Part 9A - Design Principles

1. von Neumann Architecture

Modern computers are based on the design of John von Neumann, who simplified the construction and use of computers.

There are three main attributes to this architecture:

1. Programs are stored and executed in memory
2. Separation of processing from memory
3. Different system components communicate oven a shared bus

A. The Bus

The bus is an electronic pathway that transports data between components (kinda like a highway, data moves on shared paths). There are three categories/sets of signals on the memory bus:

1. address bus
2. data bus
3. control bus

Address Bus

This bus is used by the processor to access a specific piece of data. This "address can be":

- a specific byte in memory
- a unique I0 port
- etc The more bits it has, the more memory can be accessed.

- 8-bit → 2^8 = 256 bytes
- 16-bit → 2^{16} = 64 KB (65,536 bytes)
- 32-bit → 2^{32} = 4 GB (4,294,967,296 bytes)
- 64-bit → 2^{64} = 18 EB (18,446,744,073,709,551,616)

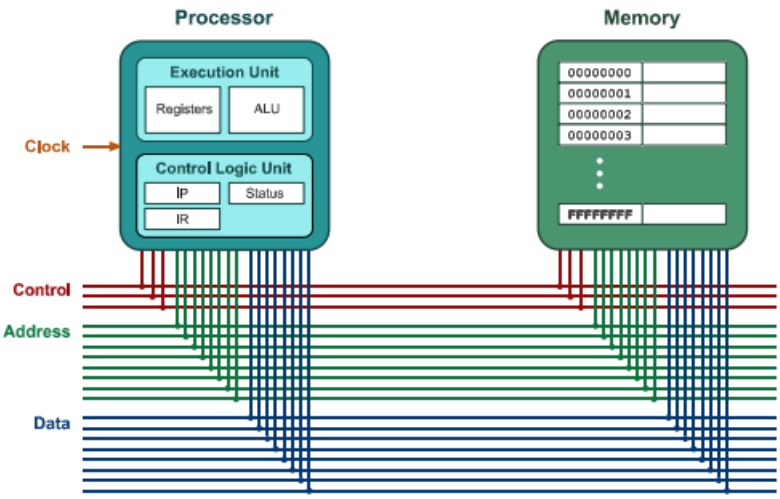
Data Bus

The actual data travels over the data bus. The number of bits that the processor uses (as its natural unit of data) is called a ***word***. A system is typically defined by its word size. For example:

- 8-bit system uses 8-bit words
- 16-bit system uses 16 bits (2 bytes) words
- 32-bit system uses 32 bits (4 bytes) words

Control Bus

The control bus controls the timing and synchronizes the subsystems. It Specifies what is happening, e.g., read data, write data, reset, etc, etc.



B. von Neumann Architecture Today

Most modern systems use a modified version of his design. In particular, they have a special high-speed bus between the processor/memory (like an HOV lane or high-speed rail; fixed source and destination and goes faster than the freeway).

2. The System Clock

The rate at which instructions are executed are controlled by the CPU clock. The faster the clock rate, the faster instructions will be executed. This is measured in ***Herts (Hz)***, the number of oscillations per second. Computers are typically labeled on the processor's clock rate.

Not all instructions are equal in execution time: some will require many cycles to execute.

3. Technological Trends

Since the design of the integrated circuit (IC), computers have advanced dramatically. Integrated circuits have improved in:

- ***Density***: the total number of transistors and wires placed in a fixed area on a silicon chip.
- ***Speed***: How quickly basic logic gates and memory devices operate.
- ***Area***: the physical size of the largest integrated circuit that can be fabricated.

The rate of improvement, which is exponential rather than linear, has been described by ***Moore's Law*** for the past 50 years: performance doubles every 18 months.

4. Computer Architectures

Computer architecture is often a contentious debate, which many issues, such as:

- how memory should be accessed
- what instructions are needed

- how are instructions to be encoded.

Typically, this debate is dominated by CISC vs. RISC, although very rarely is any one processor purely one or the other.

It's all a balance of hardware vs. software.

I. RISC

Reduced **I**nstruction **S**et **C**omputer (RISC). This emphasizes simplicity in hardware and more complexity in software. RISC will contain fewer instructions (the bare minimum needed to work) with limited memory access (often just register load and store). Thus, this architecture will also have many more registers.

Instructions will tend to take only one cylce each and the number of bytes used by instructions tend to be fixed in size.

Advantages:

- simpler instructions simplify hardware (easy to manufacture)
- less to learn and master
- less heat produced and less energy required
- memory access is minimized

II. CISC

Complex **I**nstruction **S**et **C**omputer (CISC) emphasizes flexability in instructions; hardware bears the burnt of complexity. Instructions can take multiple clocks and operands are *generalized* - each can access memory, immediates, and registers.

This architecture features fewer general purpose registers and instruction bit sizes vary.

This requires fewer instructions than RISC for the same computation and software is easier to write within the flexibility.

Advantages:

- programs written tend to take less space in memory
- varied size instructions can make it possible for the processor to evolve: add new instructions

CISC	RISC
Emphasis on hardware complexity	Emphasis on software complexity
Most operands can access memory	Load/Store instructions can access memory
Low number of registers	Higher number of registers
Instructions can have multiple clock cycles	Instructions tend towards one per clock cycle
Encoded instructions vary in size	Encoded instructions are all the same size

5. Instruction Operands

The number of instruction operands varies based on processor. More operands means more functionality, but come at the cost of having to store more bits in memory. Processors typically support either 1, 2, or 3 operands.

I. Single

Single operand processors are known as ***accumulators***. These use the accumulator register for all mathematical operations and other registers are simply used for comparison and to store temp data.

II. Double

Two operand processors allow two operands to be specified. Both are typically treated as input and one is used to store the result.

III. Make it Triple

Three operand processors can specify a third output operand to be used as an index for simple addressing or for other uses.

6. Instruction Execution

Instruction execution can be broken down into 5 steps:

1. ***Fetch***: The processor fetches the instruction from memory and stores it in the IR.
2. ***Decode***: The instruction is decoded to determine what it is and its operands. Signals are sent to the EU as input.
3. ***Read***: EU reads the values of the instruction.
4. ***Execute***: Values are passed to the ALU to act upon.
5. ***Write***: The result is written into register/memory. Processor also updates the flags and other state information such as the IP.

I. Pipelining

Doing two instructions consecutively is cumbersome and inefficient. We can use pipelining to execute instructions simultaneously. This is implemented solely in hardware and is *different* from multi-core processors: it only happens on a single core.

On modern processors, practically all the hardware is in continuous use via pipelining.

In pipelining, the speedup is directly proportional to the number of steps that we can do in parallel. As the number of loads increase, the start-up and wind-down costs (where the pipeline isn't completely full) lower.