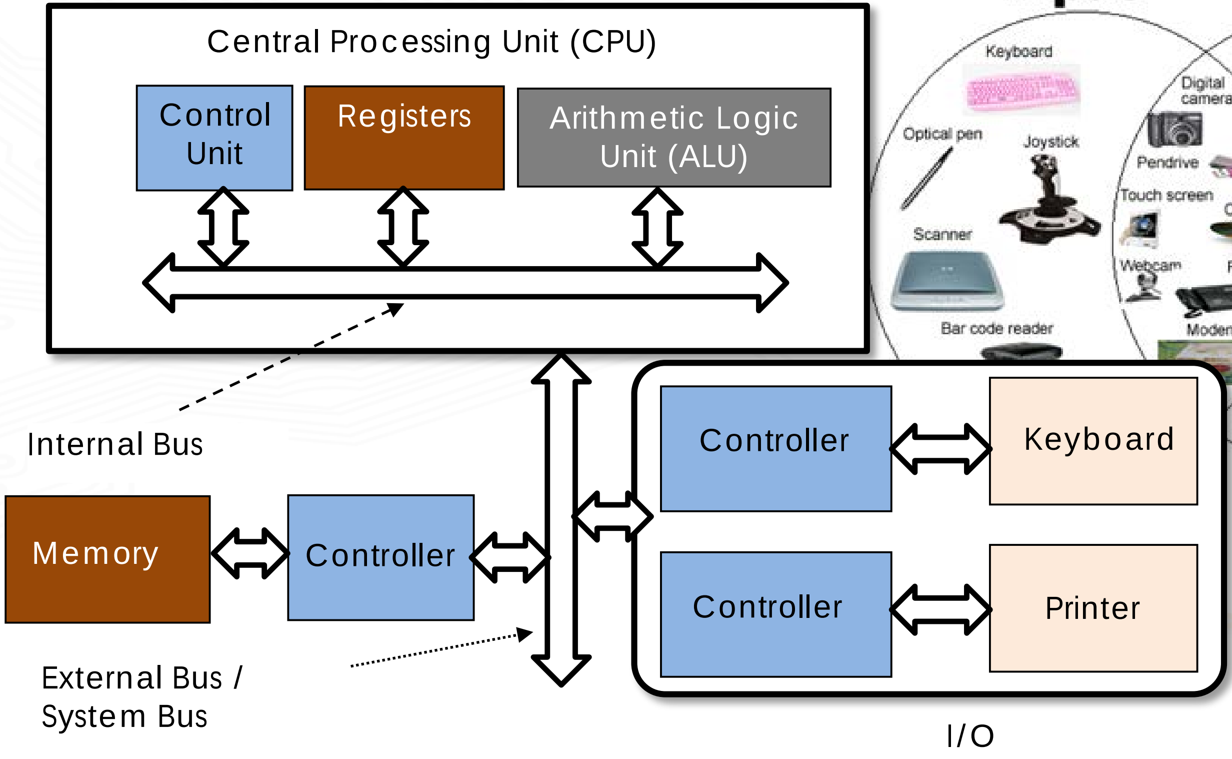
The control signals can be generated either using hardware or using microprogram.

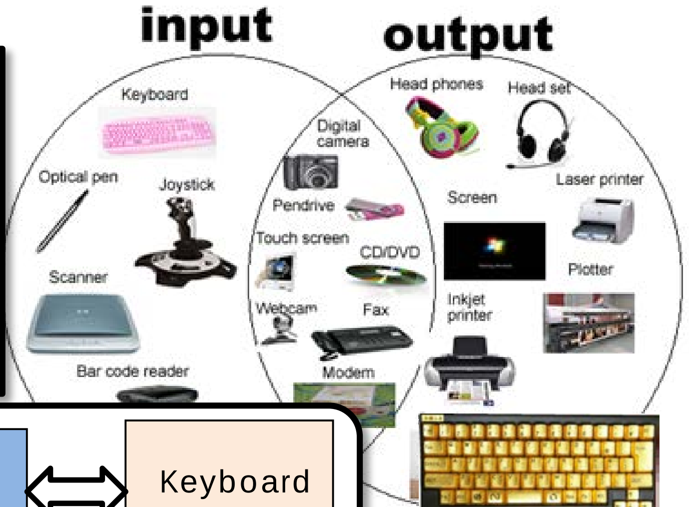
Basic Components: Memory

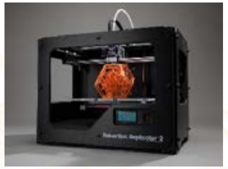




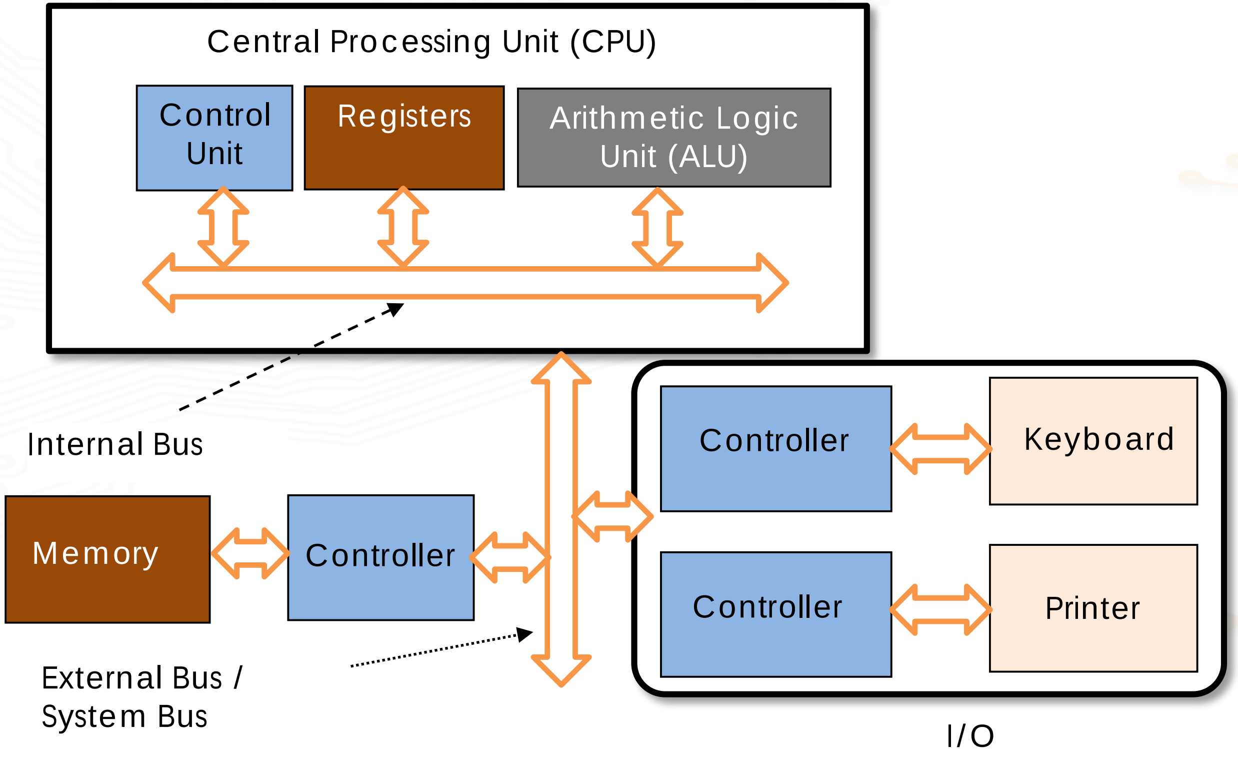
Basic Components: I/O

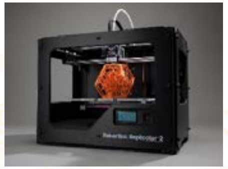






Basic Components: Bus / Communication





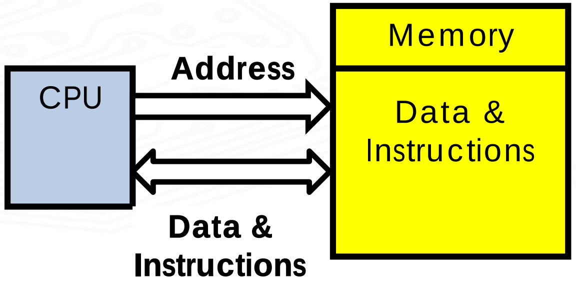
5-basic hardware components of a computer – Summary

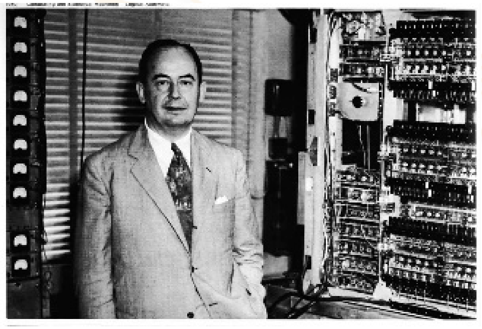
- CPU= ALU + Storage + Control
- Outside CPU = Peripheral + Bus + Storage
- Datapath is a part of CPU where data is processed/ stored/moves through
  - Performs all arithmetic and logical operations
  - Consists of ALU, registers, on-chip cache and internal buses

Overview of Computer Systems (Part 1/2)

Von Neumann Architecture

Memory holds both data and instructions, both are transferred to the CPU through the same bus.





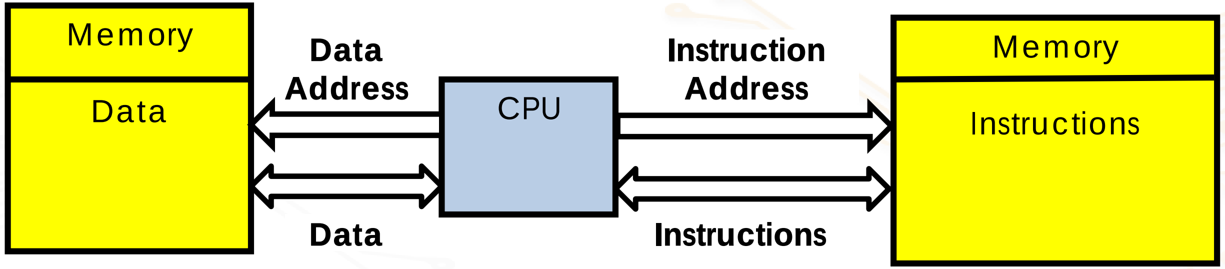
The von Neumann architecture

Von Neumann architecture can perform 1 memory access/clock cycle.

Overview of Computer Systems (Part 2/2)

Harvard Architecture

Separate memory holds data and instructions, each are transferred to the CPU through separate buses.



What is the benefit?

- In Harvard architecture, data access corresponding to the previous instruction can be performed while fetching the current instruction (in the same clock cycle).
- Harvard architecture is widely used in Digital Signal Processors (DSP). Now it's found in most modern processors.

What is computer architecture?

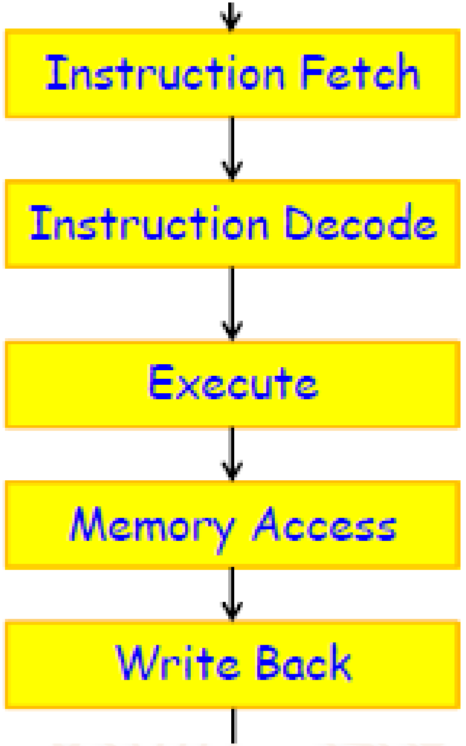
Computer architecture deals with the design and implementation of computer hardware. There are three main subcategories:

- *Instruction Set Architecture*, or ISA: The ISA defines the machine code that a processor reads and acts upon as well as the word size, memory address modes, processor registers, and data type.
- *Microarchitecture*, or *computer organization:* describes implementation of the ISA. It specifies the execution of instruction through the control, storage and computing.
- *System Design:* specifies all of the hardware components within a computing system, their functionalities and interconnection.

Program Execution – Example (Part 1/2)

Instruction cycle: the processing for execution of an instruction.

1. Instruction fetch (IF): Fetch instruction from the memory and get ready to fetch the next Instruction.
2. Instruction decode (ID): Decodes instruction, generate control signals and fetch register operand from register file.
3. Execute (EX): Execute the ALU operation as specified in the opcode of the instruction.
4. Memory access (MA): Perform read/write for load/store operations.
5. Write back (WB): Write the result back to the register file.
6. Go to step 1 to execute the next instruction.



Need for high performance

- Support for better quality of service
  - Support increasing transmission speed
  - Multimedia applications
    - 300 hours of video are uploaded to YouTube every minute!
  - Increasing security need
- Computation-intensive applications
  - DNA sequence analysis
  - Scientific computing
  - Weather forecasting
  - Big data analytics

2.5 quintillion (2.5×10^{18}) bytes of data created everyday — sensors and RFID, social network, digital pictures and videos, purchase records, GPS signals, email communications [IBM bigdata]

How to address this steady growth of computational demand?

Design goals and constraints (Part 1/2)

- Functional requirement: must process data according to the instructions.
  - tests and verifications are required at different stages, since it is not possible to modify the processor once fabricated.
- Reliability: should continue to perform correctly.
  - important for all modern computers and more important for mission critical applications.
  - reliable and fault tolerant computing is an important area of study, which considers various approaches to improve fault tolerance and reliability.

Design goals and constraints (Part 2/2)

- Cost: is a very important factor.
  - embedded consumer products are particularly highly sensitive to cost.
- **Performance**: is a basic requirement.
  - a computing system need to provide the desired performance.
- **Power consumption:** is a very important requirement now a days
  - a system need to consume less power for many reasons: battery life, cost of electricity, thermal problem and cooling cost, etc.

Performance and **power consumption**: Evolution of computer architecture is driven to improve these two goals.

Performance Metrics and Performance Enhancement Techniques

What is performance?

- Performance indicator- $Execution\ time$
- Execution time (CPU time) is the time to execute a program
  - Minimize elapsed time for program = $time_{end} - time_{start}$
  - Called response time (execution time)
  - Less the execution time → better is the performance
  - $Performance = \frac{1}{Execution\ Time}$

Example 1

Consider a program comprising of 100 instructions which runs in two different machines M_1 and M_2. Each instruction takes 2 clock cycles in M_1 and 1 clock cycle in M_2. The clock period of M_1 and M_2 are 20 nanosecond (ns) and 30 ns, respectively. Find the execution times of the program in both machines. Which machine has better performance?

Execution time on M_1 = 100 x 2 x 20 ns = 4000 ns
Execution time on M_2 = 100 x 1 x 30 ns = 3000 ns

$$Performance = \frac{1}{Execution\ Time}$$

Hence Machine M_2 has better performance

Factors affecting execution time

$$Execution\ time = IC \times CPI \times T$$

- Instruction count (IC)
  - Application/program
  - Instruction set architecture (ISA)
- Clocks per instruction (CPI)
  - Instruction set architecture (ISA)
  - Datapath design
  - Parallel and pipelined HW design
- Clock period (T)
  - Semiconductor technology
  - Datapath design and implementation
- Decrease in one may lead to increase in other two.

Challenges on performance Enhancement (Part 1/2)

- Reduction of clock cycle time (T)/ increase clock frequency
  - Power consumption increases with increase in clock frequency
  - Memory operation may take longer than a clock period leading to memory-wall problem

(Memory wall: memory being relatively slower than CPU, makes the CPU wait for data and instructions.)

Challenges on performance Enhancement (Part 2/2)

- Reduction of Instruction count (IC)
  - More complex instructions → CPI will increase
  - Multi-issue processor: VLIW/superscalar, SIMD, and vector processor can reduce instruction count
- Reduction of cycles per instruction (CPI)
  - Instruction pipelining: Pipeline datapath
  - Multi-issue processor: VLIW/superscalar processor

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Example 2

A program has 50 million instructions running at a frequency of 4MhZ

- 10 million branches(CPI= 4)
- 15 million loads(CPI=2)
- 5 million store (CPI=3)
- Rest Add(CPI=1)

Find the execution time

(10 *10^6*4 +15*10^6*2+5*10^6*3+20*10^6*1)/(4*10^6)

(40+30+15+20)/4= 26.25s

What is speedup?

Case A: Consider two computers: computer-A and computer-B speedup of computer-A over computer-B can be computed as

$$Speedup = \frac{Perf_A}{Perf_B} = \frac{Time_B}{Time_A}$$

Case B: Suppose computer-B is an enhanced version of computer-A, achieved by some specific performance enhancement technique, then the speedup achieved due to the enhancement technique can be expressed as

$$Speedup = \frac{T_{unenhanced}}{T_{enhanced}} = \frac{T_{original}}{T_{enhanced}}$$

Speed up

If fraction E of the program is enhanced by a factor of S in an enhanced machine, then determine the speedup of the enhanced machine over the machine before enhancement (original machine). What is the maximum speed up for a given E if S can be any value greater than or equal to 1?

Fraction $U = (1 - E)$ of the program is not enhanced.
if T is the execution time of the program in the unenhanced (original) machine,

Time required for the execution of unenhanced fraction = $T \times (1 - E)$
Time required for the execution of enhanced fraction = $[T \times E]/S$

Total execution time in the enhanced machine

$$T' = [T \times (1 - E)] + [T \times E]/S$$

$T' = T \times \left[(1 - E) + \frac{E}{S}\right] = T \times [1 - E(S - 1)/S]$ *(check: if $S = 1$ then $T' = T$).*

Speed up = $\frac{T}{T'} = \frac{T}{T} / \left[(1 - E) + \frac{E}{S}\right] = \frac{1}{[(1-E)+E/S]}$

Max Speed up

If fraction E of the program is enhanced by a factor of S in an enhanced machine, then determine the speedup of the enhanced machine over the machine before enhancement (original machine). What is the maximum speed up for a given E if S can be any value greater than or equal to 1?

Total execution time in the enhanced machine $T' = [T \times (1 - E)] + \frac{[T \times E]}{S}$
$= T' = T \times [1 - E(S - 1)/S]$ (check: if $S = 1$ then $T' = T$).

If S is very large (i.e., $S \rightarrow \infty$) then $[(S - 1)/S] \rightarrow 1$, and $T' \rightarrow T \times (1 - E)$
So the maximum speedup = $T/T' = T/[T \times (1 - E)] = 1/(1 - E)$
(Note that (1- E) is the unenhanced or sequential fraction of the program.)

If (1 - E)= (1/10) then speedup = 10, i.e., if 1/10th of the program is not-enhanced then, maximum achievable speedup is 10.
If (1 - E)= (1/5) then speedup = 5, i.e., if (1/5)th of the program is not-enhanced then, maximum achievable speedup is 5.
If (1 - E)= 1/2; 1/3 or 1/4 ????

Amdahl’s Law (Part 1/2)

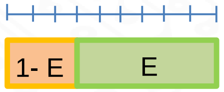
Amdahl’s law: speedup via parallelism is limited by that component of an application which cannot be enhanced (the sequential component)

- If fraction (1 – E) of an application cannot be enhanced for parallel implementation, then the speedup is limited by a factor of 1/(1-E), even if the rest of the application is infinitely sped up, and involve infinitesimal time for the computation.

check: if 20% of an application cannot be enhanced for parallel implementation, then the speedup is limited by a factor of 100/20 = 5.
check: if x% of an application cannot be enhanced for parallel implementation, then the speedup is limited by a factor of 100/x.

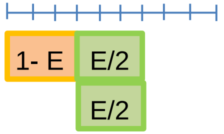
Amdahl’s Law (Part 2/2)

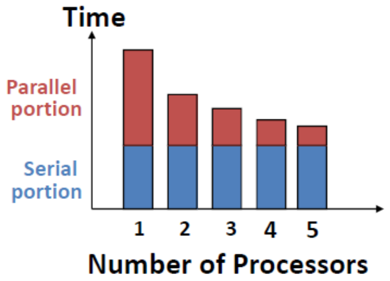
The line with the delimiters on at the top is the total time T(1).



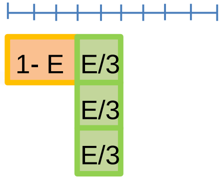
1- E = unenhanced fraction (serial)
E = enhanced fraction (parallel)

parallelization factor of 2





parallelization factor of 3



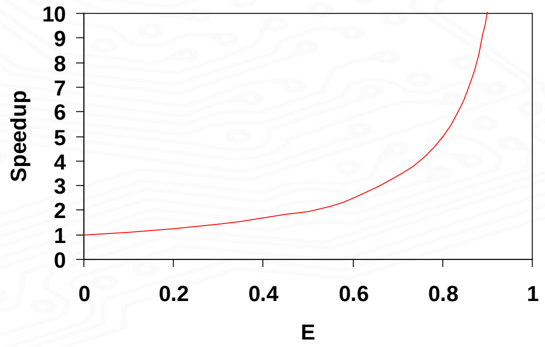
Amdahl’s law: speedup via parallelism is limited by that component of an application which cannot be enhanced (the sequential component).

Amdahl’s Law – Limit

E is the fraction of the program , enhanced by a factor of S.

- Make common case fast:

$$\lim_{s \to \infty} \frac{1}{1 - E + \frac{E}{S}} = \frac{1}{1 - E}$$



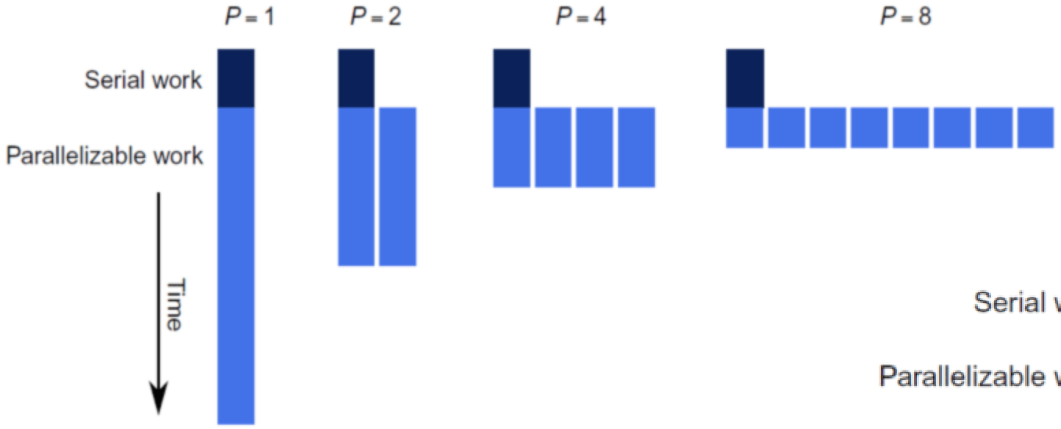
E	s	Speedup
95%	1.10	1.094
5%	10	1.047
5%	∞	1.052

$$Speedup_{max} = \frac{1}{1 - E}$$

Embarrassingly Parallel Problems

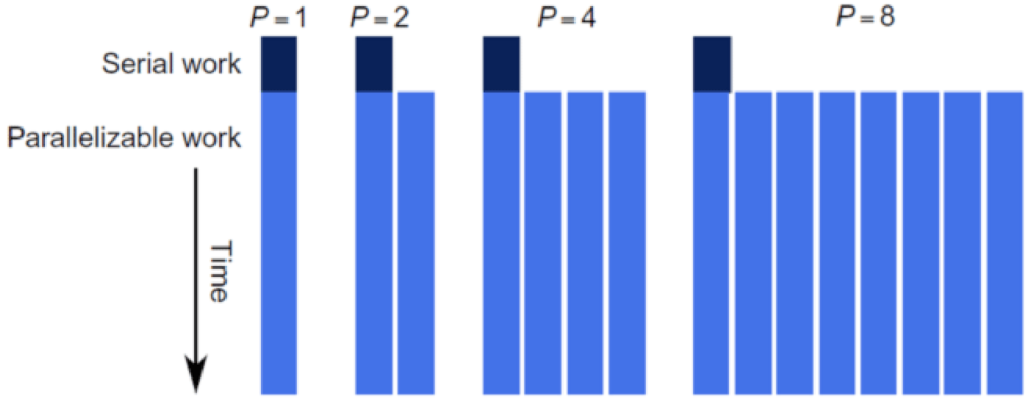
Amdahl's vs Gustafson's Law

The goal for optimizing may be to make a program run faster with the same workload (reflected in Amdahl's Law) or to run a program in the same time with a larger workload (Gustafson Law).

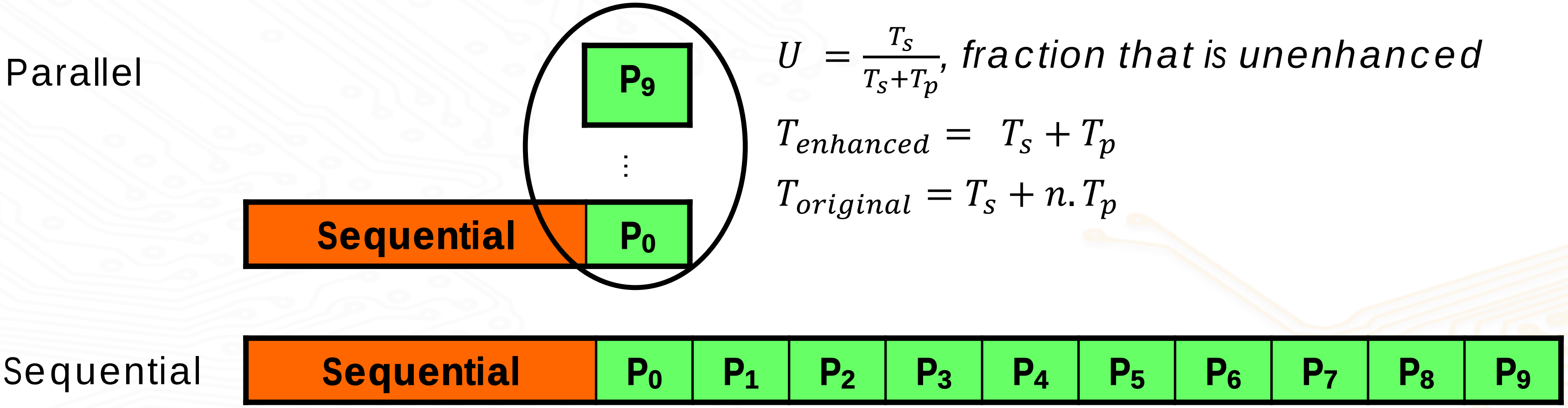


Amdahl's law-fixed workload

Gustafson's law-fixed time



Gustafson's Law – Fixed Time



$U = \frac{T_s}{T_s+T_p}$, *fraction that is unenhanced*

$T_{enhanced} = T_s + T_p$

$T_{original} = T_s + n.T_p$

$$Speedup(U,n) = \frac{T_{original}}{T_{enhanced}} = \frac{T_s + n.T_p}{T_s + T_p} = n - U(n-1)$$

→ Targets **embarrassingly parallel problems**

Example 3

A processor is running a program which has 50 billion instructions and the clock frequency of the processor is 2Ghz. The CPI and % of instruction types in program are given in the following table. Calculate the overall speed up when CPI of branch is improved from 4 to 2.

Instruction Type	% in program	CPI
R-type	40	1
branch	20	4
Load	30	2
Store	10	3

Speed up = $\frac{Time\ original}{T\ enhanced}$

Execution time before enhancement=
(0.4*1+0.2*4+0.3*2+0.1*3)*50 * 10^9/(2*10^9)
(0.4+0.8+0.6+0.3)*25= 52.5

Execution time after enhancement=
(0.4*1+0.2*2+0.3*2+0.1*3)*50 * 10^9/(2*10^9)
(0.4+0.4+0.6+0.3)*25= 42.5

Speed up=52.5/42.5=1.24

Performance Metrics

- Million of Instructions Per Second (MIPS): depends on how it is evaluated:
    - Native MIPS,
    - Peak MIPS, and
    - Relative MIPS.

$$Instruction\ Per\ Second\ (IPS)\ =\ \frac{Instruction\ Count}{Execution\ Time}$$

Other Performance Metrics (Part 1/3)

- Million of Instructions Per Second (MIPS): depends on how it is evaluated:
  - Native MIPS,
  - Peak MIPS, and
  - Relative MIPS.

Instruction count in terms of millions of instructions

$$Native\ MIPS = \frac{Instruction\ Count}{Execution\ Time \times 10^6}$$

Peak MIPS is obtained by choosing a sequence of instructions (i.e, an instruction mix) which could provide the maximum MIPS.

Other Performance Metrics (Part 2/3)

Relative MIPS: Estimated relative to an agreed-upon reference machine (e.g. Vax 11/780)

$$Relative\ MIPS = \frac{T_{ref}}{T_{machine\_to\_be\_rated}} \times MIPS_{ref}$$

T_{ref} : Execution time of reference machine
$T_{machine\_to\_be\_rated}$: Execution time machine to be rated

- MIPS varies with
  - the ISA (i.e., the complexity of instructions)
  - the choice of instruction mix (program)
- Higher MIPS does not guarantee better performance (instruction complexity)
- Relative MIPS is useful to rate evolving designs of the same computer

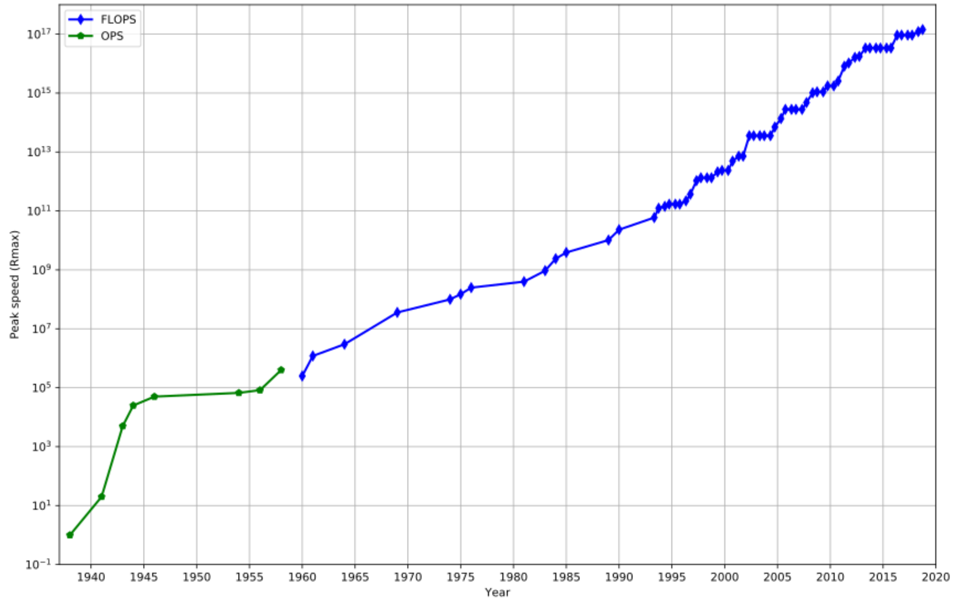
Other Performance Metrics (Part 3/3)

$$FLOPS = \frac{Number\ of\ Floating\ point\ Operations}{Execution\ Time\ (in\ seconds)}$$

- FLOPS is used for machines used in fields of scientific calculations.
- Timeline
  - June 2007: IBM Blue Gene supercomputer has a peak of 596 teraFLOPS (performs 596 trillion FLOPS.)
    - *used to simulate approximately 1% of a human cerebral cortex*
  - On June 20, 2017, China's Sunway TaihuLight was ranked the world's fastest with 93.01 petaflops on the Linpack benchmark (out of 125.4 peak petaflops).
  - Currently IBM SUMMIT, with 122.3petaflops with peak at 148.6 petaflops

Mega	$= 10^6$
Giga	$= 10^9$
Tera	$= 10^{12}$
Peta	$= 10^{15}$
Exa	$= 10^{18}$

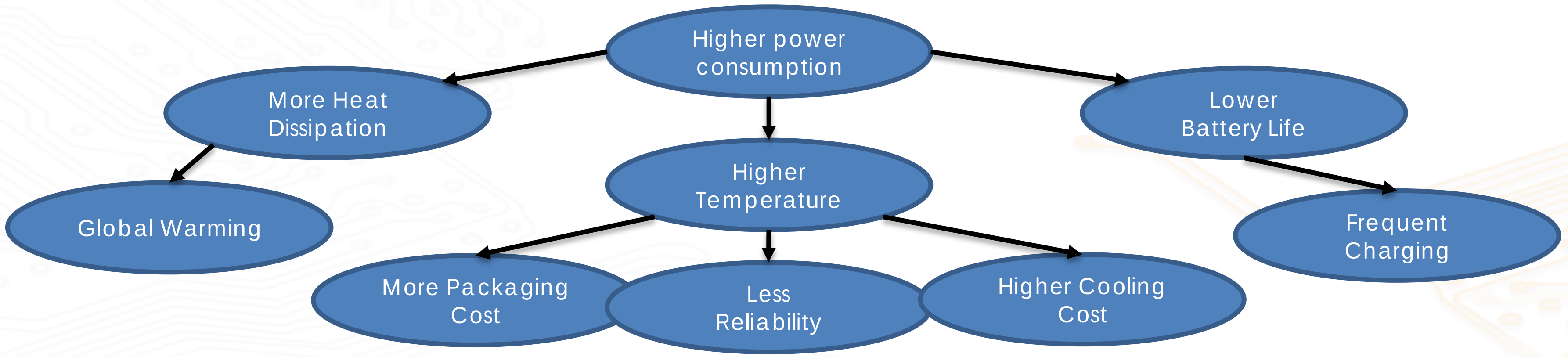
Year	Supercomputer	Peak speed (Rmax)	Location
2018	IBM Summit	122.3 PFLOPS	Oak Ridge, U.S.
2016	Sunway TaihuLight	93.01 PFLOPS	Wuxi, China
2013	NUDT Tianhe-2	33.86 PFLOPS	Guangzhou, China
2012	Cray Titan	17.59 PFLOPS	Oak Ridge, U.S.
2012	IBM Sequoia	17.17 PFLOPS	Livermore, U.S.
2011	Fujitsu K computer	10.51 PFLOPS	Kobe, Japan
2010	Tianhe-IA	2.566 PFLOPS	Tianjin, China
2009	Cray Jaguar	1.759 PFLOPS	Oak Ridge, U.S.
2008	IBM Roadrunner	1.026 PFLOPS 1.105 PFLOPS	Los Alamos, U.S.



Top supercomputer speeds: logscale speed over 60 years

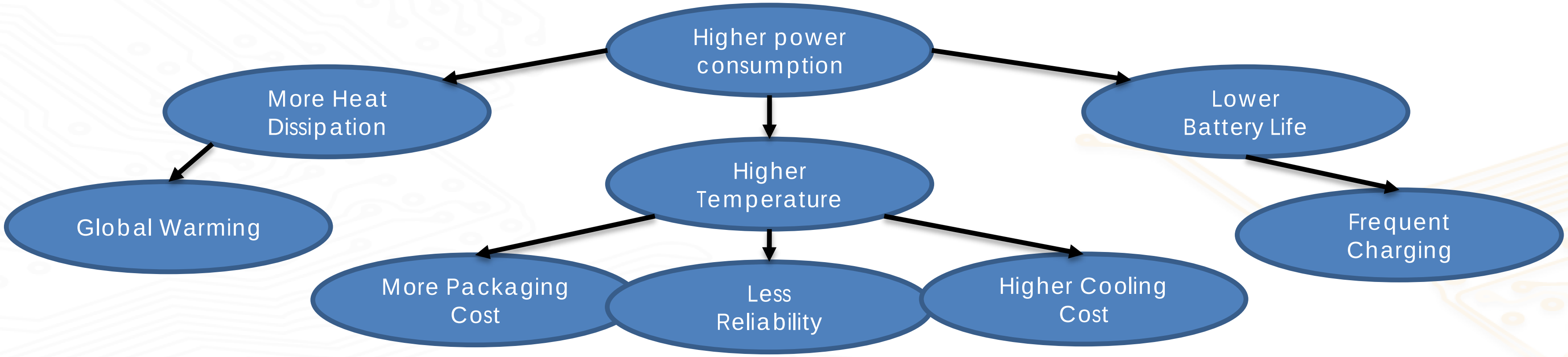
Power dissipation in processors, power metrics, and low-power design techniques

Need for reducing power consumption (Part 1/2)



- More power dissipation makes the device unreliable
  - Power → temperature rise → temperature induced effects in device functionalities.
  - More computation-intensive applications in portable devices with growing computing power.

Need for reducing power consumption (Part 2/2)



- Increasing number of battery operated devices like hand phone and tablets. As energy consumption increases battery life decreases.

High- performance with less power consumption is required for all kinds of computers (just not portable devices).

Power dissipation in a processor : Where and How?

- Power is dissipated in logic, in memory, and interconnects

- Power dissipation in logic devices
  - Dynamic power: dissipated only when computation is performed
  - Static (leakage) power: is due to the leakage current, and dissipated whenever the system is powered-on even if no computation is done

Components of power dissipation in processor (Part 1/4)

- **Power dissipation**
  - Dynamic power: (P_{dyn}) dissipated only when processor executes instruction.
    - It increases with the operating voltage (V) and clock frequency (f).
    - The faster we compute, more is the dynamic power.
  - Static (leakage) power: (P_{st}) is due to the leakage current, and dissipated whenever the system is powered-on even if no computation is done.
    - It is independent of clock frequency.
    - It increases with temperature of the processor.

Components of power dissipation in processor (Part 2/4)

- **Dynamic power consumption, $P_{dyn} = ACV^2f$**

  C: total load capacitance in the circuit
  f: clock frequency

  Reduction of frequency reduces P_{dyn}, but can degrade performance

  V : operating voltage (also called V_{dd})

  Reduction of voltage reduces power consumption significantly

  A: switching activity factor: the fraction of transistors switch during a clock cycle (in average).

  Can be reduced by turning-off the unused resources/components

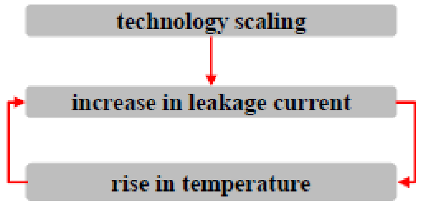
Components of power dissipation in processor (Part 3/4)

- **Static power consumption, $P_{st} = VIleak$**

  I_{leak} : Leakage current
  V : operating voltage (also called V_{dd})

- **Static power dissipation behaviour:**

  Leakage current increases cumulatively with temperature.



Cumulative Increase of Leakage Current with Increase of Temperature

Total power dissipation in processor

- Total power consumption = $ACV^2f + V\,Ileak$
  - Both dynamic and static power are reduced by voltage reduction
- Maximum operating frequency $f_{max} \propto [V - V_{th}]^2/V$,
  - V_{th} is called the threshold voltage, the gate-source voltage at which the transistor just starts conducting
  - if V_{th} is small compared to V, the maximum usable frequency $f_{max} \propto V$

What are the best methods for total power reduction?

Reducing power consumption – Voltage and frequency scaling

- $P_{dyn} = ACV^2 f, Pdyn \propto V^2$, Voltage reduction can result in considerable saving of power
- Can we reduce power consumption by reducing only frequency?

  Yes
- Can we reduce energy consumption by reducing only frequency?

  No
- We can reduce power consumption by reducing only frequency, but with the same degradation of performance.
- Energy consumption remains the same even as reduction in clock frequency increases the clock period.

$$\left(E = \int_{t=0}^{T} P_{avg}\, dt\right)$$

Reducing power consumption

- Reduction of energy/power consumption by
  - Component design, e.g., efficient cache and memory hierarchy design
  - Power gating: shutting down the unused components
  - Clock gating: to reduce unnecessary switching
  - Reducing the data movement, number of memory access, and register transfer

Trevor Mudge, Power: a first-class architectural design constraint, IEEE Computer, April 2001, pp.52-58.
Jan M. Rabaey, Low Power Design Essentials, Springer, LLC 2009

Example 4

Consider a processor while working at its maximum operating clock frequency of 500 MHz consumes 80 Watt dynamic power and 10 Watt static power. It consumes 540 kJ of energy for a given computation. If the leakage current is decreased by 10% by reduction of temperature and operating clock frequency is reduced to half what will be the energy consumptions due to static power and dynamic power consumptions.

Total power consumption = 80 + 10 = 90 Watts.
Execution time = 540 K/90 = 6000 seconds.
New static power = 10 -1 = 9 Watts.
New dynamic power = 80/2 = 40 Watts.
New execution time = 12000 seconds.
Energy consumption due to new static power = 9 × 12000 Joules = 108 kJ.
Energy consumption due to new dynamic power = 40 × 12000 Joules = 480 kJ.

Summary

- Review key features of computer architecture.
- Performance metrics and performance enhancement techniques.
- Power dissipation in processors, power metrics, and low-power design techniques.