Computer Architecture:
(As defined by the chief architects of the IBM system 360)

The structure of a computer that a machine language programmer must understand to write a correct program for that machine. This interpretation comprises of the definition of registers and memory as well as of the instruction set, instruction formats, addressing modes and the actual coding of the instructions excluding implementation and realization.

Implementation -> actual hardware
Realization -> logic technology

Computer Architecture: Modern interpretation:

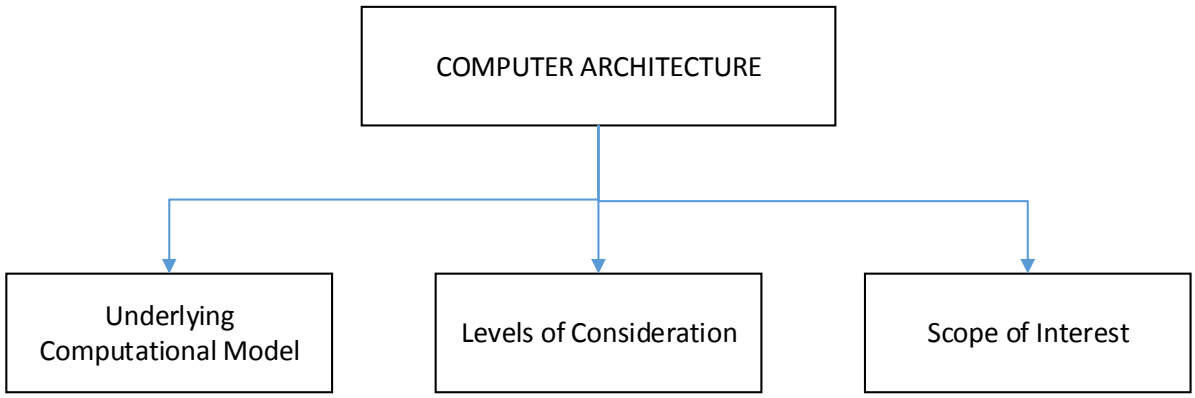
The concept of computer architecture maybe interpreted at a number of levels of increasing abstraction:

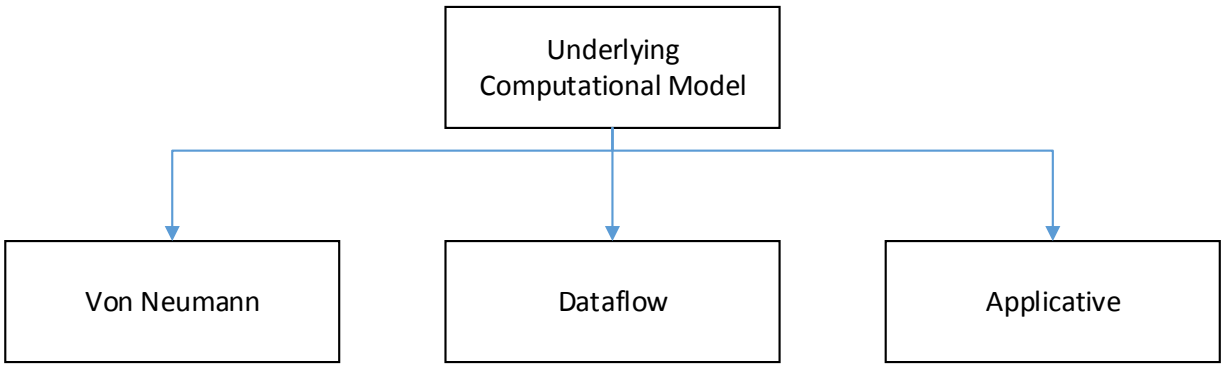
1. The micro machine level (in microprogrammed processors)
2. The process level
3. The computer-system level

Level of consideration: (in increasing order of abstraction):

1. Micro machine
2. Processor
3. Computer System

Aspects of interpreting the aspects of computer architecture:





Scope of interest at each level of consideration:

If we are interested in the functional specification of a computer, we are dealing with its abstract architecture

If we are interested in its implementation, we are concerned with its concrete architecture

Abstract architecture: The abstract architecture is a black-box specification either from the programmers point of view(programming model) or from the hardware designers point of view(hardware model).

Concrete architecture: It specifies how a computer is implemented at a particular level of abstraction, whereby a particular computational model is taken for granted.

By implementation we must mean the internal structure and operation which may be given by specifying a set of components and then interconnections as well as the data- and control-flows

The concrete architecture can also be considered from 2 different points of view. Logic or physical design

The **logic design** is an abstraction of the physical design. Its specification requires:

1. The declaration of the logical components used(actually the declaration of their black-box behavior, that is, of their abstract architecture), such as registers, execution units and so on.
2. The specification of their interconnections
3. The specification of the sequence of information transfers, which are initiated by each of the declared functions(operator, microinstructions or instructions)

The **physical design** is based on concrete circuit elements. The specification of a physical design includes

1. The declaration of the circuit elements used which also include the specification of signals
2. The specification of their interconnections
3. The declaration of initiated signal sequences

The concept of a computational model:

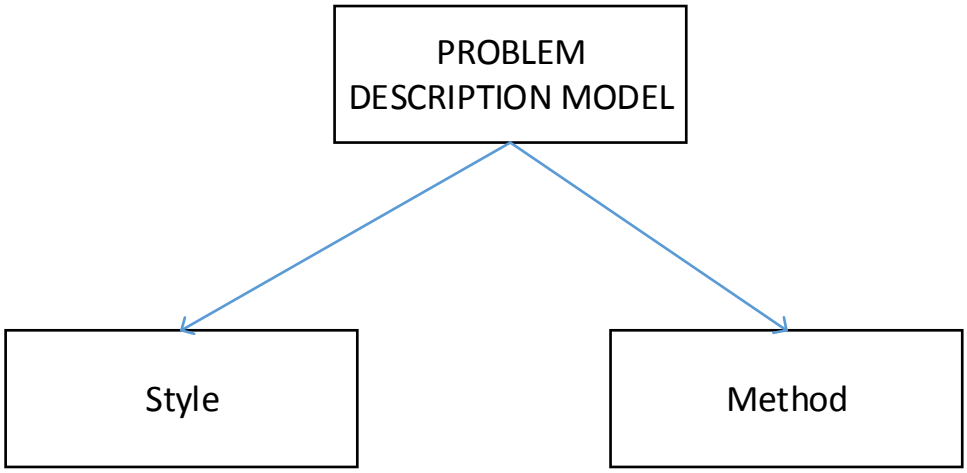
Three abstractions:

1. The basic items of computation
2. The problem description model
3. The execution model

1. The basic items of computation:

Von-Neumann: Data (represented by named entities)
Object based model: Objects or messages sent to them
Applicative model: Arguments and the functions applied to them

2. The problem description model:



Style:

Procedural Style: algorithm is stated
Declarative style: all the facts and relations relevant to the given problem have to be stated
1st method: Functions
2nd method: Predicates

Method:

Procedural Style: A solution has to be described
Declarative Style: Problem itself has to be described

Execution Model:

It has 3 components

1. Interpretation of the computations
2. Execution semantics
3. Control of the execution segments

1. Interpretation:

Von-Neumann Method: Computation has to be interpreted as the execution of the given sequence of the instructions

2. Execution semantics:

It is a rule that describes how a single execution step is to be performed

a. state transition semantics: Von Neumann and object-based models
b. Datagram semantics: data flow model
c. Reduction semantics: applicative model
d. SLD resolution: predicate logic based models

3. Control of the execution sequence

a. Control driven execution: a program exists (i.e. Von Neumann and object based models)
b. Data driven execution: an operation is activated as soon as all the input data is available(data flow model)
c. Demand driven execution: the operation will be activated only when the execution is needed to achieve the final result(applicative model)

Von Neumann Computational Model:

1. Basic items of computation: data assigned to named entities (variables)
2. Problem description model: Procedural (computational task is specified as a sequence of instructions)
3. Execution semantics: State transition semantics
4. Interpretation of how to perform the computation: the given sequence of instructions will be executed using the input data
5. Control of the execution sequence: control driven

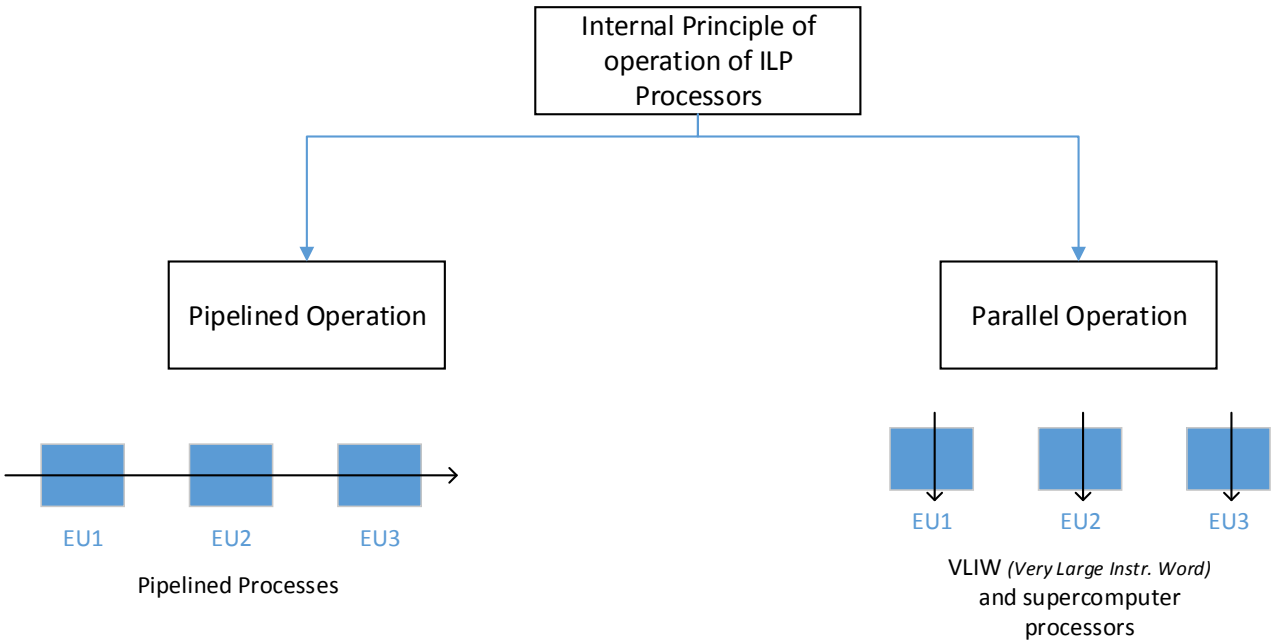
Named data entities:

The peculiarity of named data entities is that they behave like data containers that is during computation they can be assigned to new values as many times as required. They can keep the value they have until a new value is assigned.

Evolution of Von Neumann processes:

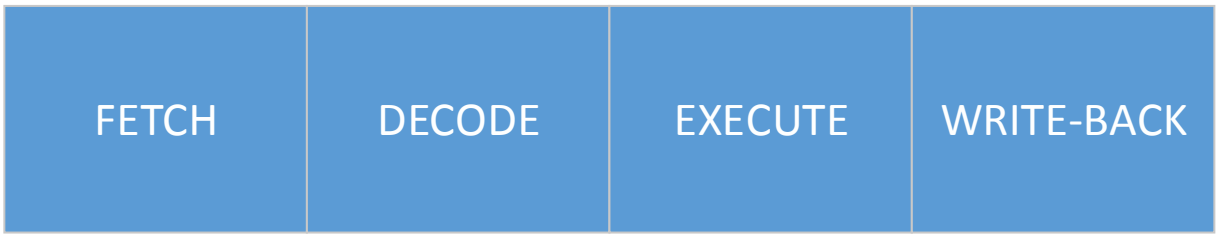
Traditional Von Neumann Processors *(sequential issue, sequential execution)*	**Scalar ILP Processors** *(sequential issue, parallel execution)*	**Super-scalar ILP Processors** *(parallel issue, parallel execution)*

Parallelism of instruction execution
Parallelism of instruction issue



Pipelined processors work like an assembly line, VLIW and super-scalar processors operate basically in parallel making use of number of concurrently working E.Us.

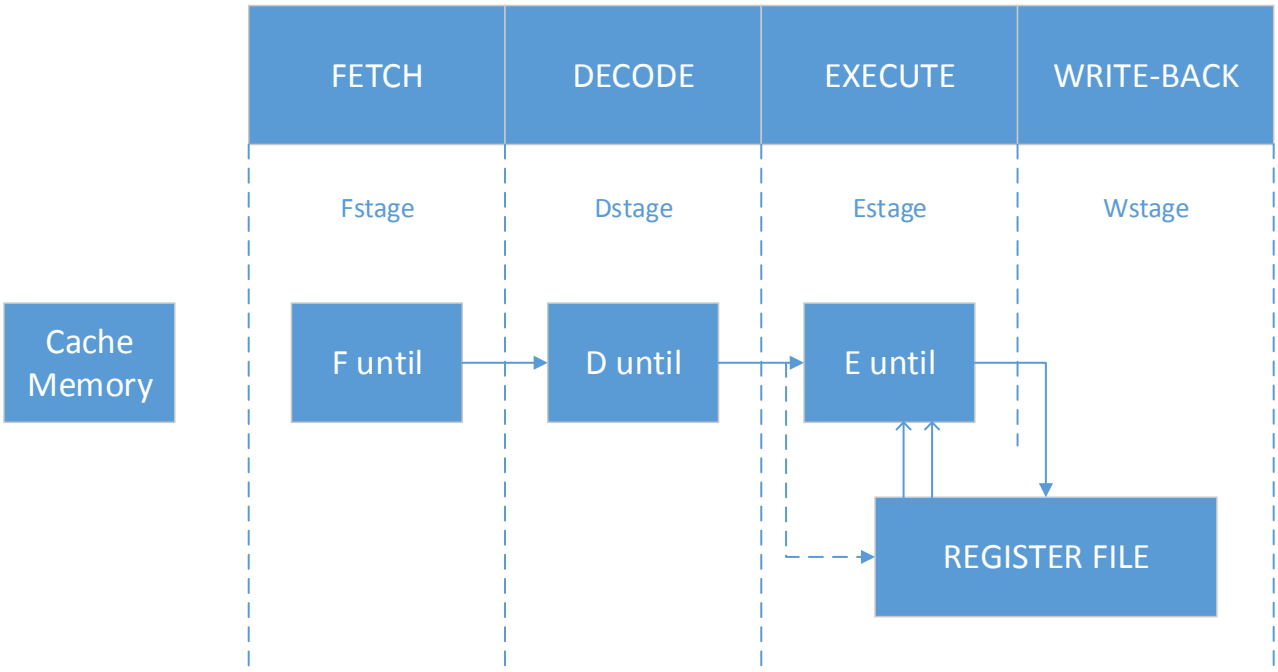
Principle of operation of pipelined processors



(SUBTASKS OF INSTRUCTION EXECUTION FOR INTEGER REGISTER REGULAR OPERATION)

Subtasks of instruction execution:

- Instruction processing is subdivided into a number of successive subtasks.
- Each subtask is performed by an associative pipeline stage.
- All the pipeline stages operate like an assembly line, with synchronous timing.



(BASIC STRUCTURE OF AN INTEGER PIPELINE)