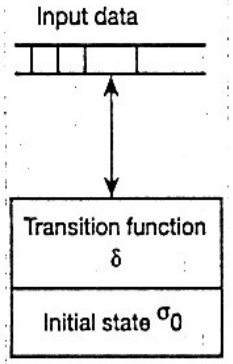
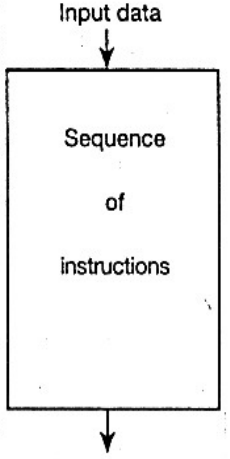
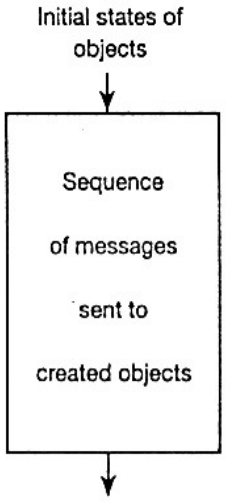
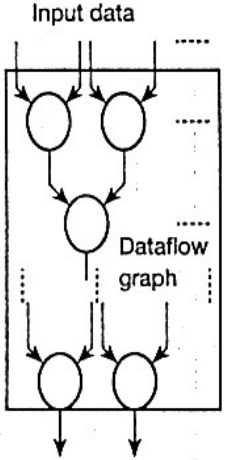
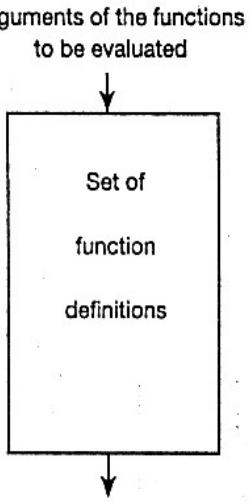
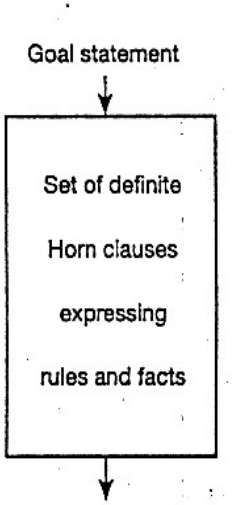
High Performance Architectures

Basic Concepts and Taxonomy

Basic computational models

- Turing
- von Neumann (and many extensions)
- Dataflow
- ...

Key features of basic computational models

Computational model	Turing	von Neumann	Object-based	Dataflow	Applicative	Predicate logic-based
Problem description model						
Basic items of computation	Elements of the tape symbol set Mappings defined on them	Data assigned to named entities (variables) Operations performed on data	Objects, which can be manipulated by a set of messages Messages sent to manipulate objects	Data assigned to named entities (variables) Operations performed on data	Arguments Applications of functions to their arguments	Elements of sets Predicates declared to them
Style	Procedural				Declarative	
Description						
Assumed execution model						
Interpretation of how to perform the computation	Starting with the initial state, a sequence of subsequent states will be performed until a final state is reached	The given sequence of instructions will be executed using the input data	The given sequence of messages will be executed	Input data flows through the dataflow graph and, as a consequence, output data is produced	The function with the given arguments is subsequently rewritten by substituting the function definitions referenced and performing the possible reductions until no more substitutions or reductions are feasible	The goal statement is subsequently rewritten using resolution until a logical interference is found or no more unifications are possible
Execution semantics	State transition semantics	State transition semantics	State transition semantics	Dataflow semantics	Reduction semantics	SLD-resolution
Control of the execution sequence	Control driven	Control driven	Control driven	Data driven (eager evalution)	Demand driven (lazy evalution)	Defined by both the goal processing rule and search rule
Concurrency	Sequential			Parallel		

The von Neumann computational model

- Basic items of computation are **data**
  - variables (named data entities)
  - memory or register locations whose addresses correspond to the names of the variables
  - data container
  - multiple assignments of data to variables are allowed
- Problem description model is **procedural** (sequence of instructions)
- Execution model is state transition semantics
  - Finite State Machine

von Neumann model vs. finite state machine

- As far as execution is concerned the von Neumann model behaves like a finite state machine (FSM)

- FSM = { I, G, δ, G_0, G_f }
- I: the input alphabet, given as the set of the instructions
- G: the set of the state (global), data state space D, control state space C, flags state space F, G = D x C x F

$\forall$ δ: the transition function: δ: I x G $\rightarrow$ G

- G_0 : the initial state
- G_f : the final state

Key characteristics of the von Neumann model

- Consequences of multiple assignments of data
  - history sensitive
  - side effects
- Consequences of control-driven execution
  - computation is basically a sequential one
- ++ easily be implemented
- Related language
  - allow declaration of variables with multiple assignments
  - provide a proper set of control statements to implement the control-driven mode of execution

Extensions of the von Neumann computational model

- new abstraction of parallel execution
- communication mechanism allows the transfer of data between executable units
  - unprotected shared (global) variables
  - shared variables protected by modules or monitors
  - message passing, and
  - rendezvous
- synchronization mechanism
  - semaphores
  - signals
  - events
  - queues
  - barrier synchronization

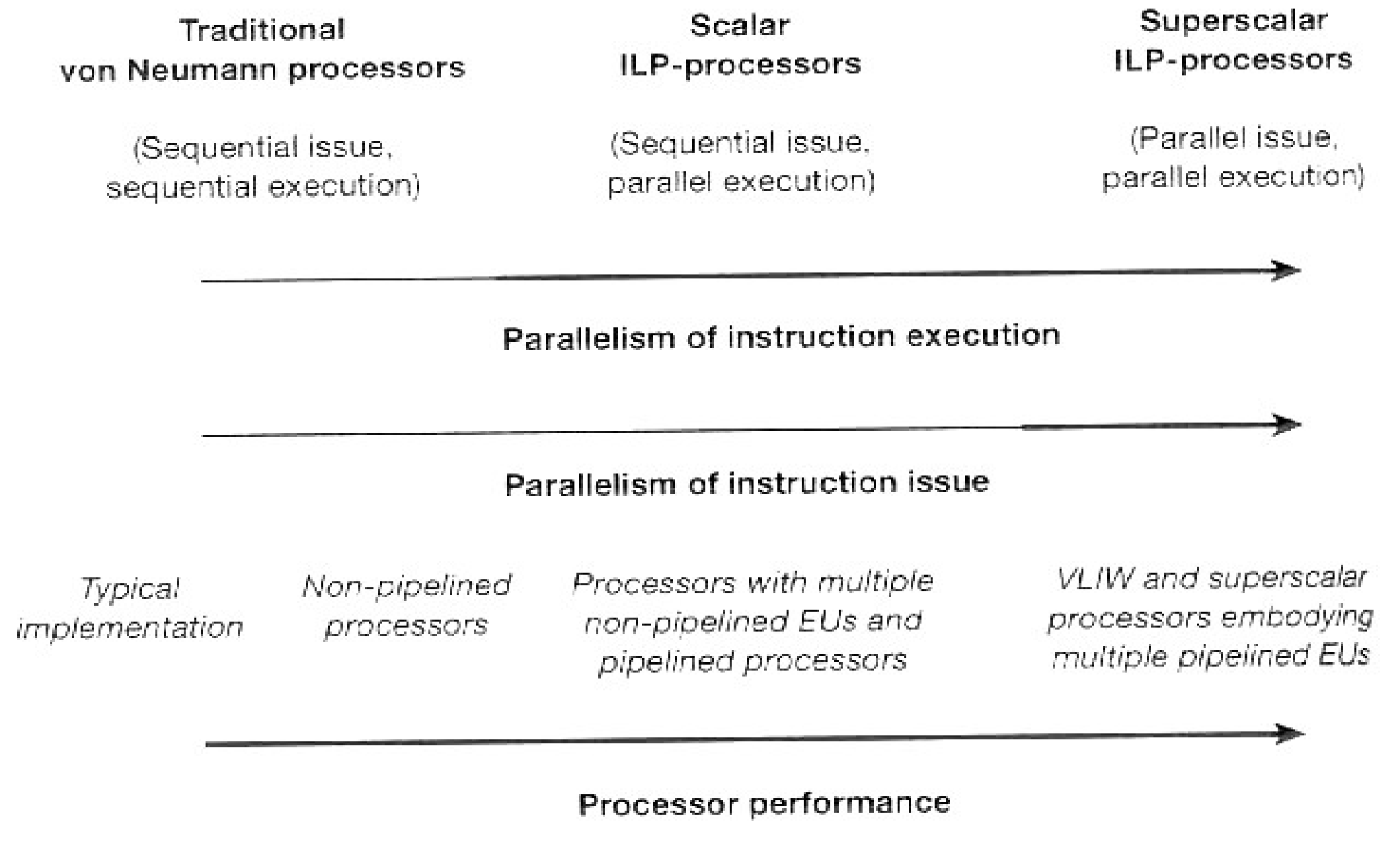
Instruction-Level Parallel Processors

{**Objective**: executing two or more instructions in parallel}

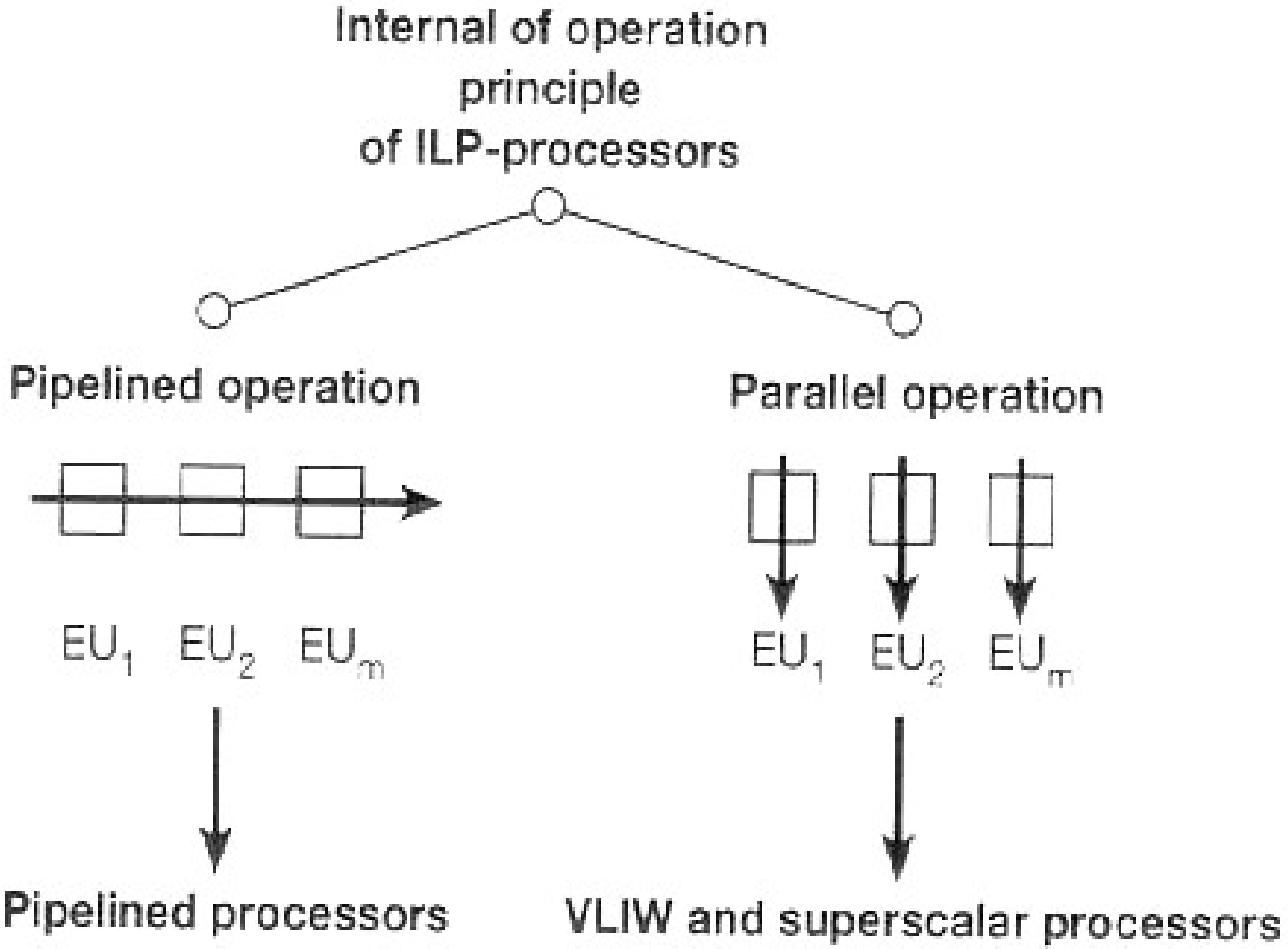
Improve CPU performance by:

- increasing clock rates
  - (CPU running at 2GHz!)
- increasing the number of instructions to be executed in parallel
  - (executing 6 instructions at the same time)

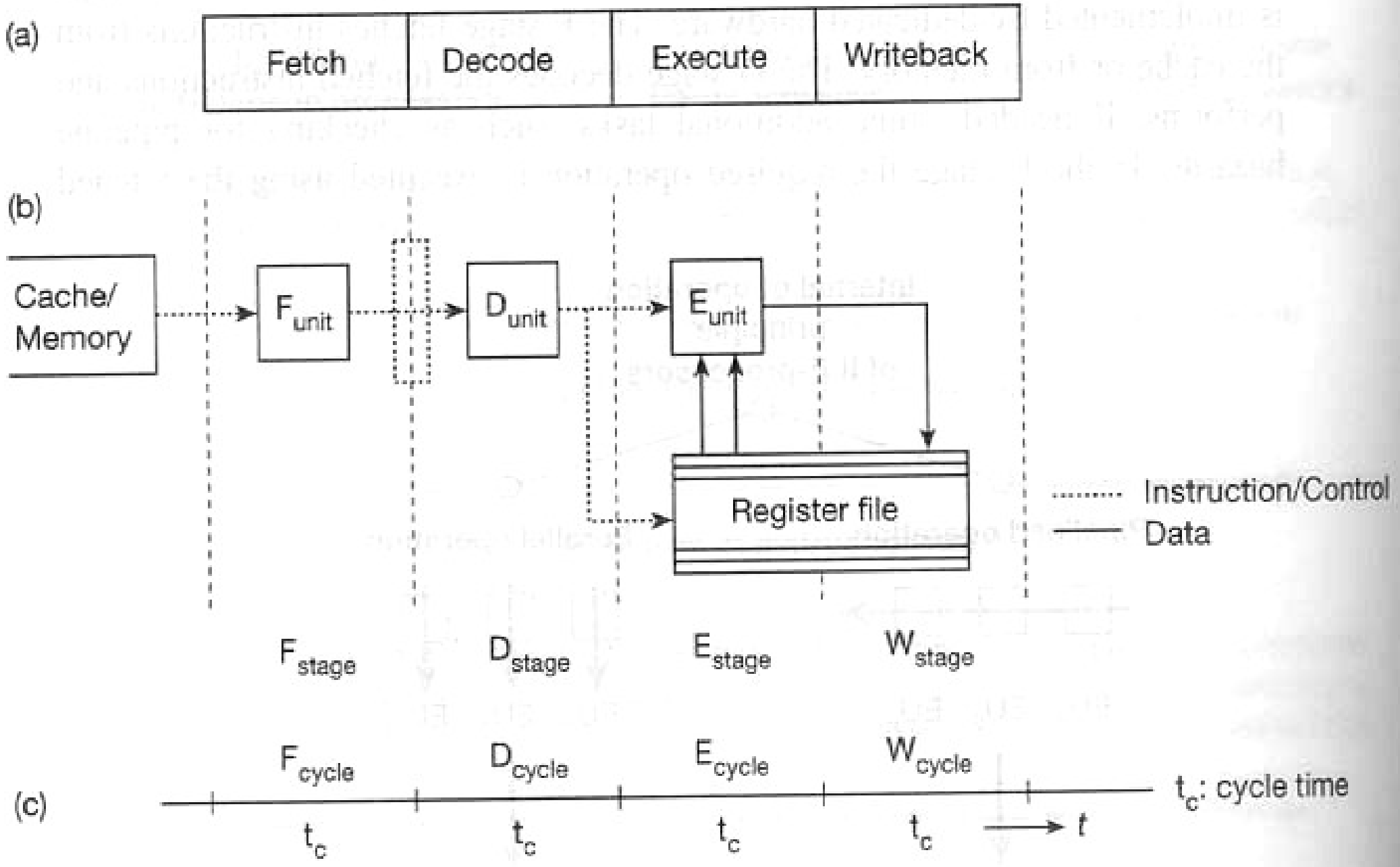
How do we increase the number of instructions to be executed?



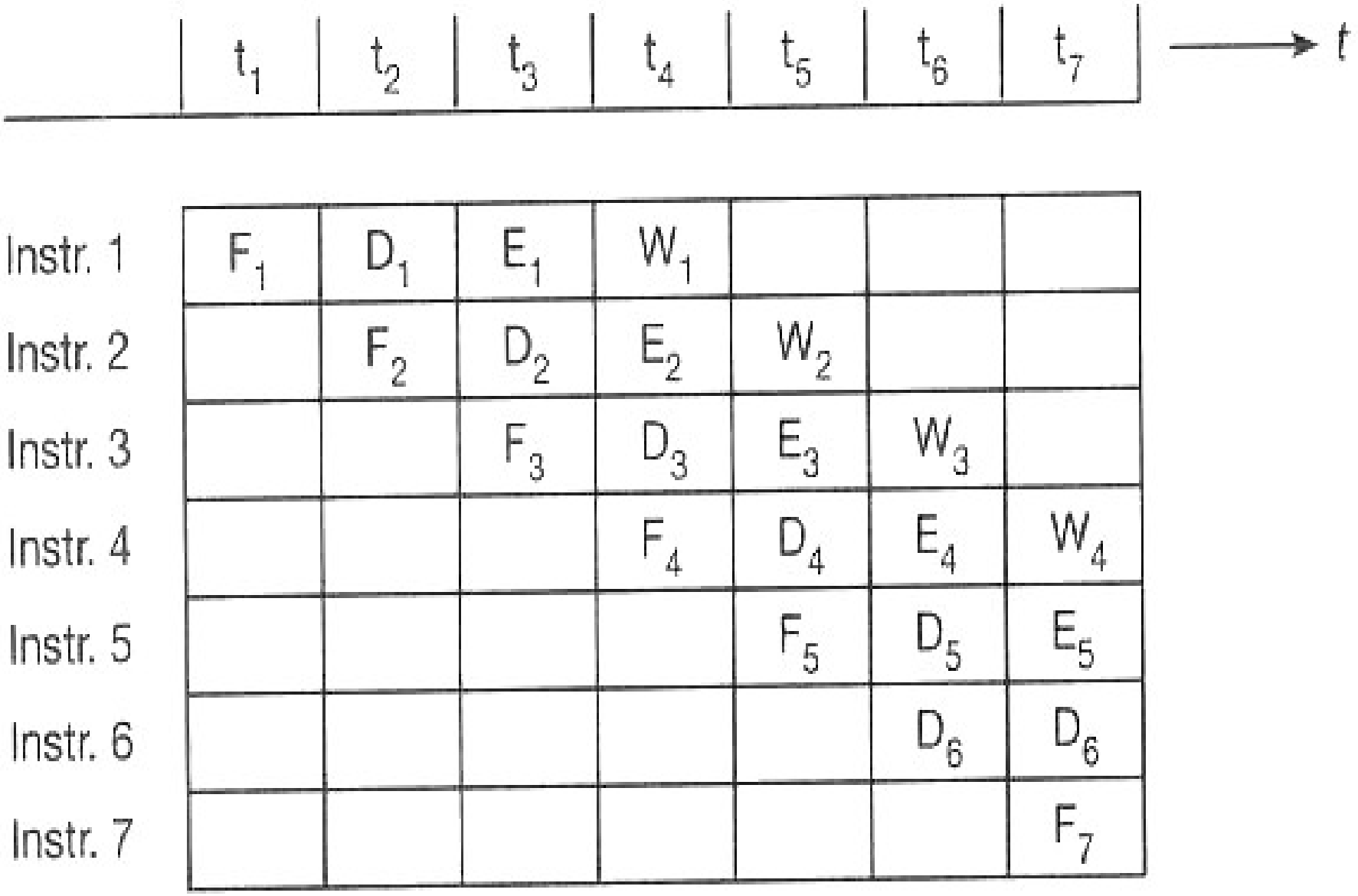
Time and Space parallelism



Pipeline (assembly line)



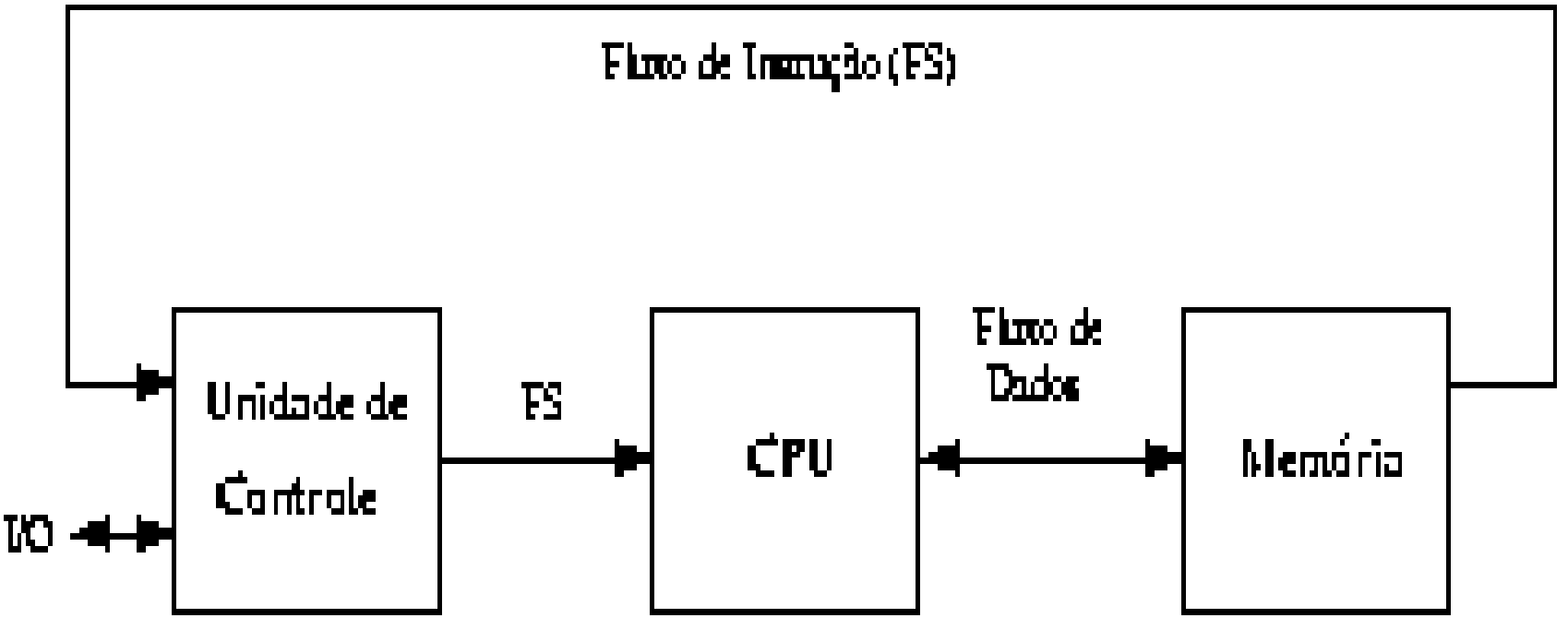
Result of pipeline (e.g.)



Flynn's Taxonomy

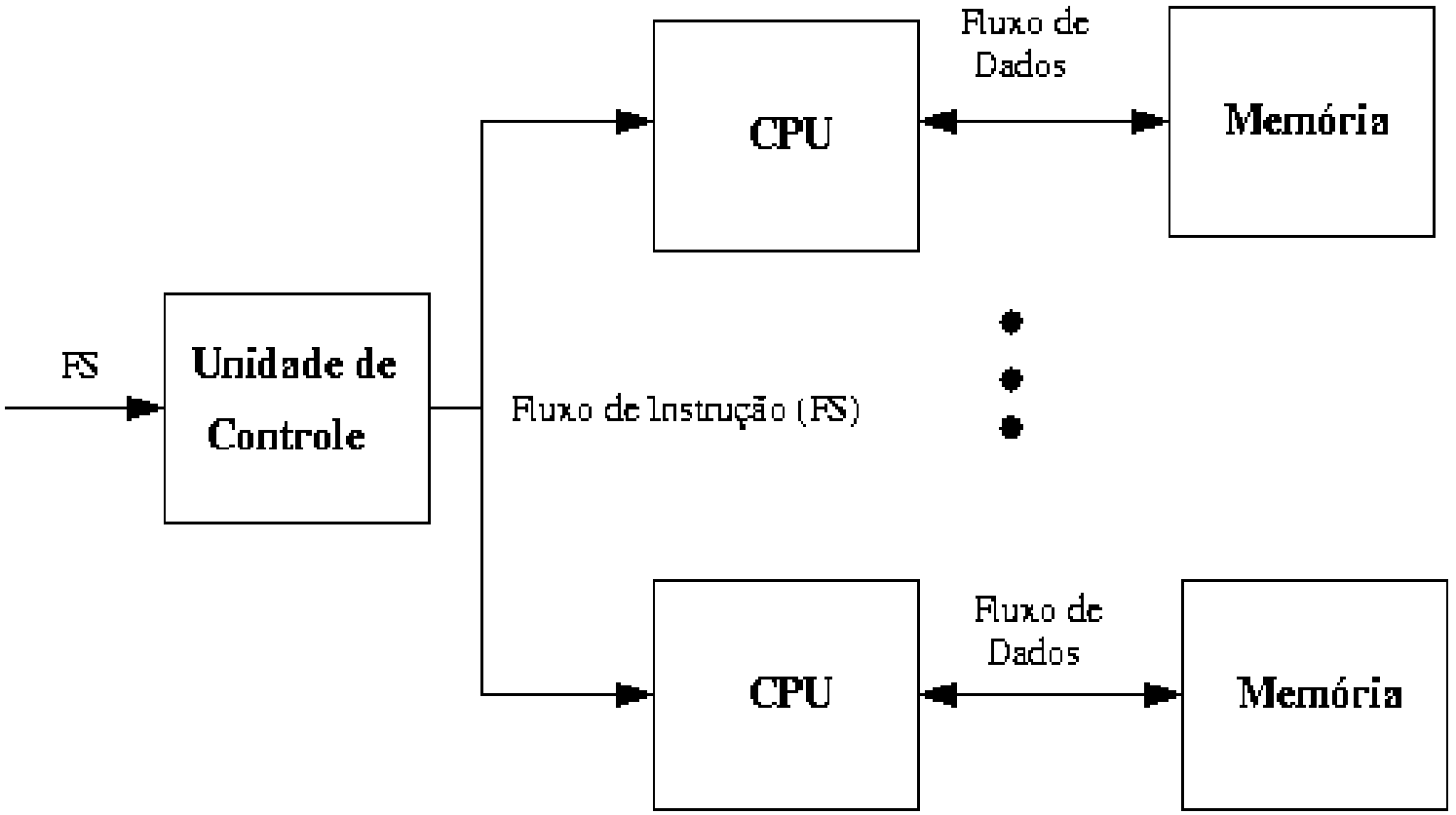
- Organizes the processor architectures according to the data and instruction stream
  - SISD - *Single Instruction, Single Data stream*
  - SIMD - *Single Instruction, Multiple Data stream*
  - MISD - *Multiple instruction, Single Data stream*
  - MIMD - *Multiple Instruction, Multiple Data stream*

SISD



Exemplos: Estações de trabalho e Computadores pessoais com um único processador

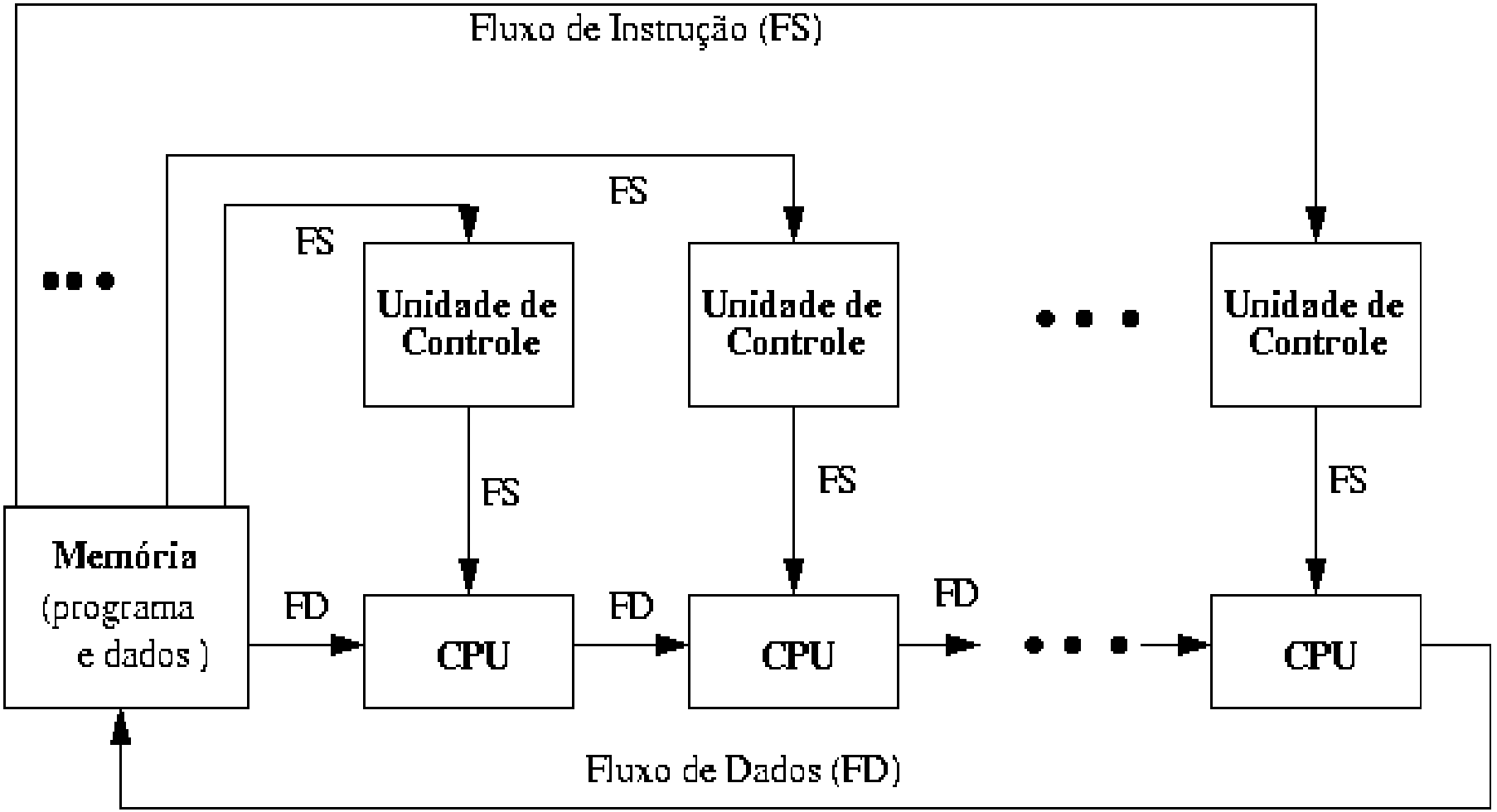
SIMD



Exemplos: ILIAC IV, MPP, DHP, MASPAR MP-2 e CPU Vetoriais

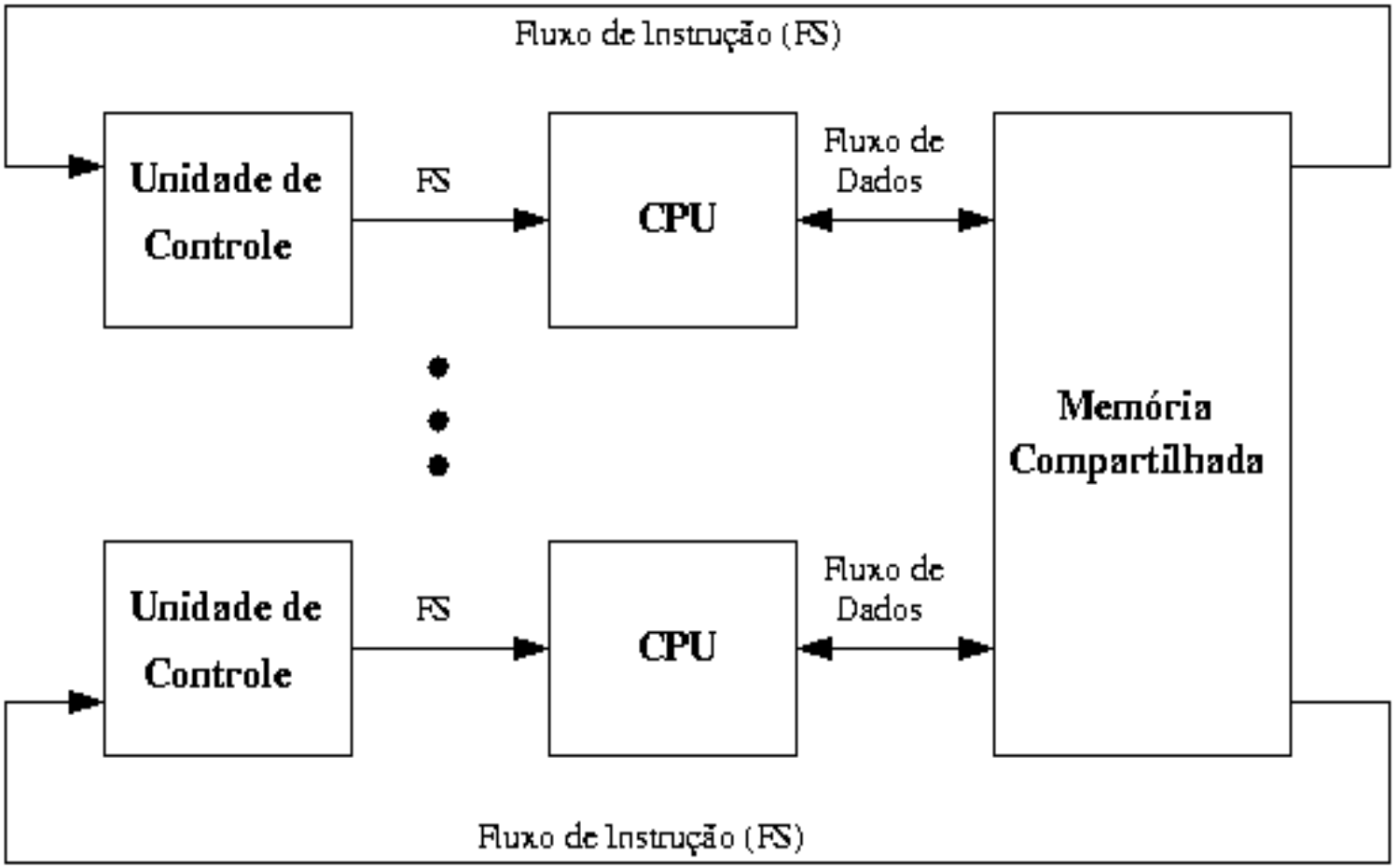
Extensões multimidia são Consideradas como uma forma de paralelismo SIMD

MISD



Exemplos: Arrays Sistólicos

MIMD



Exemplos: Cosmic Cube, nCube 2, iPSC, FX-2000, SGI Origin, Sun Enterprise 5000 e Redes de Computadores

Mais difundida
Memória Compartilhada
Memória Distribuída

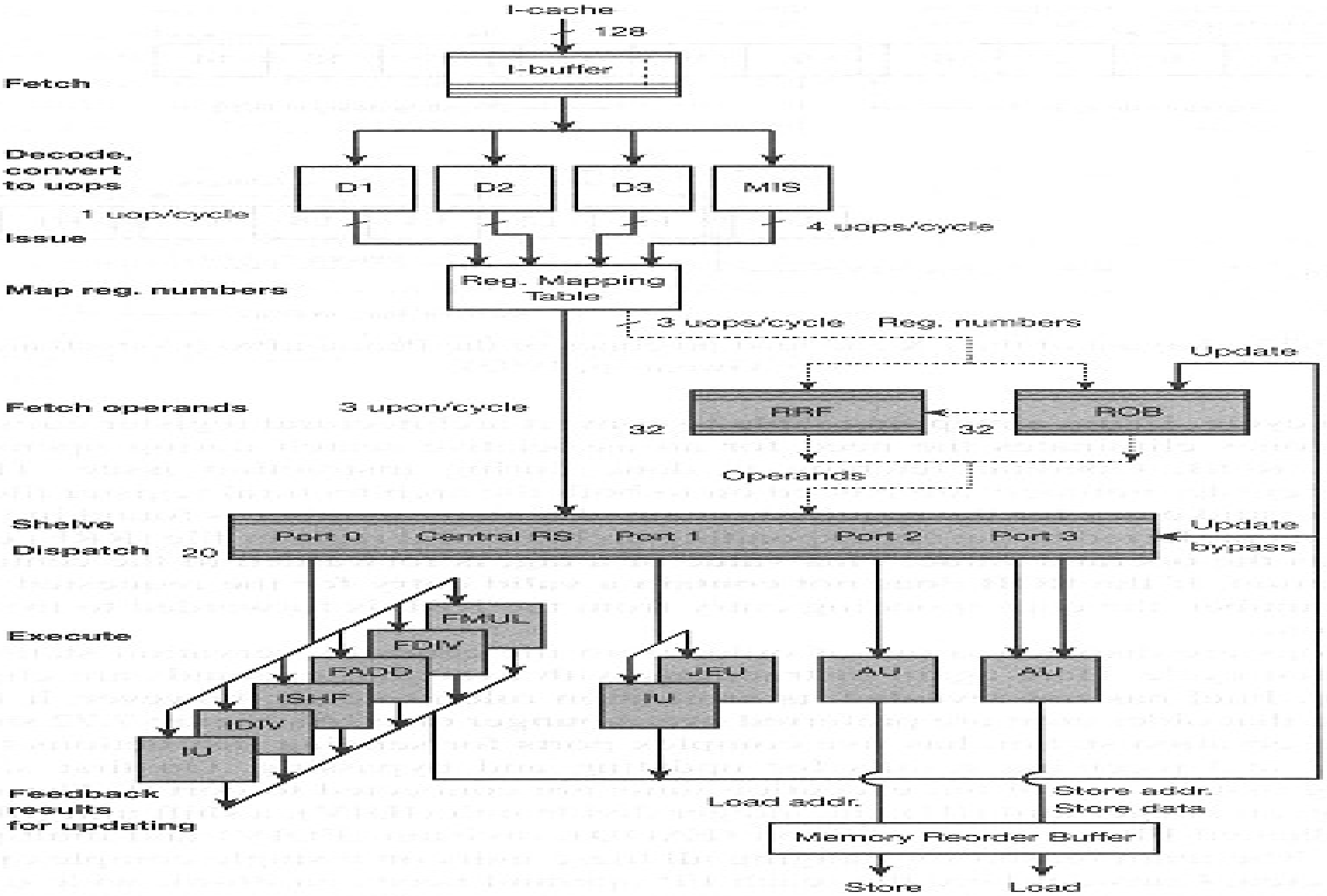
Flexível
Usa microprocessadores off-the-shelf

Resume

OBS.: The Flynn's taxonomy is a reference model. Actually, there exist some processors using features of more than one category. There are many Taxonomy proposals for Parallel Architectures. See R. Duncan's paper!

- MIMD
  - Most of the parallel machines
  - It seems adequate to the general purpose parallel computing
- SIMD
- MISD
- SISD
  - Sequential computing

Case Studies: PentiumPro
Core part of the micro-architecture



PentiumPro Long pipeline:
Layout of the FX and load pipelines