Chapter 1

COMPUTER SYSTEM PERFORMANCE

Why study computer organization and architecture?

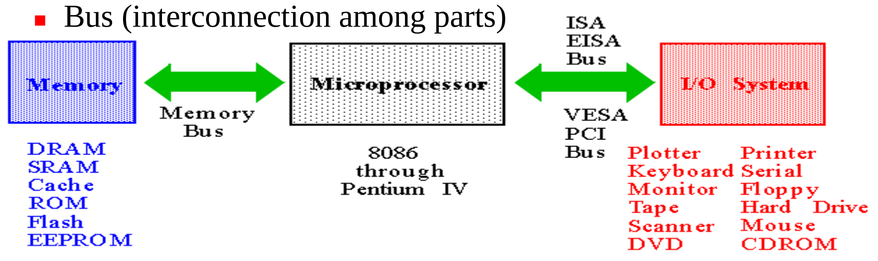
- Design better programs, including system software such as compilers, operating systems, and device drivers.
- Optimize program behavior.
- Evaluate (benchmark) computer system performance.
- Understand time, space, and price tradeoffs.

- Computer organization
  - Encompasses all physical aspects of computer systems.
  - E.g., circuit design, control signals, memory types.
  - *How does a computer work?*
- Computer architecture
  - Logical aspects of system implementation as seen by the programmer.
  - E.g., instruction sets, instruction formats, data types, addressing modes.
  - *How do I design a computer?*

Computer Components

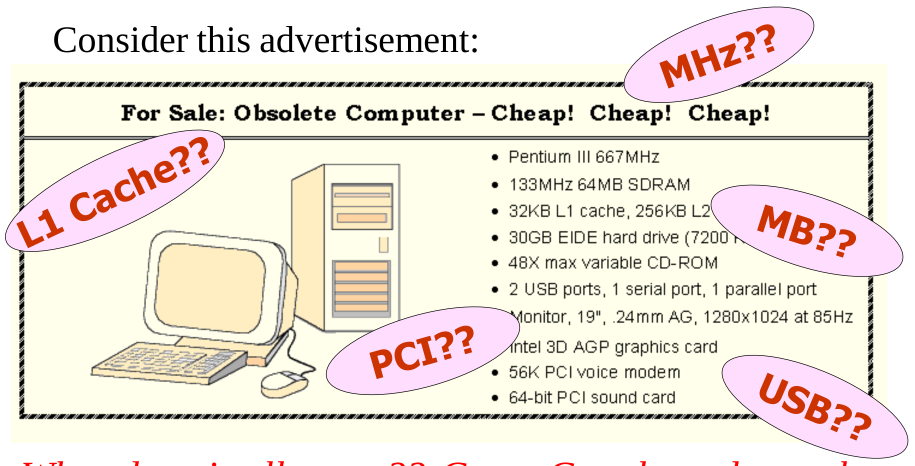
- There is no clear distinction between matters related to computer organization and matters relevant to computer architecture.
- Principle of Equivalence of Hardware and Software:
  - ***Anything that can be done with software can also be done with hardware, and anything that can be done with hard ware can also be done with software.***

- At the most basic level, a computer is a device consisting of four pieces:
  - A processor to interpret and execute programs
  - A memory to store both data and programs
  - A mechanism for transferring data to and from the outside world, include IO devices.
  - Bus (interconnection among parts)



An Example System

Consider this advertisement:



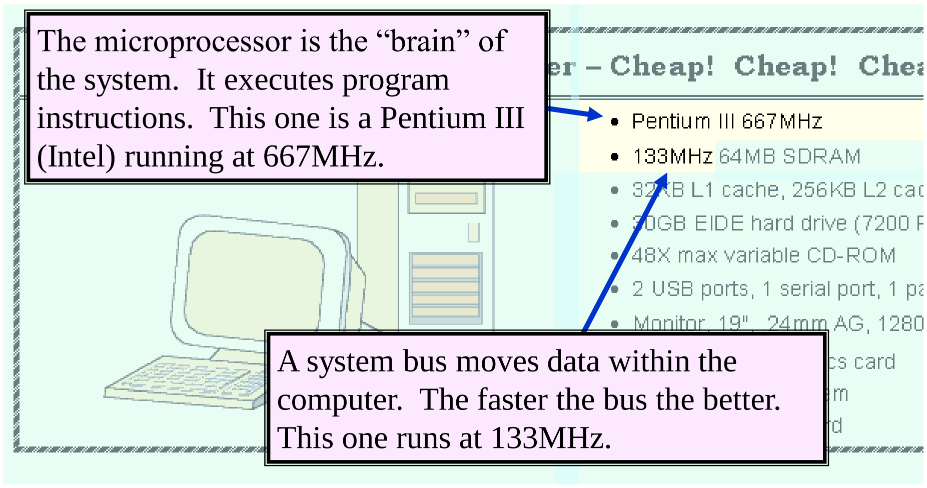
For Sale: Obsolete Computer – Cheap! Cheap! Cheap!

- Pentium III 667MHz
- 133MHz 64MB SDRAM
- 32KB L1 cache, 256KB L2
- 30GB EIDE hard drive (7200 r
- 48X max variable CD-ROM
- 2 USB ports, 1 serial port, 1 parallel port
- Monitor, 19", .24mm AG, 1280x1024 at 85Hz
- Intel 3D AGP graphics card
- 56K PCI voice modem
- 64-bit PCI sound card

What does it all mean?? Go to Google and search for PCI or USB port.

6

An Example System


er – Cheap! Cheap! Chea
Pentium III 667MHz
133MHz 64MB SDRAM
32KB L1 cache, 256KB L2 cac
30GB EIDE hard drive (7200 R
48X max variable CD-ROM
2 USB ports, 1 serial port, 1 pa
Monitor, 19", .24mm AG, 1280
cs card
m
rd

The microprocessor is the “brain” of the system. It executes program instructions. This one is a Pentium III (Intel) running at 667MHz.

A system bus moves data within the computer. The faster the bus the better. This one runs at 133MHz.

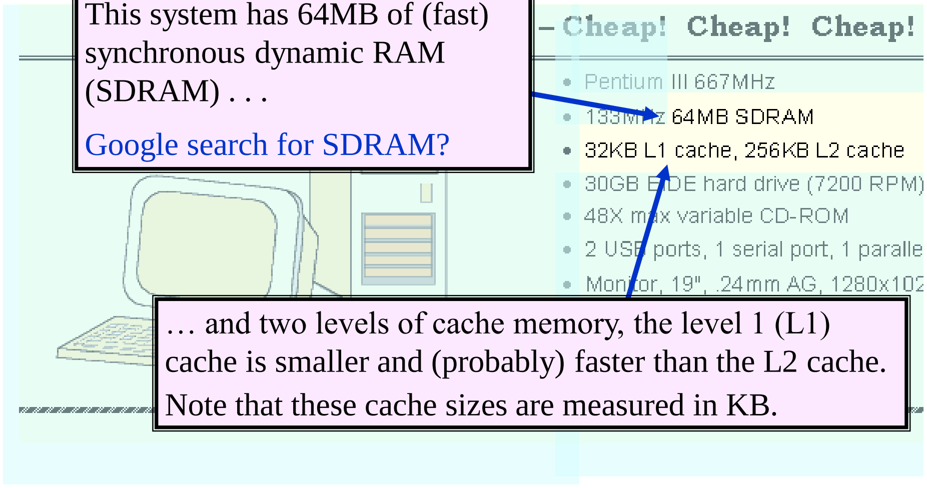
7

An Example System

- Computers with large main memory capacity can run larger programs with greater speed than computers having small memories.
- RAM is an acronym for random access memory. Random access means that memory contents can be accessed directly if you know its location.
- Cache is a type of temporary memory that can be accessed faster than RAM.

8

An Example System



– Cheap! Cheap! Cheap!

- Pentium III 667MHz
- 133MHz 64MB SDRAM
- 32KB L1 cache, 256KB L2 cache
- 30GB EIDE hard drive (7200 RPM)
- 48X max variable CD-ROM
- 2 USB ports, 1 serial port, 1 paralle
- Monitor, 19", .24mm AG, 1280x102

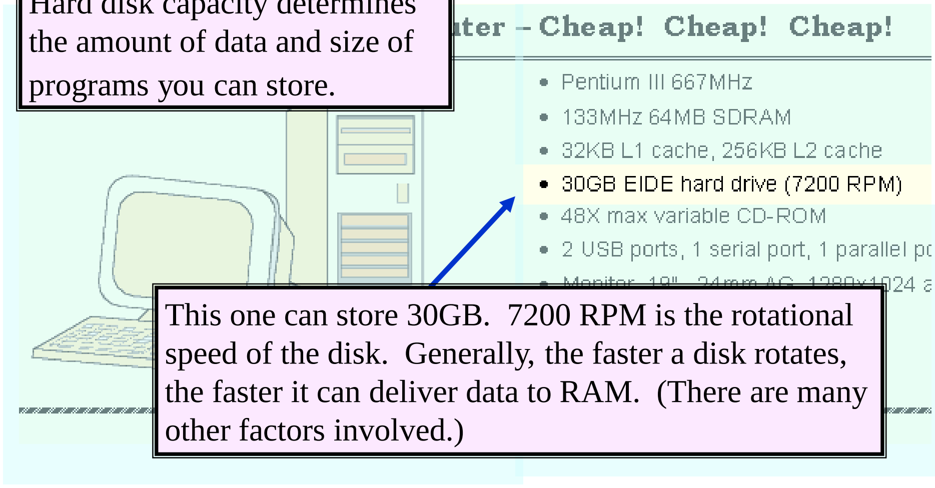
This system has 64MB of (fast) synchronous dynamic RAM (SDRAM) . . .

Google search for SDRAM?

… and two levels of cache memory, the level 1 (L1) cache is smaller and (probably) faster than the L2 cache. Note that these cache sizes are measured in KB.

9

An Example System



uter – Cheap! Cheap! Cheap!

- Pentium III 667MHz
- 133MHz 64MB SDRAM
- 32KB L1 cache, 256KB L2 cache
- 30GB EIDE hard drive (7200 RPM)
- 48X max variable CD-ROM
- 2 USB ports, 1 serial port, 1 parallel pc
- Monitor, 19", .24mm AG, 1280x1024 a

Hard disk capacity determines the amount of data and size of programs you can store.

This one can store 30GB. 7200 RPM is the rotational speed of the disk. Generally, the faster a disk rotates, the faster it can deliver data to RAM. (There are many other factors involved.)

10

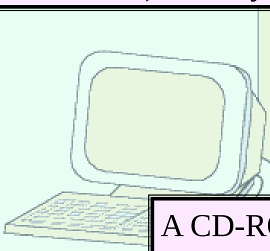
An Example System

EIDE stands for *enhanced integrated drive electronics*, which describes how the hard disk interfaces with (or connects to) other system components.

! Cheap!

AM

- 32KB L1 cache, 256KB L2 cache
- 30GB EIDE hard drive (7200 RPM)
- 48X max variable CD-ROM
- 2 USB ports, 1 serial port, 1 parallel pc
- Monitor, 19", .24mm AG, 1280x1024 a
- Intel 3D AGP graphics card



A CD-ROM can store about 650MB of data, making it an ideal medium for distribution of commercial software packages. 48x describes its speed.

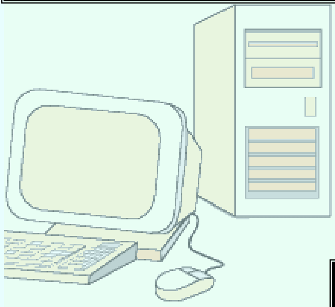
11

An Example System

Ports allow movement of data between a system and its external devices.

Cheap! Cheap! Cheap!

- Pentium III 667MHz
- 133MHz 64MB SDRAM
- 32KB L1 cache, 256KB L2 cache
- 30GB EIDE hard drive (7200 RPM)
- 48X max variable CD-ROM
- 2 USB ports, 1 serial port, 1 parallel port
- Monitor, 19", .24mm AG, 1280x1024 at 85Hz
- Intel 3D AGP graphics card
- e modem
- und card



This system has four ports.

12

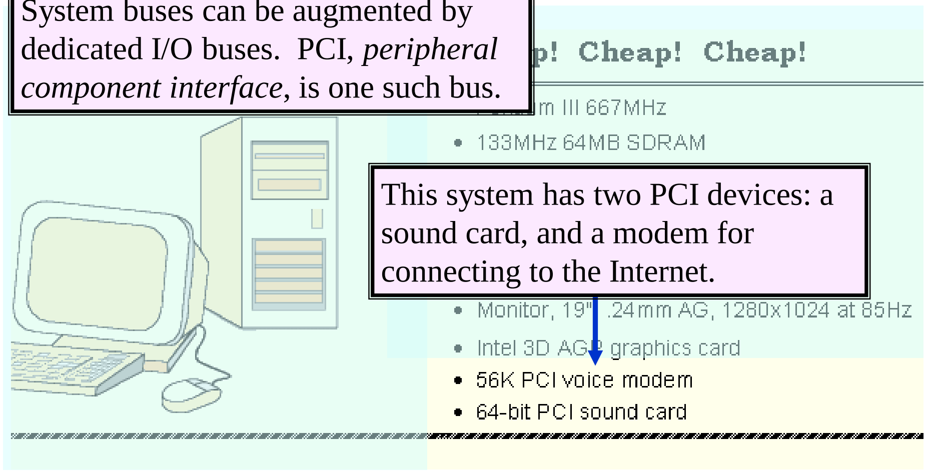
An Example System

- Serial ports send data as a series of pulses along one or two data lines.
- Parallel ports send data as a single pulse along at least eight data lines.
- USB, universal serial bus, is an intelligent serial interface that is self-configuring. (It supports "plug and play.")

13

An Example System

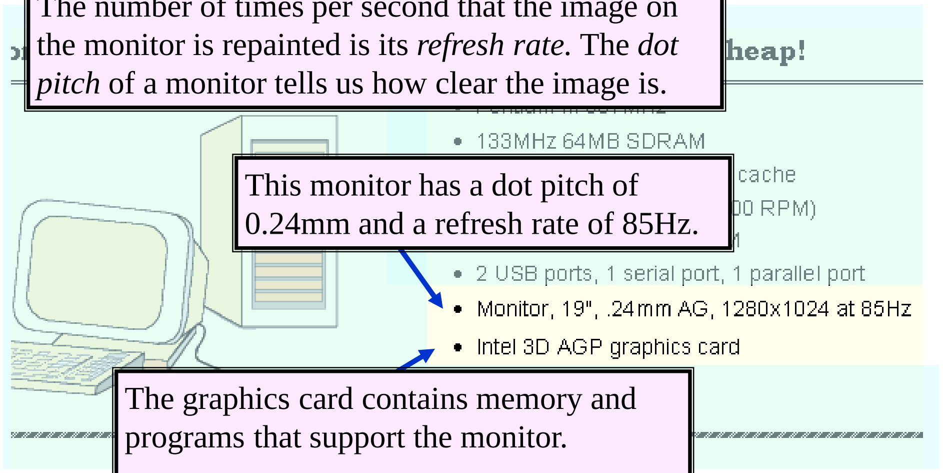
System buses can be augmented by dedicated I/O buses. PCI, *peripheral component interface,* is one such bus.



This system has two PCI devices: a sound card, and a modem for connecting to the Internet.

14

An Example System


heap!
• 133MHz 64MB SDRAM
cache
00 RPM)
• 2 USB ports, 1 serial port, 1 parallel port
• Monitor, 19", .24mm AG, 1280x1024 at 85Hz
• Intel 3D AGP graphics card

The number of times per second that the image on the monitor is repainted is its *refresh rate*. The *dot pitch* of a monitor tells us how clear the image is.

This monitor has a dot pitch of 0.24mm and a refresh rate of 85Hz.

The graphics card contains memory and programs that support the monitor.

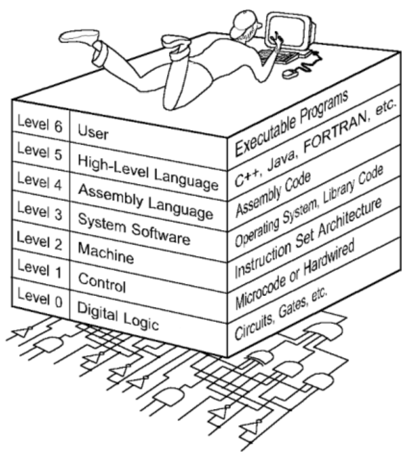
Google search for AGP?

15

The computer level hierachy

- Computers consist of many things besides chips.
- Before a computer can do anything worthwhile, it must also use software.
- Writing complex programs requires a “divide and conquer” approach, where each program module solves a smaller problem.
- Complex computer systems employ a similar technique through a series of virtual machine layers.

- Each virtual machine layer is an abstraction of the level below it.
- The machines at each level execute their own particular instructions, calling upon machines at lower levels to perform tasks as required.
- Computer circuits ultimately carry out the work.



- Level 6: The User Level
  - Program execution and user interface level.
  - The level with which we are most familiar.
- Level 5: High-Level Language Level
  - The level with which we interact when we write programs in languages such as C, Pascal, Lisp, and Java.
- Level 4: Assembly Language Level
  - Acts upon assembly language produced from Level 5, as well as instructions programmed directly at this level.

- Level 3: System Software Level
  - Controls executing processes on the system.
  - Protects system resources.
  - Assembly language instructions often pass through Level 3 without modification.
- Level 2: Machine Level
  - Also known as the Instruction Set Architecture (ISA) Level.
  - Consists of instructions that are particular to the architecture of the machine.
  - Programs written in machine language need no compilers, interpreters, or assemblers.

- Level 1: Control Level
  - A *control unit* decodes and executes instructions and moves data through the system.
  - Control units can be *microprogrammed* or *hardwired*.
  - A microprogram is a program written in a low-level language that is implemented by the hardware.
  - Hardwired control units consist of hardware that directly executes machine instructions.
- Level 0: Digital Logic Level
  - This level is where we find digital circuits (the chips).
  - Digital circuits consist of gates and wires.
  - These components implement the mathematical and control logic of all other levels.

The von Neumann Model

- Inventors of the ENIAC, John Mauchley and J. Presper Eckert, conceived of a computer that could store instructions in memory.
- The invention of this idea has since been ascribed to a mathematician, John von Neumann, who was a contemporary of Mauchley and Eckert.
- Stored-program computers have become known as von Neumann Architecture systems.

21

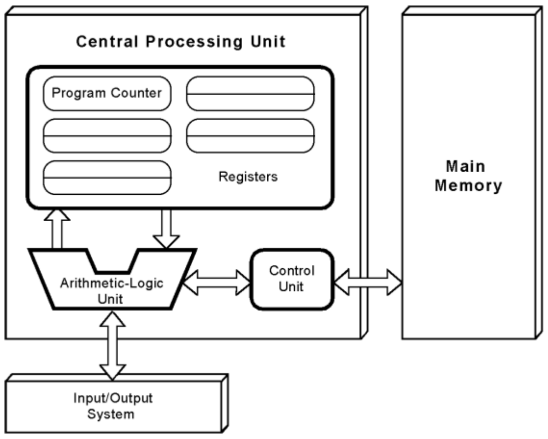
The von Neumann Model

- Today’s stored-program computers have the following characteristics:
  - Three hardware systems:
    - A central processing unit (CPU)
    - A main memory system
    - An I/O system
  - The capacity to carry out sequential instruction processing.
  - A single data path between the CPU and main memory.
    - This single path is known as the *von Neumann bottleneck*.

22

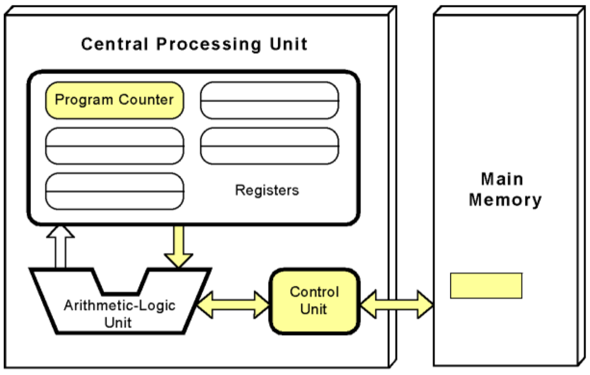
The von Neumann Model

- This is a general depiction of a von Neumann system:
- These computers employ a fetch-decode-execute cycle to run programs as follows . . .



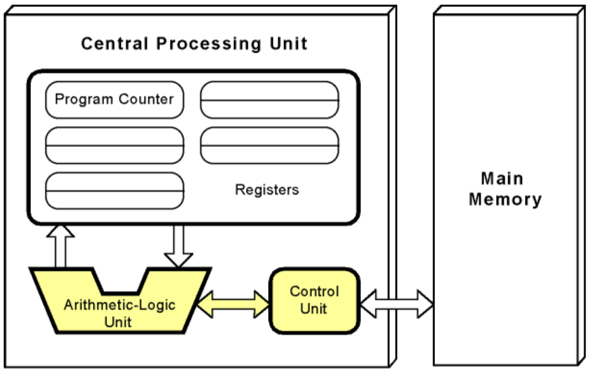
The von Neumann Model

- The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.



The von Neumann Model

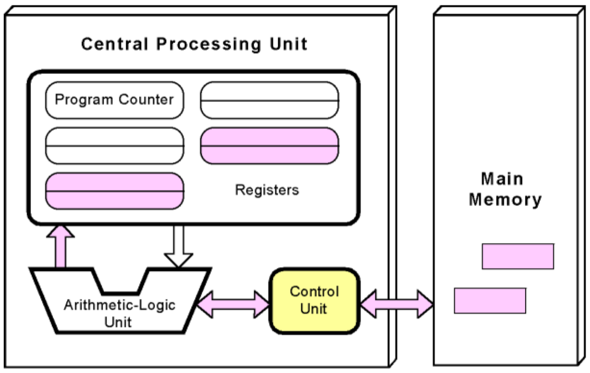
- The instruction is decoded into a language that the ALU can understand.



25

The von Neumann Model

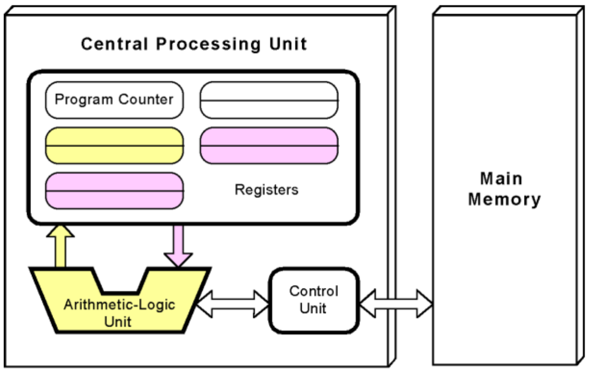
- Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.



26

The von Neumann Model

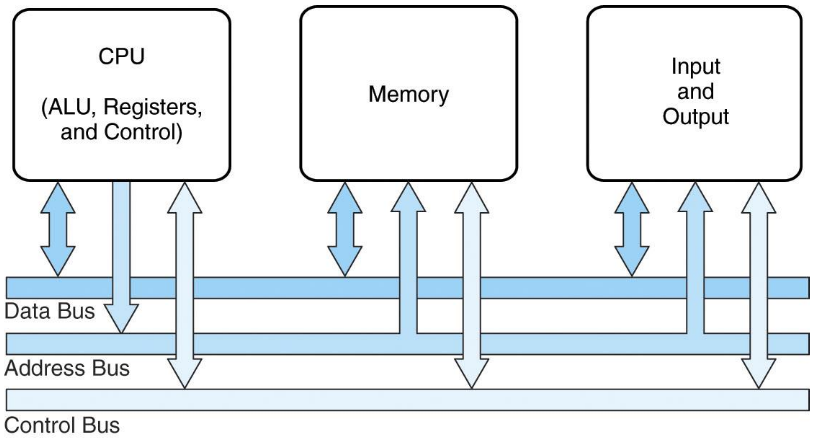
- The ALU executes the instruction and places results in registers or memory.



27

The Modified von Neumann

Adding a system bus



28

Microprocessor speed Techniques

- Pipelining
- Branch prediction
- Data flow analysis
- Speculative execution

Pineling

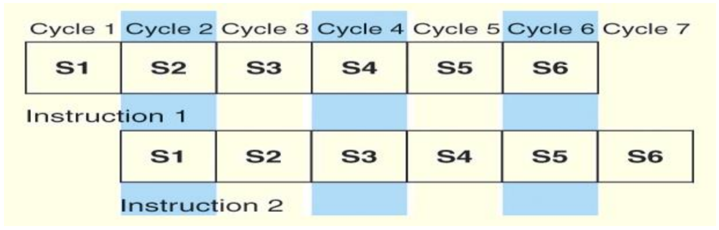
- Some CPUs divide the fetch-decode-execute cycle into smaller steps.
- These smaller steps can often be executed in parallel to increase throughput.
- Such parallel execution is called *instruction-level pipelining.*
- This term is sometimes abbreviated *ILP* in the literature.

- Suppose a fetch-decode-execute cycle were broken into the following smaller steps:

1. Fetch instruction.
2. Decode opcode.
3. Calculate effective address of operands.
4. Fetch operands.
5. Execute instruction.
6. Store result.

- Suppose we have a six-stage pipeline. S1 fetches the instruction, S2 decodes it, S3 determines the address of the operands, S4 fetches them, S5 executes the instruction, and S6 stores the result.

- For every clock cycle, one small step is carried out, and the stages are overlapped.



S1. Fetch instruction.
S2. Decode opcode.
S3. Calculate effective address of operands.
S4. Fetch operands.
S5. Execute.
S6. Store result.

Real-World Examples of pipeling

- We return briefly to the Intel and MIPS architectures from the last chapter, using some of the ideas introduced in this chapter.
- Intel introduced pipelining to their processor line with its Pentium chip.
- The first Pentium had two five-stage pipelines. Each subsequent Pentium processor had a longer pipeline than its predecessor with the Pentium IV having a 24-stage pipeline.
- The Itanium (IA-64) has only a 10-stage pipeline.

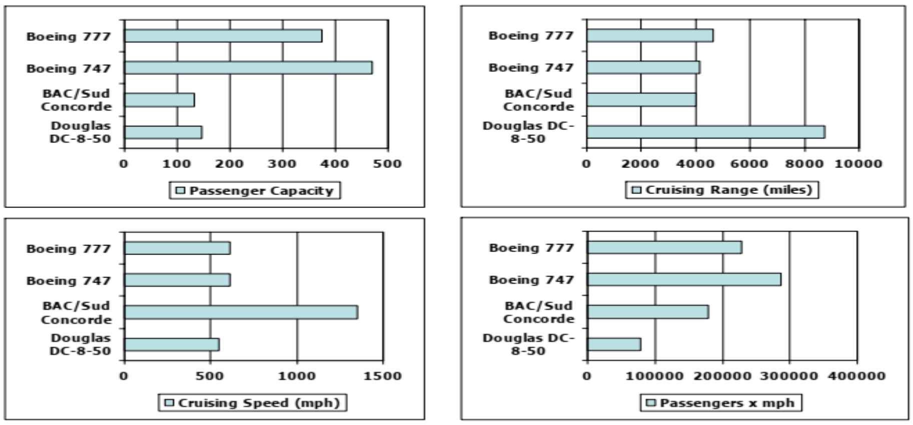
Branch Prediction

- Branch prediction is another approach to minimizing branch penalties.
- *Branch prediction* tries to avoid pipeline stalls by guessing the next instruction in the instruction stream.
  - This is called *speculative execution.*
- Branch prediction techniques vary according to the type of branching. If/then/else, loop control, and subroutine branching all have different execution profiles.
- There are various ways in which a prediction can be made:
  - *Fixed predictions* do not change over time.
  - *True predictions* result in the branch being always taken or never taken.
  - *Dynamic prediction* uses historical information about the branch and its outcomes.
  - *Static prediction* does not use any history.

- **Data flow analysis:** The processor analyzes which instructions are dependent on each other’s results, or data, to create an optimized schedule of instructions.
- **Speculative execution:** Using branch prediction and data flow analysis, some processors speculatively execute instructions ahead of their actual appearance in the program execution, holding the results in temporary locations.

Performance

- Defining Performance : Which airplane has the best performance?



Response Time and Throughput

- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time (tasks/transactions/… per hour)
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

Relative Performance

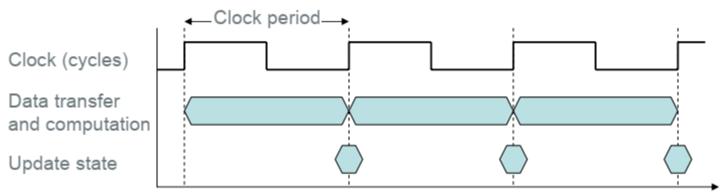
- Define Performance = 1/Execution Time
- "X is *n* time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y = \text{Execution time}_Y / \text{Execution time}_X = n$$

- Example: time taken to run a program
  10s on A, 15s on B
  Execution TimeB / Execution TimeA
  = 15s / 10s = 1.5
  So A is 1.5 times faster than B

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- Clock period: duration of a clock cycle
  - e.g., 250ps = 0.25ns = 250×10^{-12}s
- Clock frequency (rate): cycles per second
  - e.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Time

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$
$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock be?

$$\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6\text{s}}$$

$$\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A$$

$$= 10\text{s} \times 2\text{GHz} = 20 \times 10^9$$

$$\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6\text{s}} = \frac{24 \times 10^9}{6\text{s}} = 4\text{GHz}$$

Instruction Count and CPI

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
- Average CPI affected by instruction mix

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$$\begin{aligned}\text{CPU Time}_A &= \text{Instruction Count} \times \text{CPI}_A \times \text{Cycle Time}_A \\ &= I \times 2.0 \times 250\text{ps} = I \times 500\text{ps} \\ \text{CPU Time}_B &= \text{Instruction Count} \times \text{CPI}_B \times \text{Cycle Time}_B \\ &= I \times 1.2 \times 500\text{ps} = I \times 600\text{ps} \\ \frac{\text{CPU Time}_B}{\text{CPU Time}_A} &= \frac{I \times 600\text{ps}}{I \times 500\text{ps}} = 1.2\end{aligned}$$

A is faster…

…by this much

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \underbrace{\frac{\text{Instruction Count}_i}{\text{Instruction Count}}}_{\text{Relative frequency}} \right)$$

CPI Example

- Alternative compiled code sequences using instructions in classes A, B, C

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

- Sequence 1: IC = 5
  - Clock Cycles
    $= 2\times1 + 1\times2 + 2\times3$
    $= 10$
  - Avg. CPI = 10/5 = 2.0
- Sequence 2: IC = 6
  - Clock Cycles
    $= 4\times1 + 1\times2 + 1\times3$
    $= 9$
  - Avg. CPI = 9/6 = 1.5

Performance Summary

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, Tc

MIPS as a Performance Metric

- MIPS: Millions of Instructions Per Second
  - Doesn't account for
- Differences in ISAs between computers
- Differences in complexity between instructions

$$\text{MIPS} = \frac{\text{Instruction count}}{\text{Execution time} \times 10^6} = \frac{\text{Instruction count}}{\frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}} \times 10^6} = \frac{\text{Clock rate}}{\text{CPI} \times 10^6}$$

- CPI varies between programs on a given CPU

Amdahl's Law

- The overall performance of a system is a result of the interaction of all of its components.
- System performance is most effectively improved when the performance of the most heavily used components is improved.
- This idea is quantified by Amdahl's Law:

$$S = \frac{1}{(1-f) + \frac{f}{k}}$$

where S is the overall speedup;
f is the fraction of work performed by a faster component; and
k is the speedup of the faster component.

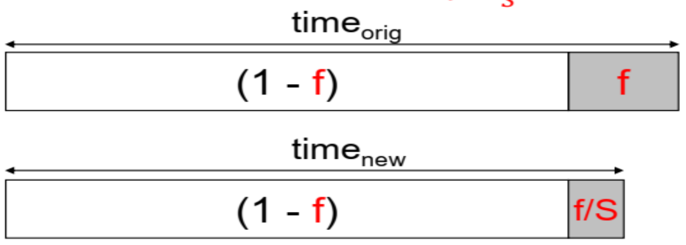
Amdahl's Law

$$\text{Speedup} = \frac{time_{without\ enhancement}}{time_{with\ enhancement}}$$

Suppose an enhancement speeds up a fraction f of a task by a factor of S

$$\text{time}_{new} = \text{time}_{orig} \cdot \left((1 - f) + \frac{f}{S} \right)$$

$$S_{overall} = \frac{1}{(1-f)+\frac{f}{S}}$$



- Amdahl's Law gives us a handy way to estimate the performance improvement we can expect when we upgrade a system component.
- On a large system, suppose we can upgrade a CPU to make it 50% faster for $10,000 or upgrade its disk drives for $7,000 to make them 250% faster.
- Processes spend 70% of their time running in the CPU and 30% of their time waiting for disk service.
- An upgrade of which component would offer the greater benefit for the lesser cost?

- The processor option offers a speedup of 1.3 times, (S = 1.3) or 30%:

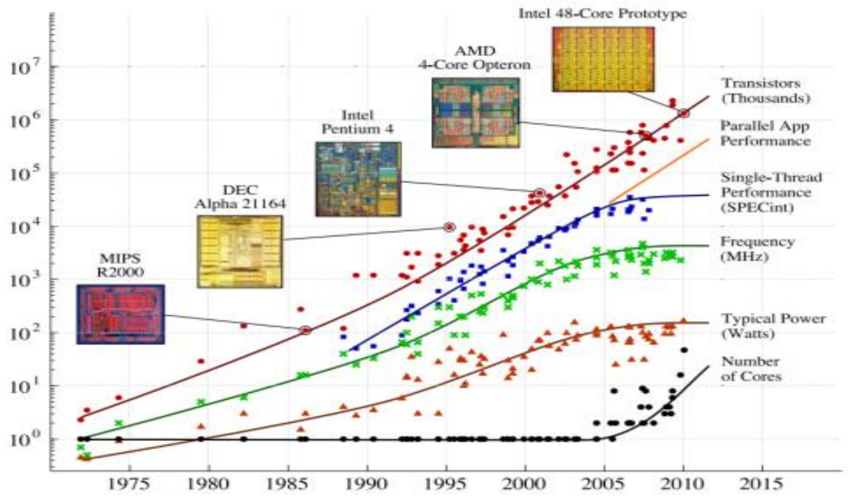
$$f = 0.70, \quad k = 1.5 \qquad S = \frac{1}{(1 - 0.7) + 0.7/1.5}$$

- And the disk drive option gives a speedup of 1.22 times (S = 1.22), or 22%:

$$f = 0.30, \quad k = 2.5 \qquad S = \frac{1}{(1 - 0.3) + 0.3/2.5}$$

- Each 1% of improvement for the processor costs $333 (10000/30), and for the disk a 1% improvement costs $318 (7000/22).

Performance and Power Trends



Evolution of Computer Technology

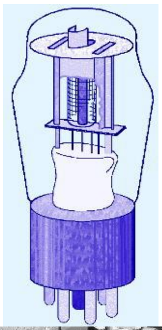
- The 0 generation
- The 1st generation
- The 2nd generation
- The 3rd generation
- Evolution of Intel process

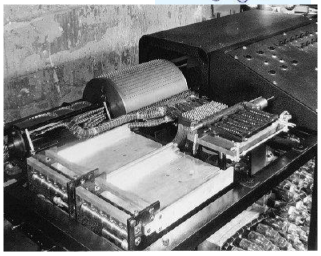
Generation Zero

- Generation Zero: Mechanical Calculating Machines (1642 - 1945)
  - Calculating Clock - Wilhelm Schickard (1592 - 1635).
  - Pascaline - Blaise Pascal (1623 - 1662).
  - Difference Engine - Charles Babbage (1791 - 1871), also designed but never built the Analytical Engine.
  - Punched card tabulating machines - Herman Hollerith (1860 - 1929).

The first generation

- The First Generation: Vacuum Tube Computers (1945 - 1953)
  - Atanasoff Berry Computer (1937 - 1938) solved systems of linear equations.
  - John Atanasoff and Clifford Berry of Iowa State University.





The First Generation

- The ENIAC (Electronic Numerical Integrator And Computer), designed and constructed at the University of Pennsylvania: The world's first general purpose electronic digital computer.
- Weighing 30 tons, occupying 1500 square feet of floor space, and containing more than 18,000 vacuum tubes. When operating, it consumed 140 kilowatts of power. It was capable of 5000 additions per second.
- The major drawback of the ENIAC was that it had to be programmed manually by setting switches, plugging and unplugging cables

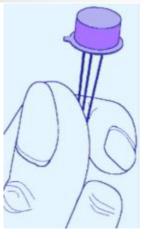
The First Generation

- In 1946, von Neumann and his colleagues began the design of a new **stored-program computer**, referred to as the IAS computer, at the Princeton **I**nstitute for **A**dvanced **S**tudies.
- Although not completed until 1952, is the prototype of all subsequent general-purpose computers

1rst Generation : Comercial

- The **UNIVAC I** (Universal Automatic Computer) (1950) was the first successful commercial computer. It was intended for both scientific and commercial applications.
- The **UNIVAC II**, which had greater memory capacity and higher performance than the UNIVAC I, was delivered in the late 1950s.
- The **IBM 701** (1953), which was delivered by IBM - the first electronic stored-program computer (punched-card processing equipment), which intended primarily for scientific applications.
- The **IBM 702** (1955) which had a number of hardware features that suited it to business applications.

The second generation



- The Second Generation: Transistorized Computers (1954 - 1965)
  - IBM 7094 (scientific) and 1401 (business)
  - Digital Equipment Corporation (DEC) PDP-1
  - Univac 1100
  - . . . and many others.

The Second Generation

- The second generation saw the introduction of **more complex arithmetic and logic units and control units**, the use of **high-level programming languages**, and the provision of ***system software*** with the computer.
- In broad terms, system software provided the ability to load programs, move data to peripherals, and libraries to perform common computations, similar to what modern OSes like Windows and Linux do.

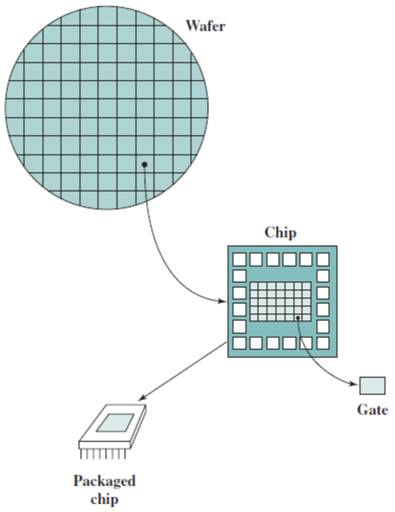
The third Generation

- The Third Generation: Integrated Circuit Computers (1965 - 1980)
  - IBM 360
  - DEC PDP-8 and PDP-11
  - Cray-1 supercomputer
  - . . . and many others.

The Third Generation

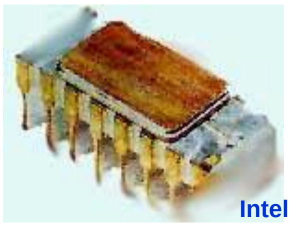
- **THE MOORE LAW**

The number of transistors that could be put on a single chip was **doubling every year** and correctly predicted that this pace **would continue into the near future**. To the surprise of many, including Moore, the pace **continued year after year** and **decade after decade**. The pace slowed to a doubling every 18 months in the 1970s but has sustained that rate ever since.



The fourth generation

- The Fourth Generation: VLSI Computers (1980 - ?)
  - Very large scale integrated circuits (VLSI) have more than 10,000 components per chip.
  - Enabled the creation of microprocessors.
  - The first was the 4-bit Intel 4004. Later versions, such as the 8080, 8086, and 8088 spawned the idea of "personal computing."


Intel 4004

The Microprocessors

- In 1971, Intel developed 4004: the first chip to contain *all* of the components of a CPU on a single chip. The 4004 can add two 4-bit numbers and can multiply only by repeated addition.
- In 1972, Intel developed 8008. This was the first 8-bit microprocessor and was almost twice as complex as the 4004.
- In 1974, Intel developed 8080 (8-bit), which was designed to be the CPU of a general-purpose microcomputer.
- By the end of 70s, general-purpose 16-bit microprocessors appeared. One of these was the 8086.

Evolution of Intel Processors

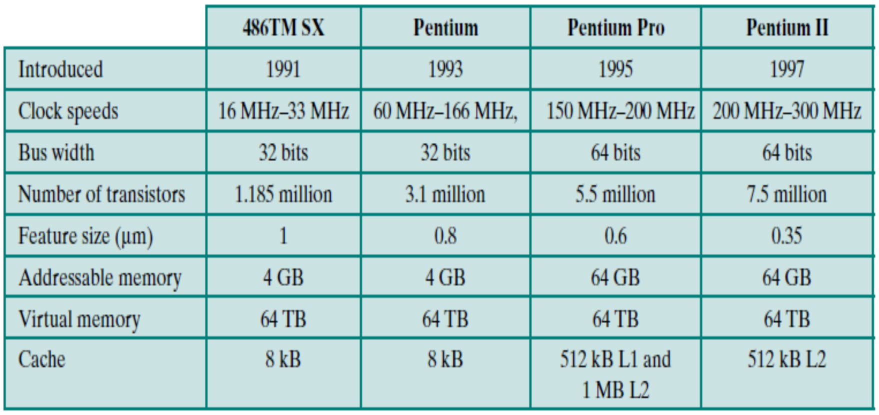
- 1970s

	4004	8008	8080	8086	8088
Introduced	1971	1972	1974	1978	1979
Clock speeds	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Bus width	4 bits	8 bits	8 bits	16 bits	8 bits
Number of transistors	2300	3500	6000	29,000	29,000
Feature size (µm)	10		6	3	6
Addressable memory	640 Bytes	16 kB	64 kB	1 MB	1 MB

- 1980s

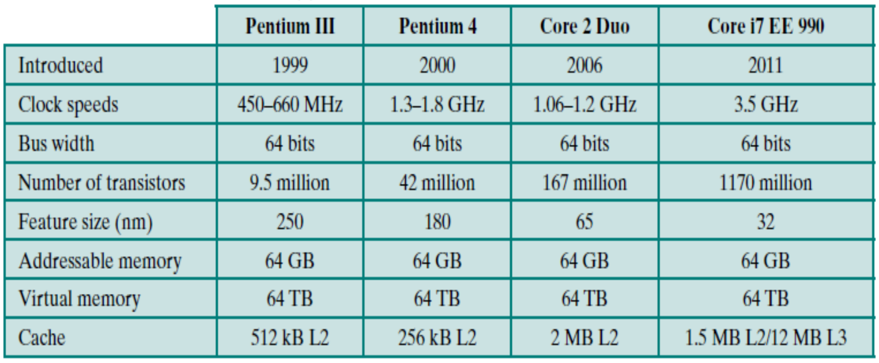
	80286	386TM DX	386TM SX	486TM DX CPU
Introduced	1982	1985	1988	1989
Clock speeds	6 MHz–12.5 MHz	16 MHz–33 MHz	16 MHz–33 MHz	25 MHz–50 MHz
Bus width	16 bits	32 bits	16 bits	32 bits
Number of transistors	134,000	275,000	275,000	1.2 million
Feature size (µm)	1.5	1	1	0.8–1
Addressable memory	16 MB	4 GB	16 MB	4 GB
Virtual memory	1 GB	64 TB	64 TB	64 TB
Cache	–	–	–	8 kB

1990s



	486TM SX	Pentium	Pentium Pro	Pentium II
Introduced	1991	1993	1995	1997
Clock speeds	16 MHz–33 MHz	60 MHz–166 MHz,	150 MHz–200 MHz	200 MHz–300 MHz
Bus width	32 bits	32 bits	64 bits	64 bits
Number of transistors	1.185 million	3.1 million	5.5 million	7.5 million
Feature size (µm)	1	0.8	0.6	0.35
Addressable memory	4 GB	4 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	8 kB	8 kB	512 kB L1 and 1 MB L2	512 kB L2

Recent processors



	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 990
Introduced	1999	2000	2006	2011
Clock speeds	450–660 MHz	1.3–1.8 GHz	1.06–1.2 GHz	3.5 GHz
Bus width	64 bits	64 bits	64 bits	64 bits
Number of transistors	9.5 million	42 million	167 million	1170 million
Feature size (nm)	250	180	65	32
Addressable memory	64 GB	64 GB	64 GB	64 GB
Virtual memory	64 TB	64 TB	64 TB	64 TB
Cache	512 kB L2	256 kB L2	2 MB L2	1.5 MB L2/12 MB L3

Multicore, MICS, GPUS

- Improvements in Chip Organization and Architecture :
- Increase the hardware speed (clock speed) of the processor:
  - Increase heat dissipation (w/cm2)
  - RC delay
  - Memory latency
- The use of multiple processors on the same chip, also referred to as multiple cores, or **multicore**, provides the potential to increase performance without increasing the clock rate.

- Chip manufacturers are now in the process of making a huge leap forward in the number of cores per chip (more than 50).
- The leap in performance as well as the challenges in developing software to exploit such a large number of cores have led to the introduction of a new term: **many integrated core (MIC)**

Evolution of Intel x86 Architecture

- Two processor families:
  - Intel x86: the sophisticated design principles once found on mainframes, supercomputers and serves (**CISC** – Complex Instruction Set Computers),
  - The ARM architecture is used in a wide variety of embedded systems and is one of the most powerful and best-designed RISC-based systems on the market (**RISC** - Reduced Instruction Set Computers),

CISC and RISC

- RISC Machines
  - The underlying philosophy of RISC machines is that a system is better able to manage program execution when the program consists of only a few different instructions that are the same length and require the same number of clock cycles to decode and execute.
  - RISC systems access memory only with explicit load and store instructions.
  - In CISC systems, many different kinds of instructions access memory, making instruction length variable and fetch-decode-execute time unpredictable.

- The difference between CISC and RISC becomes evident through the basic computer performance equation:

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{avg. cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

- RISC systems shorten execution time by reducing the clock cycles per instruction.
- CISC systems improve performance by reducing the number of instructions per program.

- The simple instruction set of RISC machines enables control units to be hardwired for maximum speed.
- The more complex-- and variable-- instruction set of CISC machines requires microcode-based control units that interpret instructions as they are fetched from memory. This translation takes time.
- With fixed-length instructions, RISC lends itself to pipelining and speculative execution.

- Consider the the program fragments:

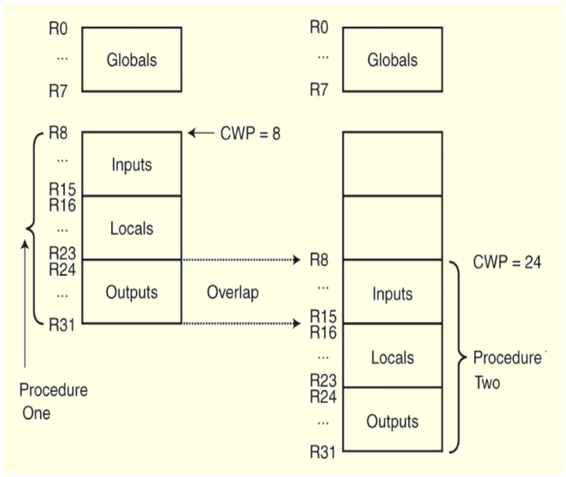
CISC
```
mov ax, 10
mov bx, 5
mul bx, ax
```

RISC
```
         mov ax, 0
         mov bx, 10
         mul cx, 5
Begin    add ax, bx
         loop Begin
```

- The total clock cycles for the CISC version might be:
  (2 movs × 1 cycle) + (1 mul × 30 cycles) = 32 cycles
- While the clock cycles for the RISC version is:
  (3 movs × 1 cycle) + (5 adds × 1 cycle) + (5 loops × 1 cycle) = 13 cycles
- With RISC clock cycle being shorter, RISC gives us much faster execution speeds.

- Because of their load-store ISAs, RISC architectures require a large number of CPU registers.
- These register provide fast access to data during sequential program execution.
- They can also be employed to reduce the overhead typically caused by passing parameters to subprograms.
- Instead of pulling parameters off of a stack, the subprogram is directed to use a subset of registers.

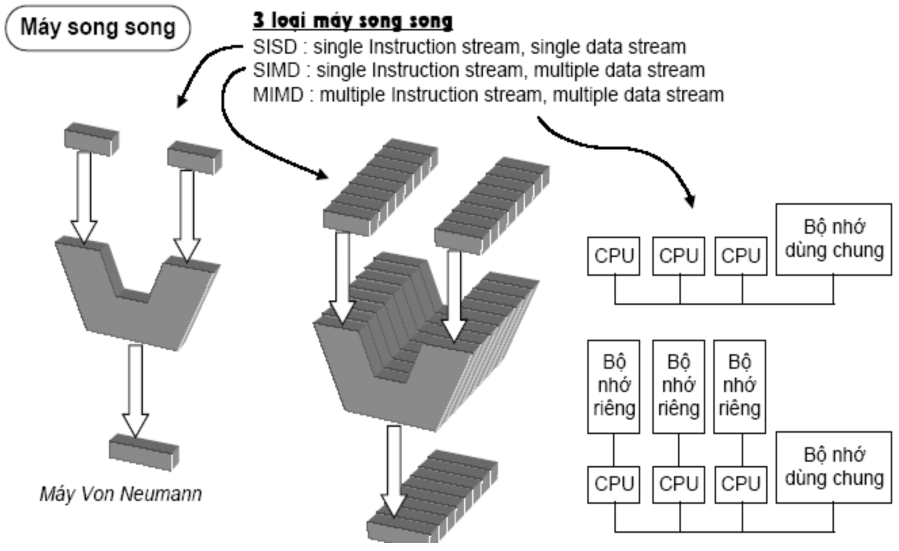
- This is how registers can be overlapped in a RISC system.
- The *current window pointer* (CWP) points to the active register window.



Flynn’s Taxonomy

- Many attempts have been made to come up with a way to categorize computer architectures.
- *Flynn’s Taxonomy* has been the most enduring of these, despite having some limitations.
- Flynn’s Taxonomy takes into consideration the number of processors and the number of data paths incorporated into an architecture.
- A machine can have one or many processors that operate on one or many data streams.

- The four combinations of multiple processors and multiple data paths are described by Flynn as:
  - **SISD:** Single instruction stream, single data stream. These are classic uniprocessor systems.
  - **SIMD:** Single instruction stream, multiple data streams. Execute the same instruction on multiple data values, as in vector processors.
  - **MIMD:** Multiple instruction streams, multiple data streams. These are today's parallel architectures.
  - **MISD:** Multiple instruction streams, single data stream.


Máy song song
3 loại máy song song
SISD : single Instruction stream, single data stream
SIMD : single Instruction stream, multiple data stream
MIMD : multiple Instruction stream, multiple data stream
Máy Von Neumann
CPU
CPU
CPU
Bộ nhớ dùng chung
Bộ nhớ riêng
Bộ nhớ riêng
Bộ nhớ riêng
CPU
CPU
CPU
Bộ nhớ dùng chung