Computer Abstractions and Technology

→ Computer architecture vs Computer Organization

Computer architecture → attributes of the computer that are visible to the programmer which have a direct impact on the logical execution of a program. (e.g. instruction set, processor registers, format of different data types, memory addressing modes, I/O mechanisms)

Computer organization → role of internal operational units and ways they interconnect to implement the architectural specification (eg. hardware details transparent to programmer, interfaces between processor and memory, computer and I/O devices, memory technology being used.

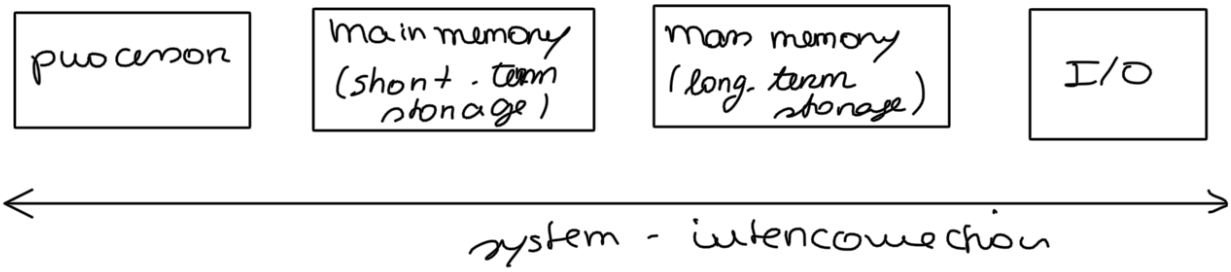
For microcomputers, the relationship between architecture and organization is very close.

→ Structure and function

Structure: how many components there are and the way in which they are interconnected.

function: the operation of each component as part of the whole. Basic functions: inputting data, outputting data, processing data, storing data and control.

↳ Top level structure



processor → controls the operation and perform its data processing functions.

main memory → stores data during processing (volatile)

mass memory → stores data in-between processing and large data to be retrieved and updated. Non-volatile (special I/O device).

I/O → moves data between computer and external environment

system interconnection → data transfer among the components (bus)

processor

Instruction set: instructions of the following

type → data movement (between registers and main memory / I/O controller), arithmetic / logic, branching (modifying sequentiality), subroutine calling

→ Classes of computer systems:

- personal mobile devices: energy efficiency, < cost, weight requirements, apps are most web-based and media-oriented.
- desktop computing: low-end laptops to high-end workstations, price-performance optimization.
- servers: backbone of large-scale enterprise computing availability and scalability are important, efficiency.
- clusters / warehouse-scale computers: collection of desktop computers on servers, connected by lan to act as a single computer. price-performance is critical, also availability
- embeded computers: found in every day machines, meeting performance at minimum price.

→ Classes of parallelism

Two types of parallelism in applications

- data-level parallelism: multiple data items that can be processed at same time
- task-level parallelism: task is subdivided into subtasks that operate independently

These two can be exploited into four ways:

- instruction-level parallelism: pipelining
- through special hardware (vector-architectures and GPUs): GPUs use same instruction to a collection of data in

parallel.

- thread-level parallelism: concurrent threads.
- request-level parallelism: specified by programmer.

Computers are in four categories:

SISD - single instruction and single data stream: uniprocessor.

SIMD - " " multiple data streams: same instruction executed by multiple processing units in different data streams.

MISD - multiple instruction and single data stream: multiple instructions one same data piece.

MIMD - " " multiple data: each processor runs its own data.

→ Amdahl's law

$$\text{speedup} = \frac{1}{\left(1 - \text{frac}_{\text{enhanc}}\right) + \frac{\text{frac enhanc}}{\text{speedup enhanc}}}$$

$\text{frac}_{\text{enhanc}}$ → time fraction in the original computer system which can be converted to take advantage of the faster execution.

$\text{speedup}_{\text{enhanc}}$ → speed up to be gained.

→ Quantitive principes of computer design

Take advantage of parallelism.

Take advantage of the principle of locality (reuse instructions and they have been recently used)

Focus on the common case.

focus on the common

→ Power and energy in integrated circuits

Techniques to improve energy efficiency:

- keep track on operations (turn off clock of inactive modules)
- dynamic voltage - frequency scaling
- design for typical case
- over clocking

→ Dependability

MTTF = mean time to failure

FIT = failures in time

MTTR = mean time to repair (service interruptions)

MTBF = mean time between failures = MTTF + MTTR

module availability is a measure of continuous mode of operation:

$$\text{module availability} = \frac{MTTF}{MTTF + MTTR}$$

→ Reasoning Performance

Benchmarks running programs that are simpler:

- **kernels**: small key portions of real apps
- **toy programs**: small programs
- **synthetic code**: fake programs to match behaviour of real apps.

$$\frac{\text{Geometric mean } A}{\text{Geometric mean } B} = \frac{\sqrt[n]{\prod_{i=1}^{n} SPEC_{ratio\,A}}}{\sqrt[n]{\prod_{i=1}^{n} SPEC_{ratio\,B}}} =$$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{\text{SPEC ratio A}}{\text{SPEC ratio B}}} = \sqrt{\prod_{i=1}^{n} \frac{\text{exec time B}}{\text{exec time A}}} = \sqrt{\prod_{i=1}^{n} \frac{\text{Performance A}}{\text{Performance B}}}$$

$$\text{CPU execution time} = \text{CPU clock cycles} \cdot \text{clock cycle time}$$

$$= \text{instruction count} \cdot \text{CPI} \cdot \text{clock cycle time}$$

dependent on pc architecture and compiler technology

average num. of clock cycles per instruction

dependent on pc architecture and organization

dependent on hw technology and pc organization

$$\text{CPU execution time} = \text{instruction count} \cdot \sum_{i} \left(\frac{\text{instruction count}_i}{\text{instruction count}} \times \text{CPI} \right) \cdot \text{clock cycle time}$$