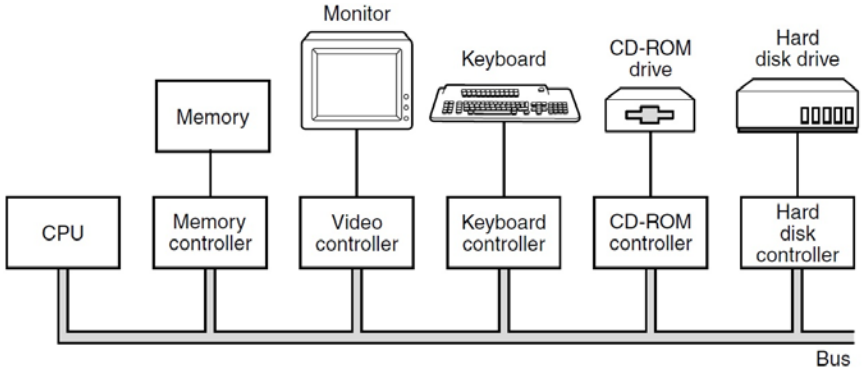
# Processor and System Architecture



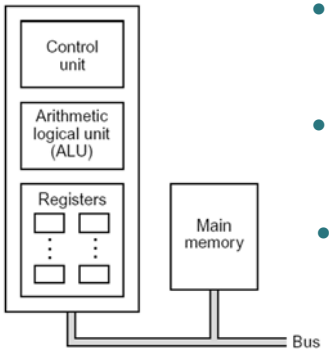
## Logical structure of a computer system




- Controller: controls its device and handles bus access



## Processor structure




- *Control unit*
  - Fetches instructions from main memory, determines their types
- *ALU*
  - Executes the instruction fetched
- *Registers*
  - Store temporary results and control information
  - Can be read/written at high speed
  - General/special purpose

## Processor specifications

- Processors can be identified by two main parameters
  - Processor speed
  - Processor width



Processor speed

- Counted in megahertz (MHz) and gigahertz (GHz)

Processor width

- Expressed with three main specifications
  - ✓ Data I/O bus
  - ✓ Address bus
  - ✓ Internal registers

### Data I/O bus

- Other names: FSB, PSB or just CPU bus
- Bus between processor and memory controller
- Each wire carries a single bit
- Width defines rate at which data can be moved into or out of processor
- Shared, bi-directional (up to 2004)

## Data I/O bus

- Improvements:
  - Dual independent buses [2 processors share 1 bus
  - Quad buses
  - Dedicated FSB for each core
  - QPI (Quick Path Interconnect) [2 unidirectional links for each pair of components]
- Intel 8088 (1979): 8 bits wide data bus
- Intel Pentium 4 (2000): 64 bits
- AMD Athlon (2003): 64 bits
- AMD Phenom II (2009): 16 bits
- Current processors: 64 bits wide data bus



## Address bus

- Carries memory addresses (to send / retrieve data)
- Each wire carries a single bit (single digit in memory address)
- Width indicates maximum amount of RAM the processor can handle

| Processor Family | Address Bus | Address Space |
|---|---|---|
| 8088, 8086 | 20-bit | 1MB |
| 286, 386SX | 24-bit | 16MB |
| 486, Pentium, Athlon XP | 32-bit | 4GB |
| Celeron, Pentium II, Pentium IV | 36-bit | 64GB |
| Athlon 64 FX, Opteron, Core 2 Duo/Quad | 40-bit | 1TB |
| Itanium, Itanium II | 44-bit | 16TB |



## Internal registers (Internal data bus)

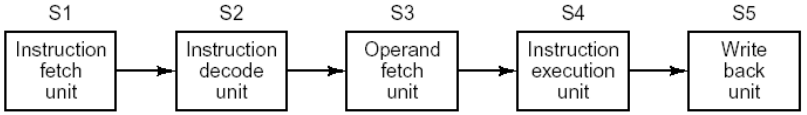
- Size indicates how much information the processor can operate at a time
- Also describes type of software or instructions the chip can run
  - Ex. processor with 32-bit internal registers can run 32-bit operating systems and software
- *n* bits wide internal registers ⇒ *n*-bit processor
  - Pentium 4: 32-bit, Core 2 and newer: 64-bit

## Instruction pipelining

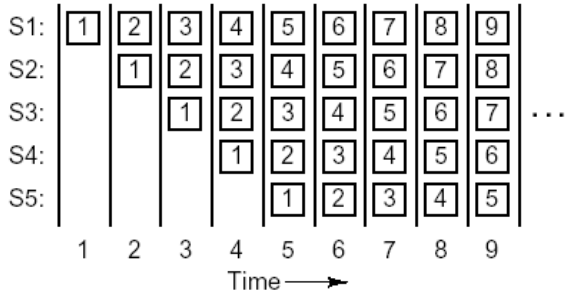
- Divides instruction execution into many parts; each handled by different hardware and can run in parallel



## Computer Pipelines




- S1 – Fetch instruction from memory and place it in a buffer until it is needed
- S2 – Decode the instruction; determine its type and operands it needs
- S3 – Locate and fetch operands from memory (or registers)
- S4 – Execute instruction
- S5 – Write back result in a register



Let

*T* – cycle time (ns)

*n* – no. of stages in the pipeline

***Latency***: time taken to execute an instruction = *nT* ns

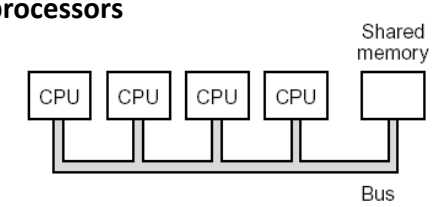
***Processor Bandwidth***: no. of MIPS the CPU has = $\frac{1000}{T}$ MIPS

| Processor | Pipeline Depth |
|---|---|
| Pentium | 5-stage |
| Pentium III | 10-stage |
| Athlon/XP | 10-stage |
| Athlon 64 | 12-stage |
| P4 | 20-stage |
| P4 Prescott | 31-stage |
| Pentium D | 31-stage |
| Pentium M/Core | 10-stage |
| AMD Phenom | 12-stage |
| Core 2/i5/i7 | 14-stage |

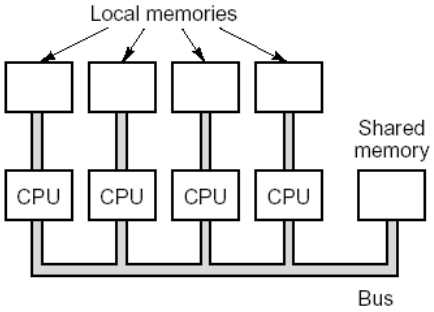


**Processor-level Parallelism**

**Multiprocessors**




High bus traffic




Low bus traffic



# Measuring Performance



**Performance metrics**

- **Execution time (response time):**
  - time between start and completion of a task (including disk accesses, memory accesses, etc.)
- **Throughput (bandwidth):**
  - total amount of work done in a unit time

- **CPU execution time (CPU time):**
  - Actual time CPU spends computing for a task (does not include any overhead)

$$Performance_X = \frac{1}{Execution\ time_X}$$

$$\frac{Performance_X}{Performance_Y} = n \quad \Rightarrow \quad X \text{ is } n \text{ times faster than } Y$$

- Computer clock determines when events take place in the hardware
- **Clock cycle time**
  - length of one complete clock period
- **Clock rate**
  - inverse of clock cycle time

$$\text{Clock rate} = \frac{1}{\text{Clock cycle time}}$$

- **CPI** (clock cycles per instruction)
  - average no. of CPU clock cycles each instruction takes to execute
- **IC** (instruction count)
  - no. of instructions executed in a program

CPU clock cycles for a program = IC x CPI

**Remark:**

CPI can be used to compare two different implementations of the same instruction set architecture since the instruction count is the same.



**CPU performance equation**

CPU time = CPU clock cycles for a program x Clock cycle time

= IC x CPI x Clock cycle time

$$= \frac{\text{IC x CPI}}{\text{Clock rate}}$$

**Example:**

A program runs in 10s on computer A having 1.5GHz clock. A new computer B, which could run the same program in 6s, has to be designed. Further, B should have 1.2 times as many clock cycles as A.

What should be the clock rate of B?

**Answer:**

$$\text{CPU clock cycles}_A = \text{CPU time}_A \text{ x Clock rate}_A$$
$$= 10 \text{ x } 1.5 \text{ x } 10^9 \text{ cycles}$$
$$= 15 \text{ x } 10^9 \text{ cycles}$$

$$\text{Clock rate}_B = \frac{\text{CPU clock cycles}_B}{\text{CPU time}_B} = \frac{1.2 \text{ x } 15 \text{ x } 10^9}{6} = 3.0 \text{ GHz}$$



**Example:**

Consider two implementations of the same instruction set architecture. For a certain program, details of time measurements of two machines are given below.

| Machine | Clock cycle time | CPI |
|---|---|---|
| *A* | 1 ns | 2.0 |
| *B* | 2 ns | 1.2 |

Which machine is faster for this program and by how much?

**Answer:**

CPU time = IC x CPI x Clock cycle time

$\text{CPU time}_A$ = IC x 2.0 x 1 = 2 x IC ns

$\text{CPU time}_B$ = IC x 1.2 x 2 = 2.4 x IC ns

$$\frac{\text{CPU time}_B}{\text{CPU time}_A} = \frac{2.4 \text{ x IC}}{2 \text{ x IC}} = 1.2$$

$\Rightarrow$ A is 1.2 times faster than B



**How to measure components of CPU performance equation?**

- Clock rate
  - Published by the manufacturer
- IC
  - Simulator of the architecture
  - Software tools that profile the execution
  - Hardware counters included in the processors
- CPI
  - Hardware counters included in the processors

- Processors execute different types of instruction
- Suppose n different types of instruction

Let

$IC_i$ – No. of times instruction type i is executed in a program

$CPI_i$ – No. of clock cycles required for instruction i

$\Rightarrow$ CPU clock cycles = $\sum_{i=1}^{n} CPI_i \times IC_i$

Overall CPI:

$$\frac{\sum_{i=1}^{n} IC_i \times CPI_i}{IC} = \sum_{i=1}^{n} CPI_i \times \left(\frac{IC_i}{IC}\right)$$



**Example:**

Suppose we have made the following measurements:

- Frequency of FP operations (other than FPSQR) is 25%
- Average CPI of FP operations is 4.0
- Average CPI of other instructions is 1.33
- Frequency of FPSQR is 2%
- CPI of FPSQR is 20

There are two design alternatives:

1. decrease CPI of FPSQR to 2
2. decrease average CPI of all FP operations to 2.5

Compare these two design alternatives using CPU performance equation.

**Answer:**

Note that only CPI changes; clock rate, IC remain identical

CPI with neither enhancement:

$$CPI_{Original} = \sum_{i=1}^{n} CPI_i \times \left(\frac{IC_i}{IC}\right)$$

$= (4 \times 25\%) + (1.33 \times 75\%) = 2.0$



CPI for enhanced FPSQR:

$CPI_{New\ FPSQR} = CPI_{Original} - 2\%\ (CPI_{Old} - 2) = 2.0 - 2\% \times (20 - 2) = 1.64$

CPI for enhanced FP:

$CPI_{New\ FP} = (75\% \times 1.33) + (25\% \times 2.5) = 1.625$

**Amdahl's Law**

Performance improvement that can be gained from some faster mode of execution is limited by fraction of time the faster mode can be used

Speedup depends on

- fraction of computation time in original machine that can be converted to take advantage of the enhancement ($Fraction_{Enhanced}$)
- improvement gained by enhanced execution mode ($Speedup_{Enhanced}$)

**Example:**

Total execution time of a program = 50s

Execution time that can be enhanced = 30s

$\Rightarrow Fraction_{Enhanced} = 30/50$



**Example:**

Normal mode execution time for some portion of a program = 30s

Enhanced mode execution time for the same portion = 10s

$\Rightarrow Speedup_{Enhanced} = 30/10 = 3$

$$\text{Execution time}_{New} = \text{Execution time}_{Old} \times \left[\left(1 - \text{Fraction}_{Enhanced}\right) + \frac{\text{Fraction}_{Enhanced}}{\text{Speedup}_{Enhanced}}\right]$$

$$Speedup_{Overall} = \frac{\text{Execution time}_{Old}}{\text{Execution time}_{New}} = \frac{1}{\left(1 - \text{Frantion}_{Enhanced}\right) + \frac{\text{Fraction}_{Enhanced}}{\text{Speedup}_{Enhanced}}}$$

**Example:**

Suppose we consider an enhancement to the processor of a server system used for Web serving. New CPU is 10 times faster on computation in Web application than original CPU. Assume original CPU is busy with computation 40% of the time and is waiting for I/O 60% of time.

What is the overall speedup gained from enhancement?

$Fraction_{Enhanced} = 0.4$

$Speedup_{Enhanced} = 10$

$\Rightarrow Speedup_{Overall} = \dfrac{1}{(1 - 0.4) + \dfrac{0.4}{10}} = 1.56$

**Remark:**

If an enhancement is only usable for a fraction of a task, we cannot speedup by more than $\dfrac{1}{(1 - Fraction_{Enhanced})}$

**Example:**

A common transformation required in graphics engines is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics.

Suppose FP square root (FPSQR) is responsible for 20% of execution time of a critical graphics program.



Design alternatives

1. Enhance FPSQR hardware and speed up this operation by a factor of 10
2. Make all FP instructions run faster by a factor of 1.6

FP instructions are responsible for a total of 50% of execution time. Design team believes they can make all FP instructions run 1.6 times faster with same effort as required for fast square root.

Compare these two design alternatives.

**Answer:**

$Speedup_{FPSQR} = \dfrac{1}{(1 - 0.2) + \dfrac{0.2}{10}} = 1.22$

$Speedup_{FP} = \dfrac{1}{(1 - 0.5) + \dfrac{0.5}{1.6}} = 1.23$



**Performance of a program depends on**

- Algorithm (IC, CPI affected)
- Programming language (IC, CPI)
- Compiler (IC, CPI)
- Instruction Set Architecture (IC, CPI, Clock rate)