Computer Architectures

Scientific Computing

Jörn Behrens

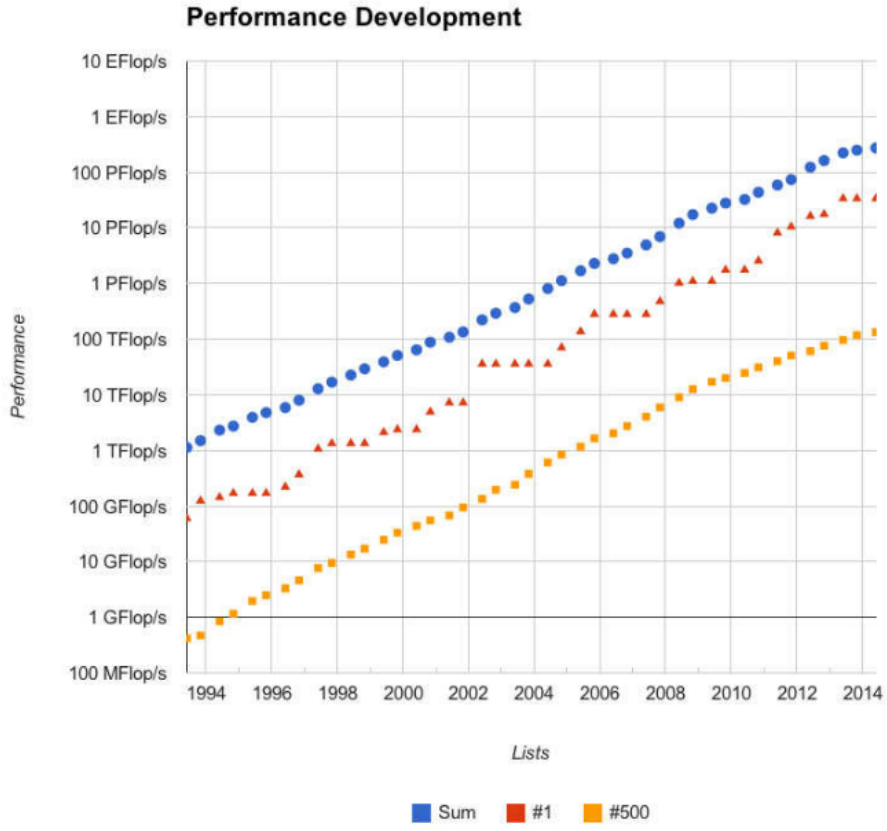
Introduction

Why deal with Computer Architectures?

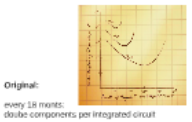
- Most programs achieve 10-20% of peak performance, why?
- How does a computer work, anyways?
- What makes the perfomance?

Why deal with Computer Architectures?

- Most programs achieve 10-20% of peak performance, why?
- How does a computer work, anyways?
- What makes the perfomance?

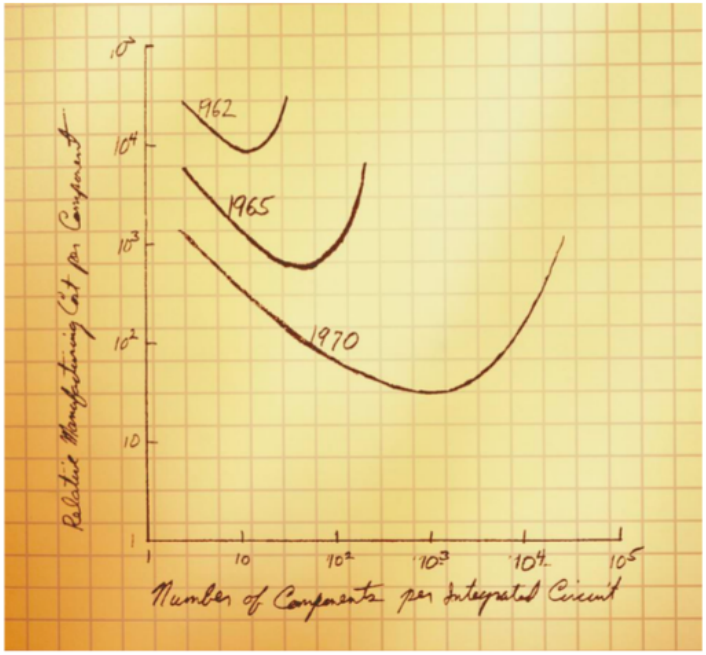


Moore's law



every 18 months:
double computational performance!

Moore's law

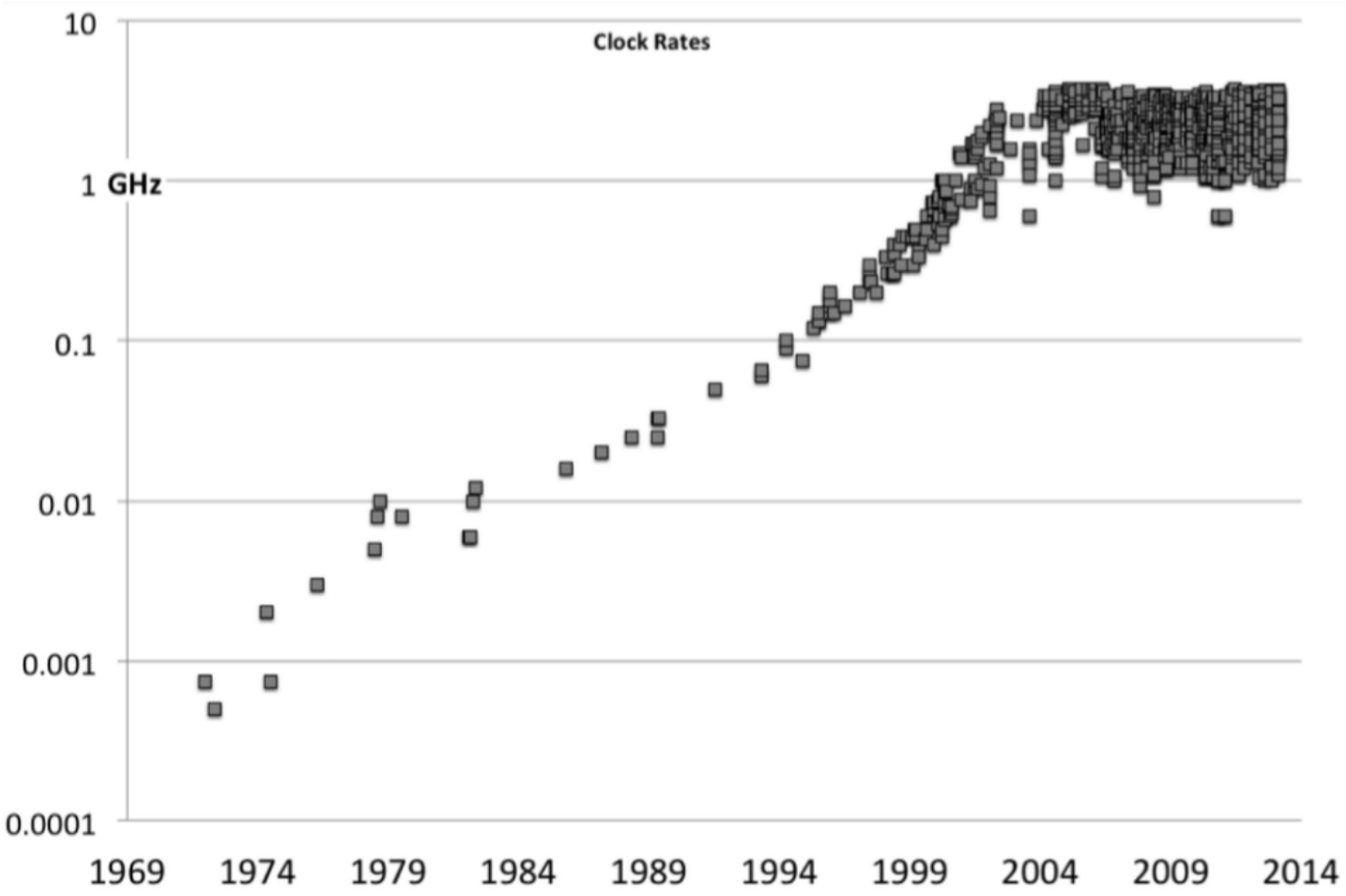


Original:

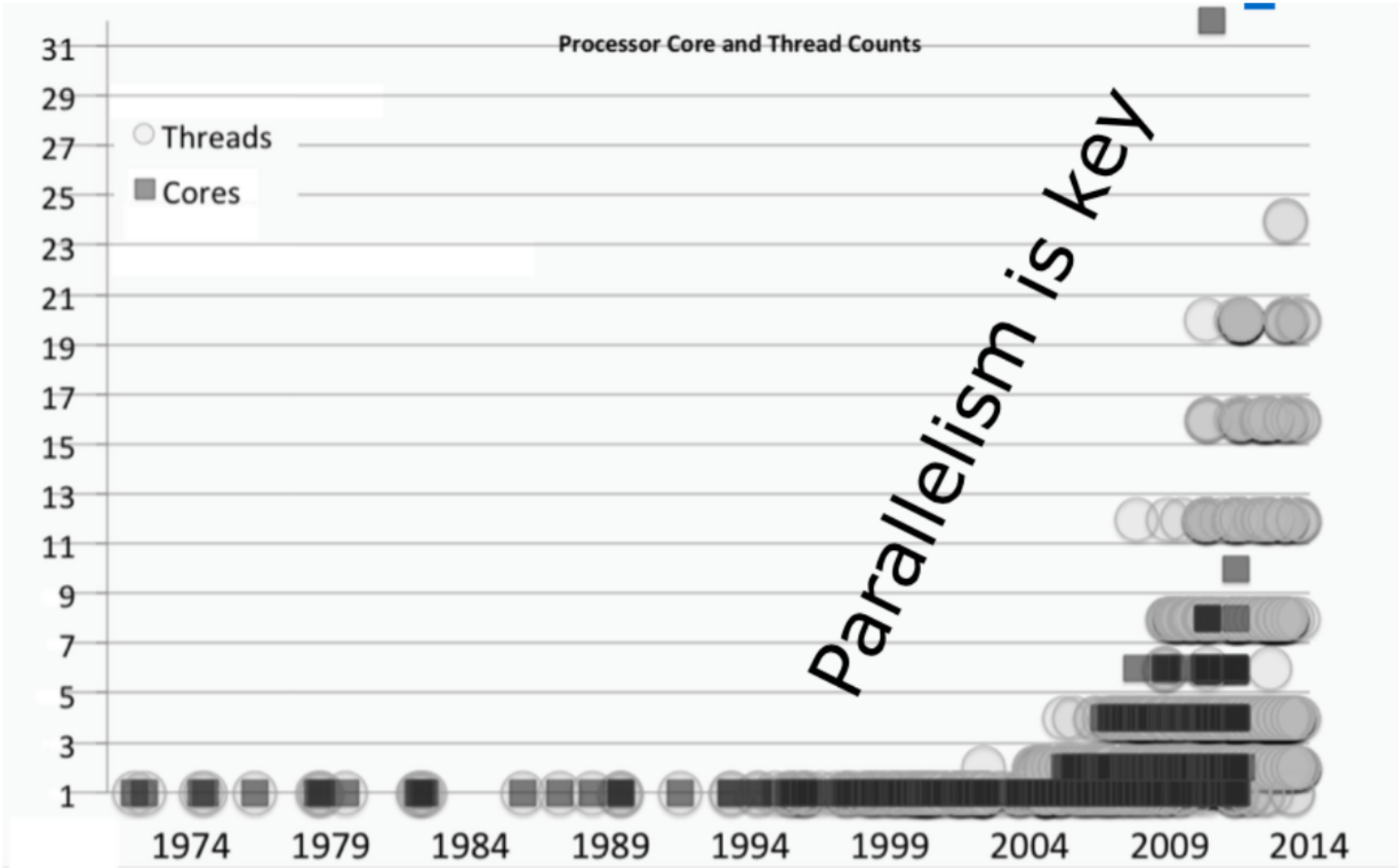
every 18 monts:
doube components per integrated circuit

every 18 months:
double computational performance!

Increase in Clockrate?

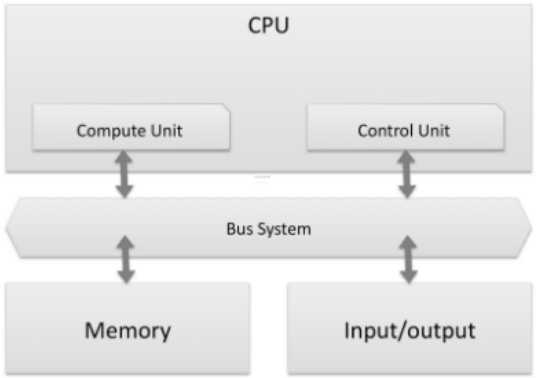


Parallelism is the Key



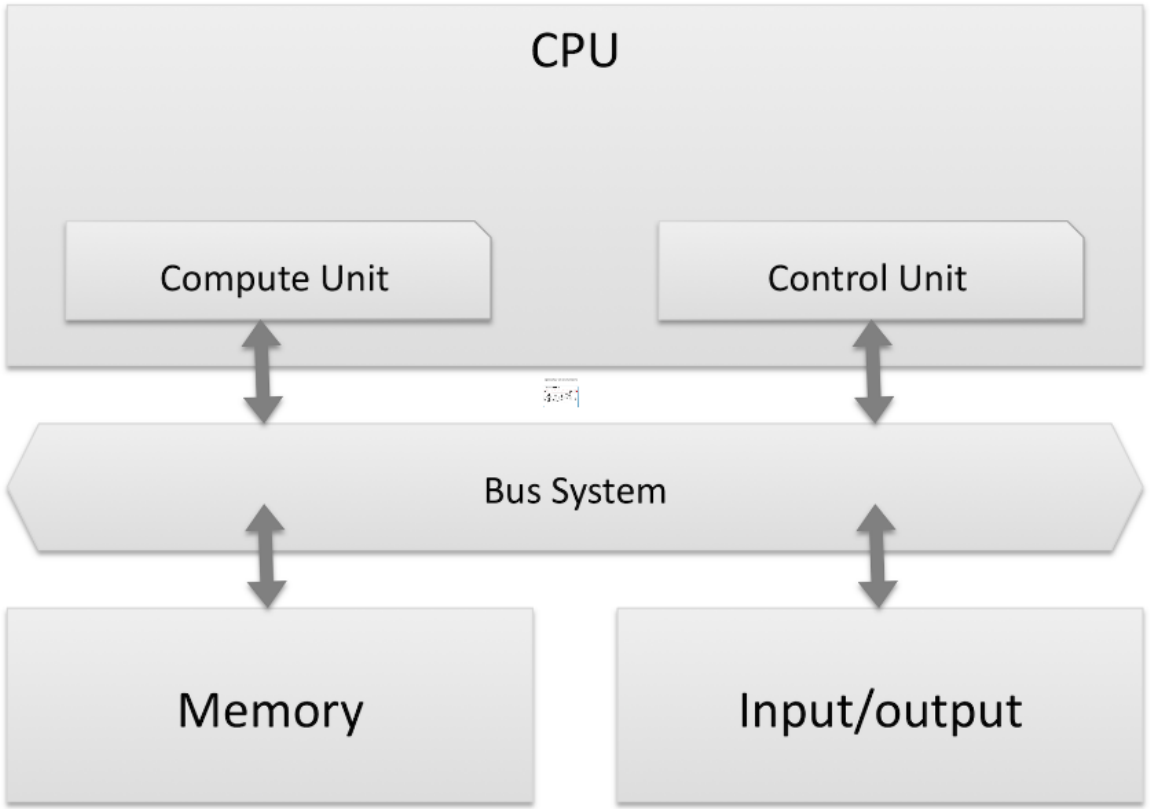
Basics

Von Neumann Architecture



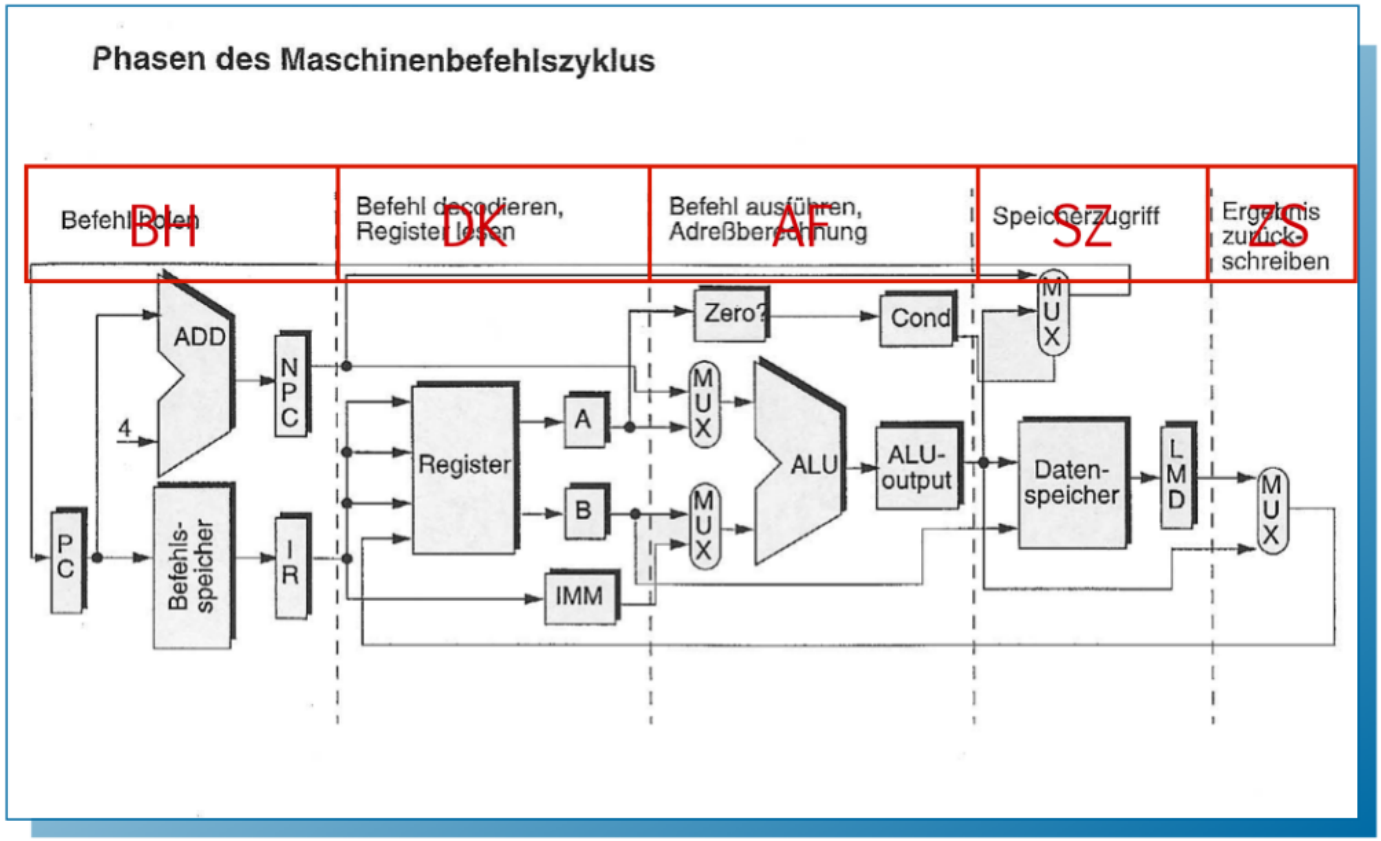
John von Neumann
*1903 ..1957

Von Neumann Architecture

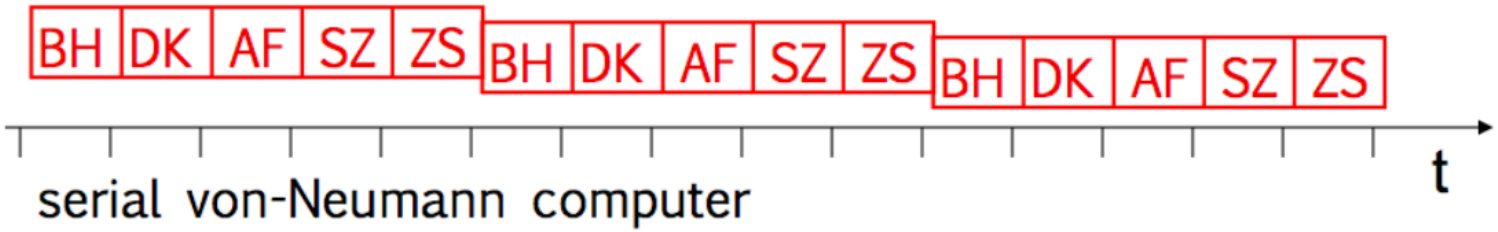


John von Neumann
*1903 ..1957

Operation via Instructions



How to accelerate?



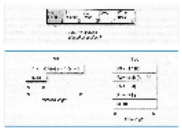
What are your ideas?

Acceleration Features

Parallelism on the processor level

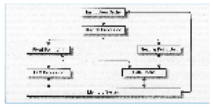
RISC vs. CISC

reduce instruction complexity

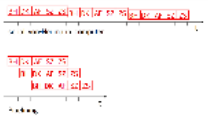


Super-scalar processors

- One floating point instruction
- One fixed point instruction (or memory reference)
- One condition code manipulation
- A branch instruction

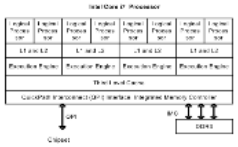


Pipelining

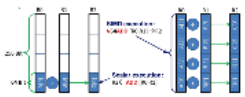


Parallel Pipelines

Parallel Units on Chip

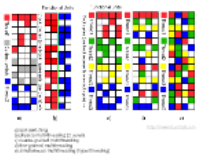


Vector Units (SIMD)

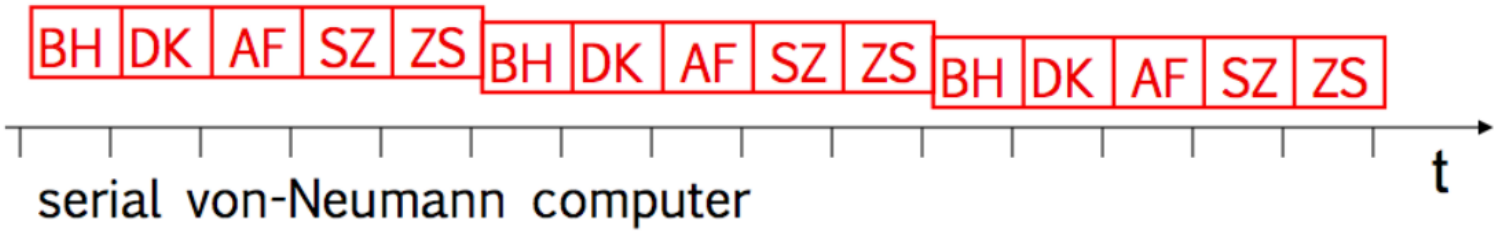


Multithreading/Hyperthreading

Multithreading/Hyperthreading



Pipelining

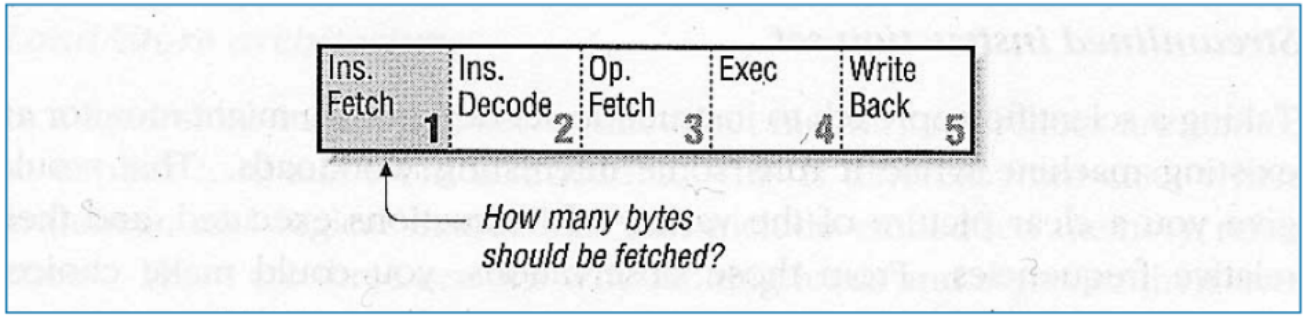


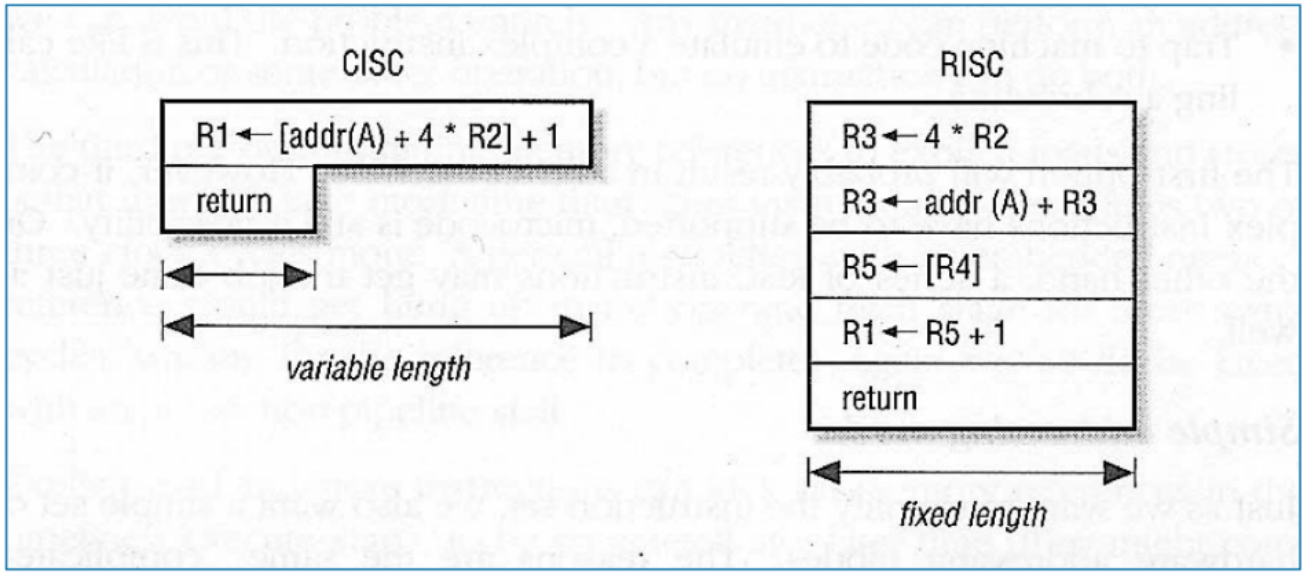
serial von-Neumann computer

Pipelining

RISC vs. CISC

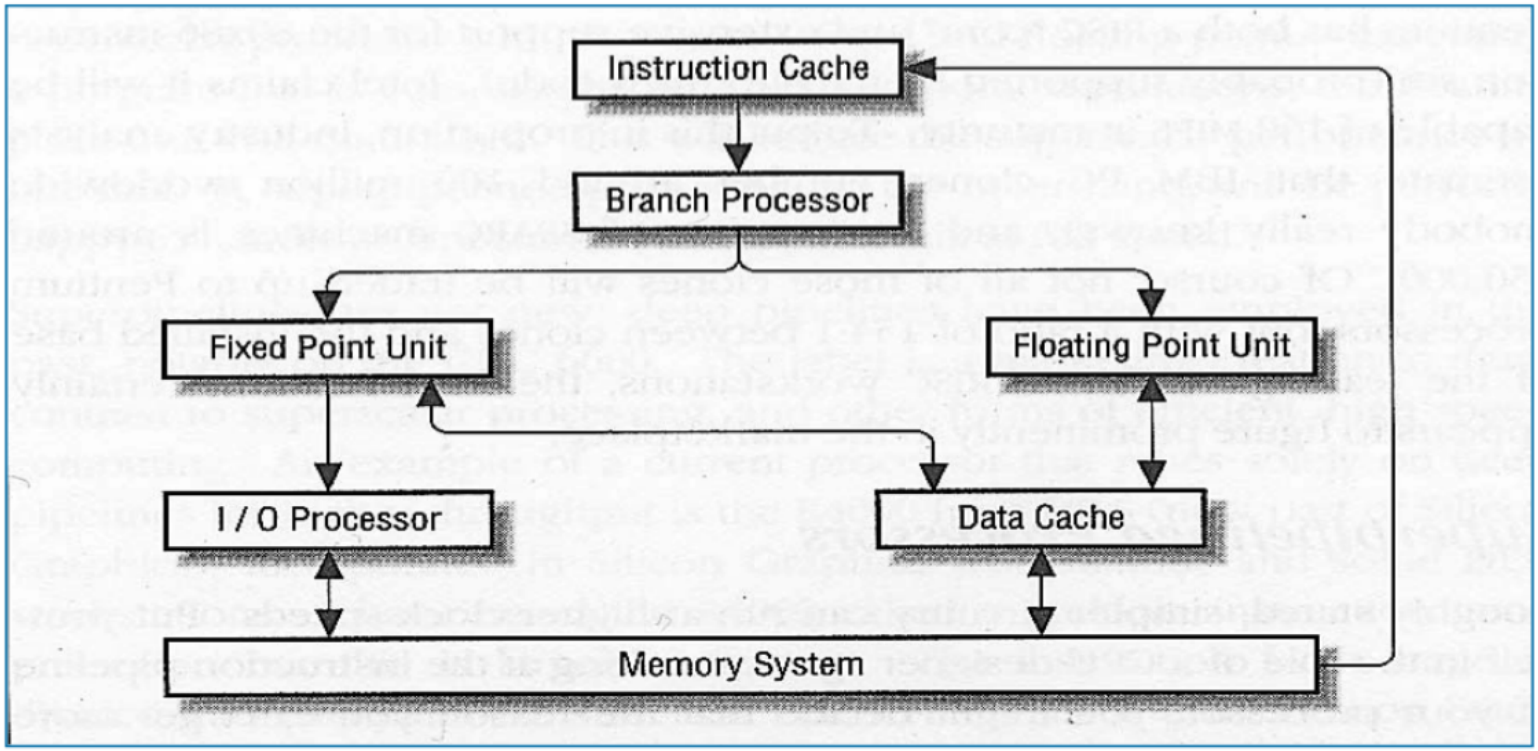
reduce instruction complexity



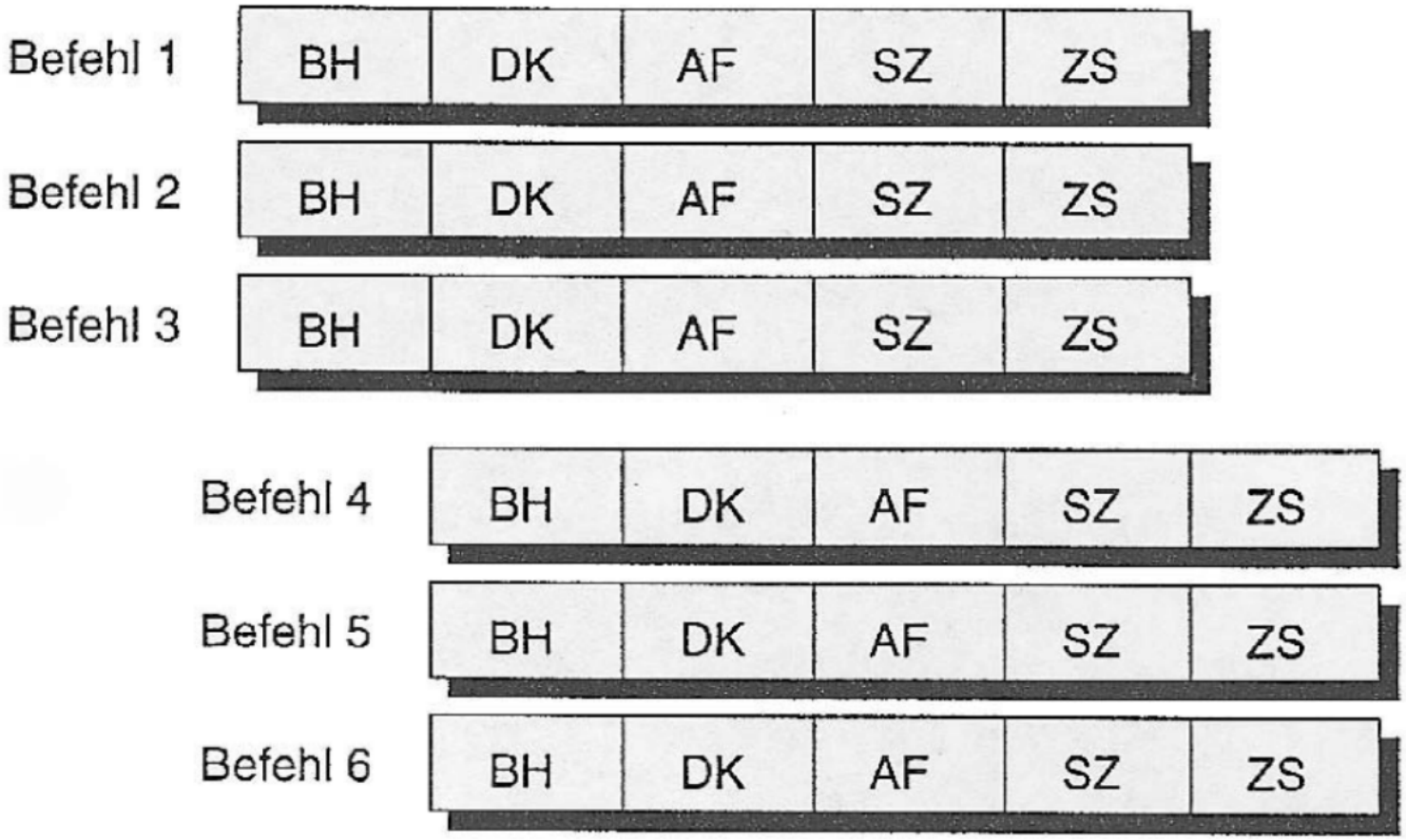


Super-scalar processors

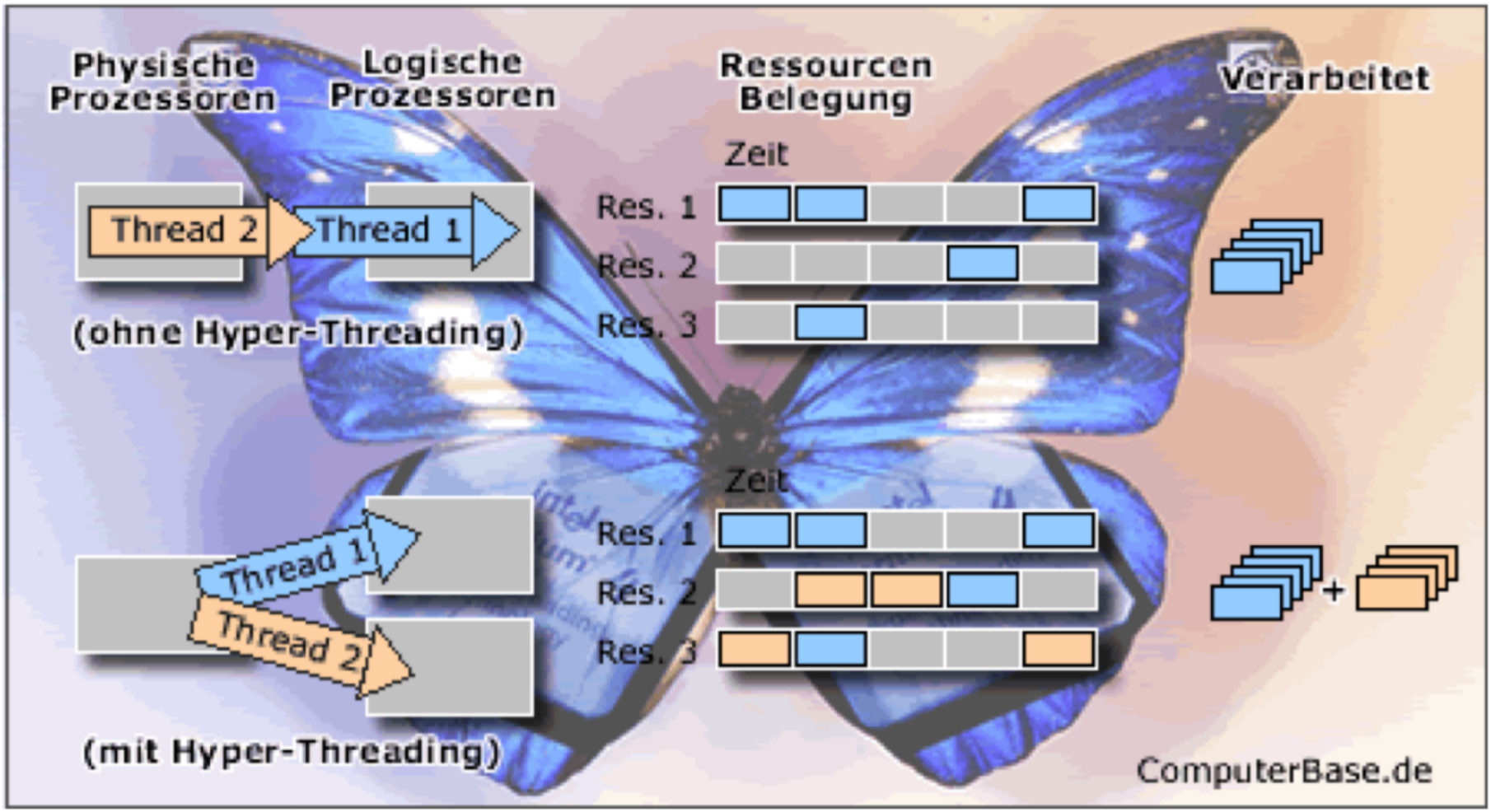
- One floating point instruction
- One fixed point instruction (or memory reference)
- One condition code manipulation
- A branch instruction



Parallel Pipelines

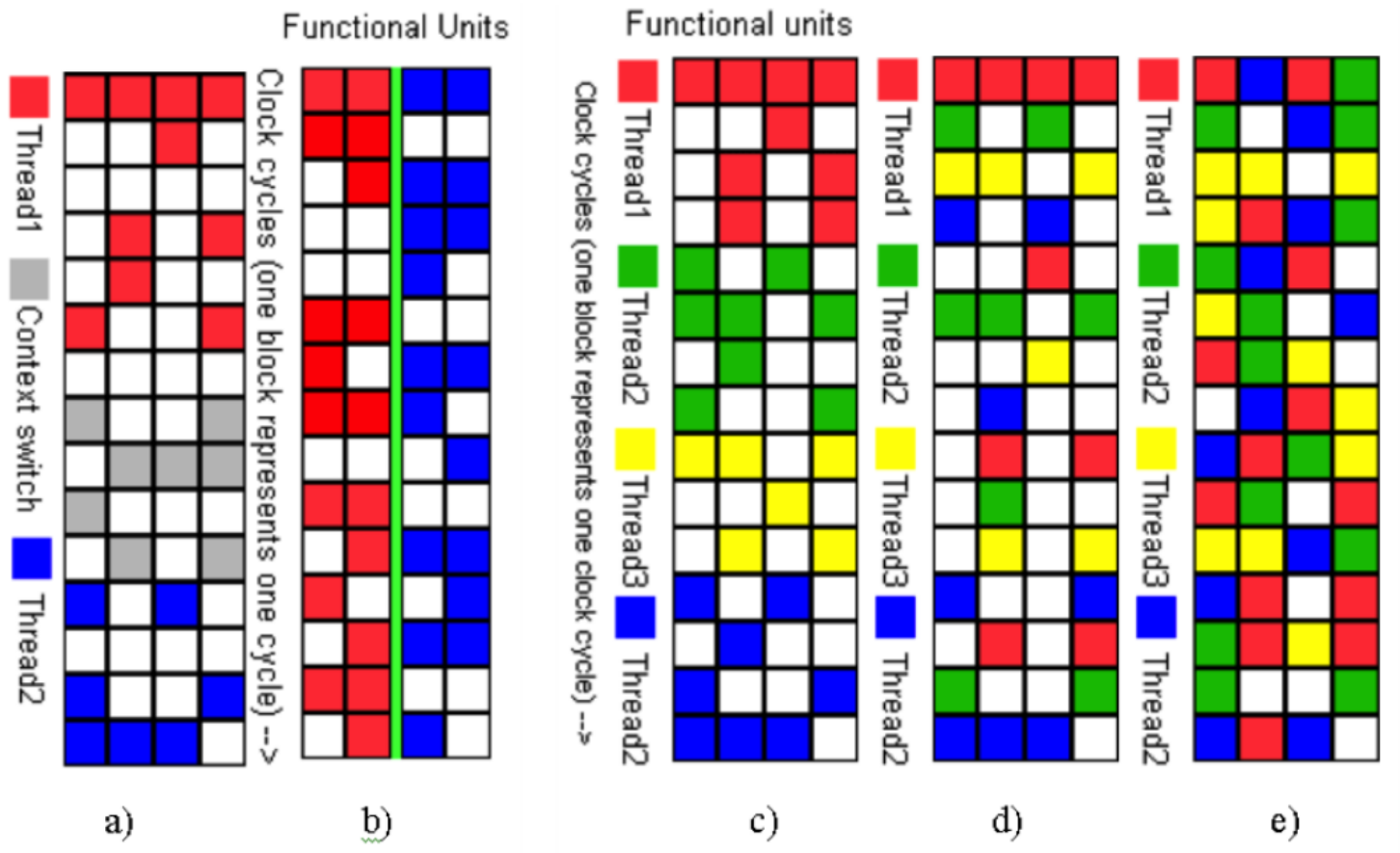


Multithreading/Hyperthreading



http://www.computerbase.de/

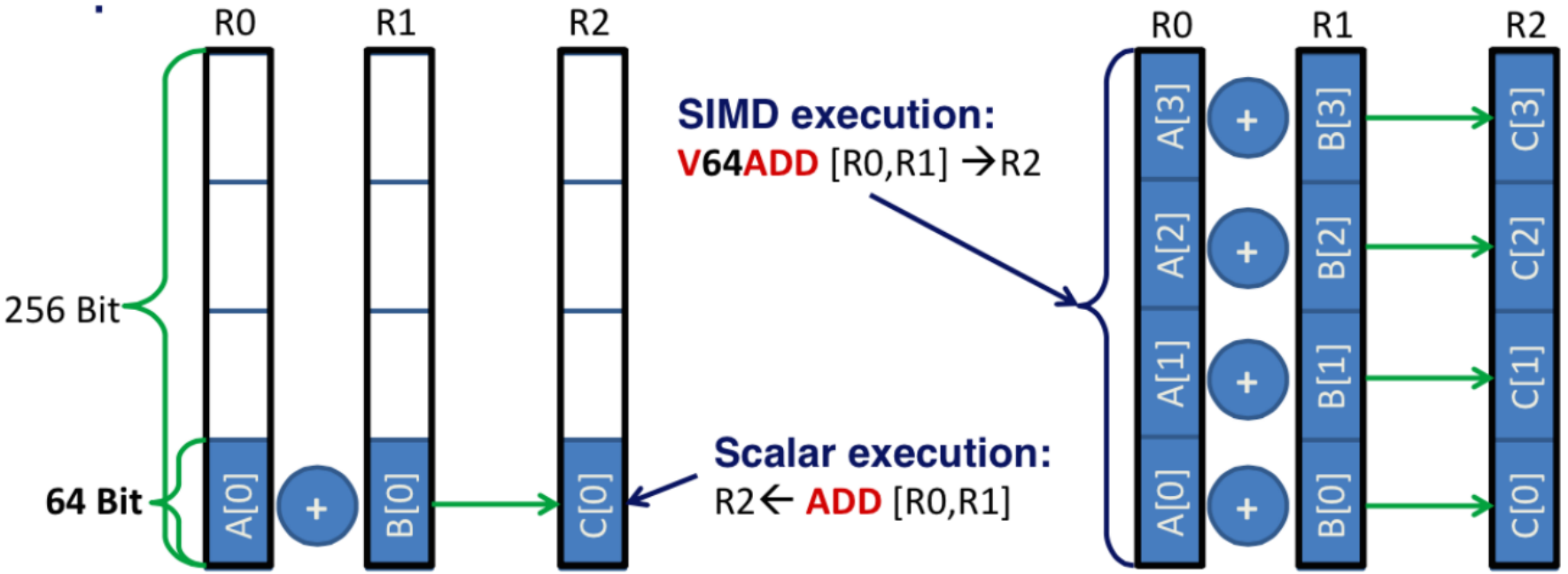
Multithreading/Hyperthreading



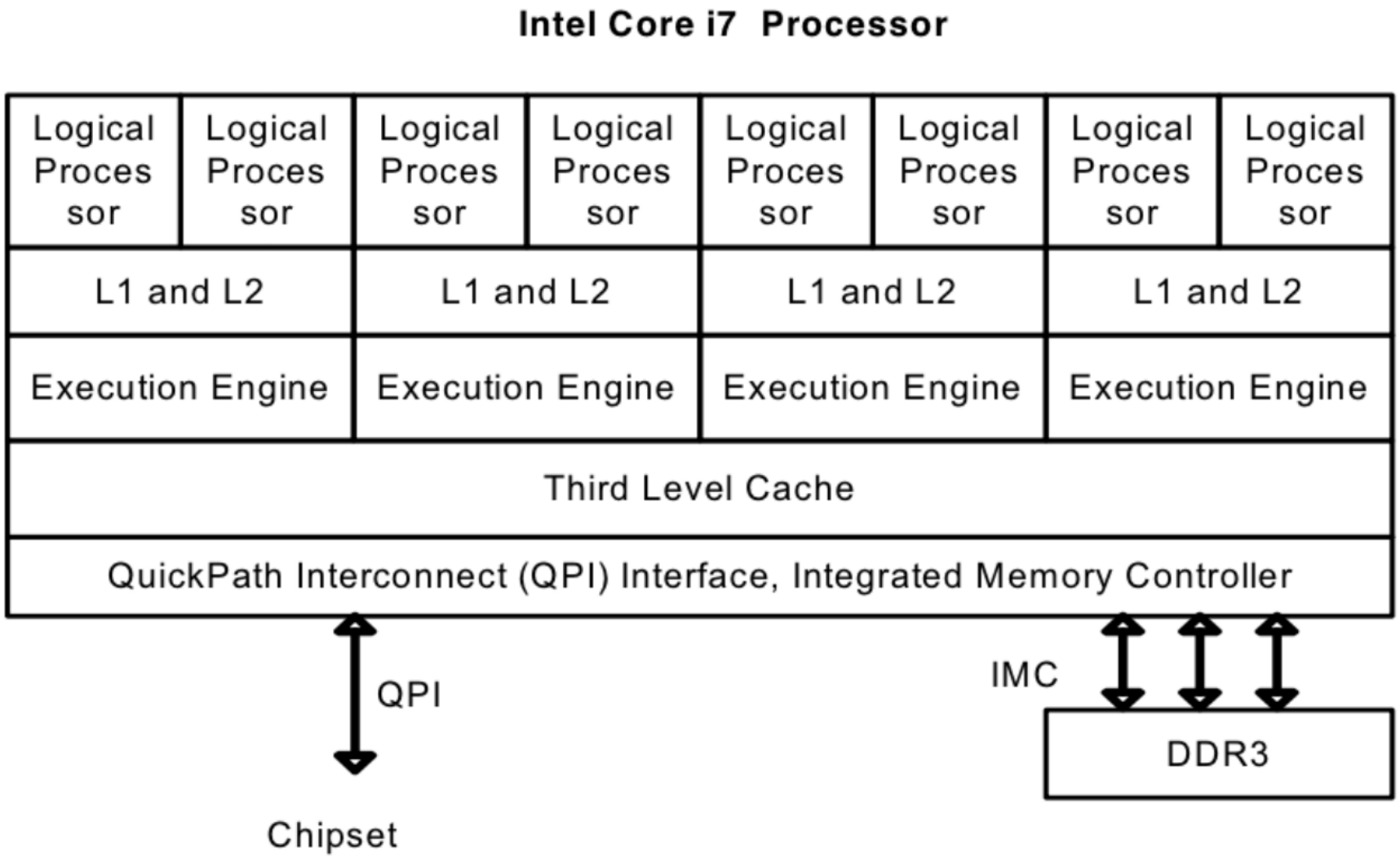
http://www.slcentral.com

a) task switching
b) physica multi-threading (2 cores)
c) coarse-grained multithreading
d) fine-grained multithreading
e) simultaneous multithreading (hyperthreading)

Vector Units (SIMD)



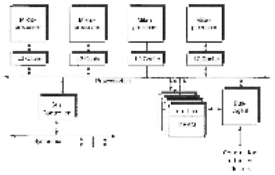
Parallel Units on Chip



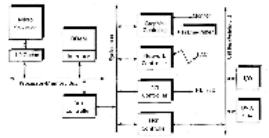
System Architectures

Replicating functional units

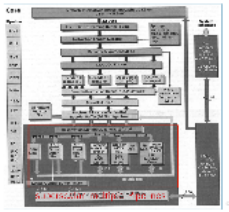
Multiprocessor Server Architecture



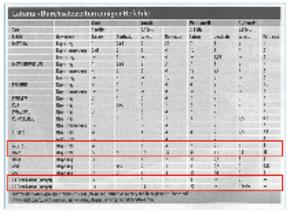
Worksstation Architecture



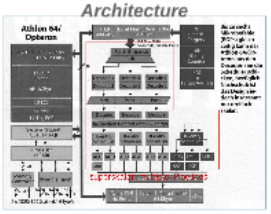
Intel Core Architecture



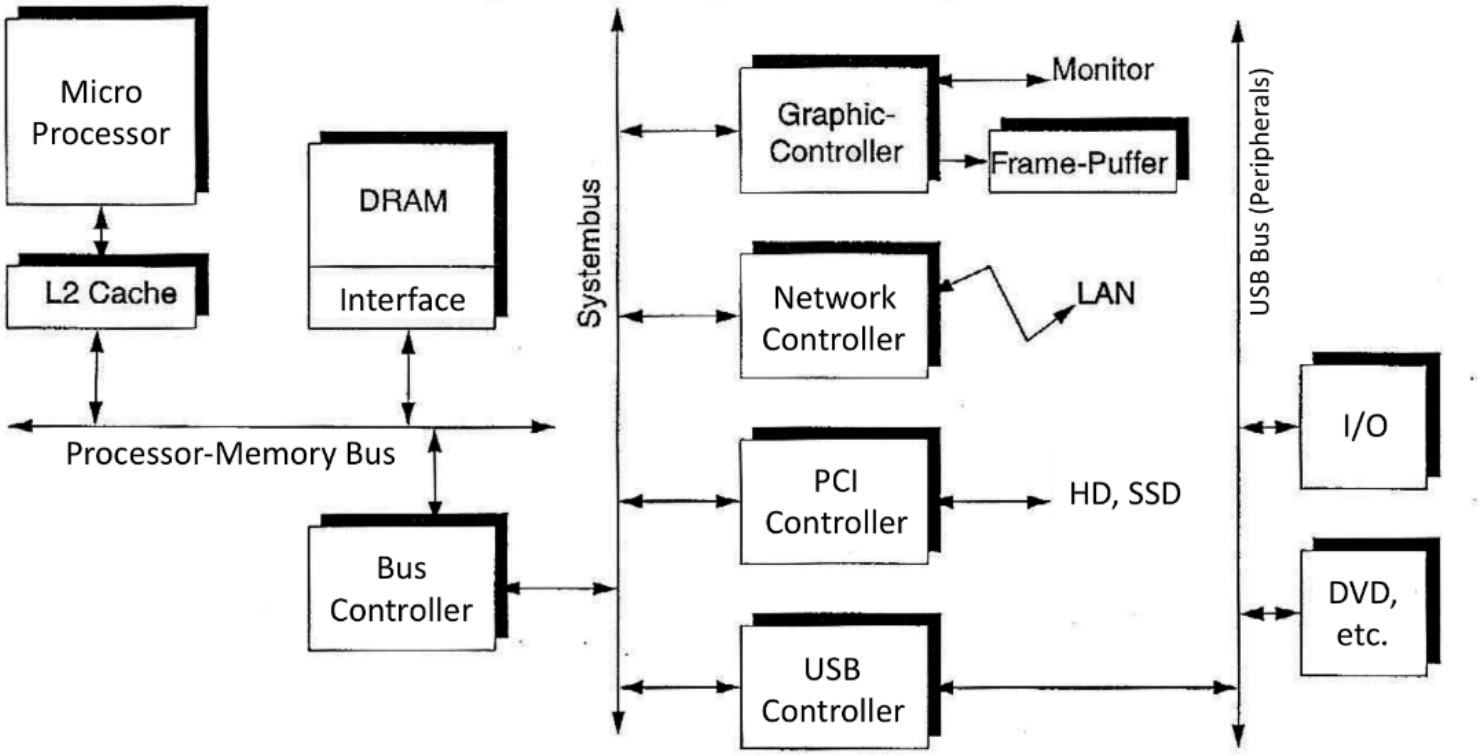
Latencies and Performance Indicators



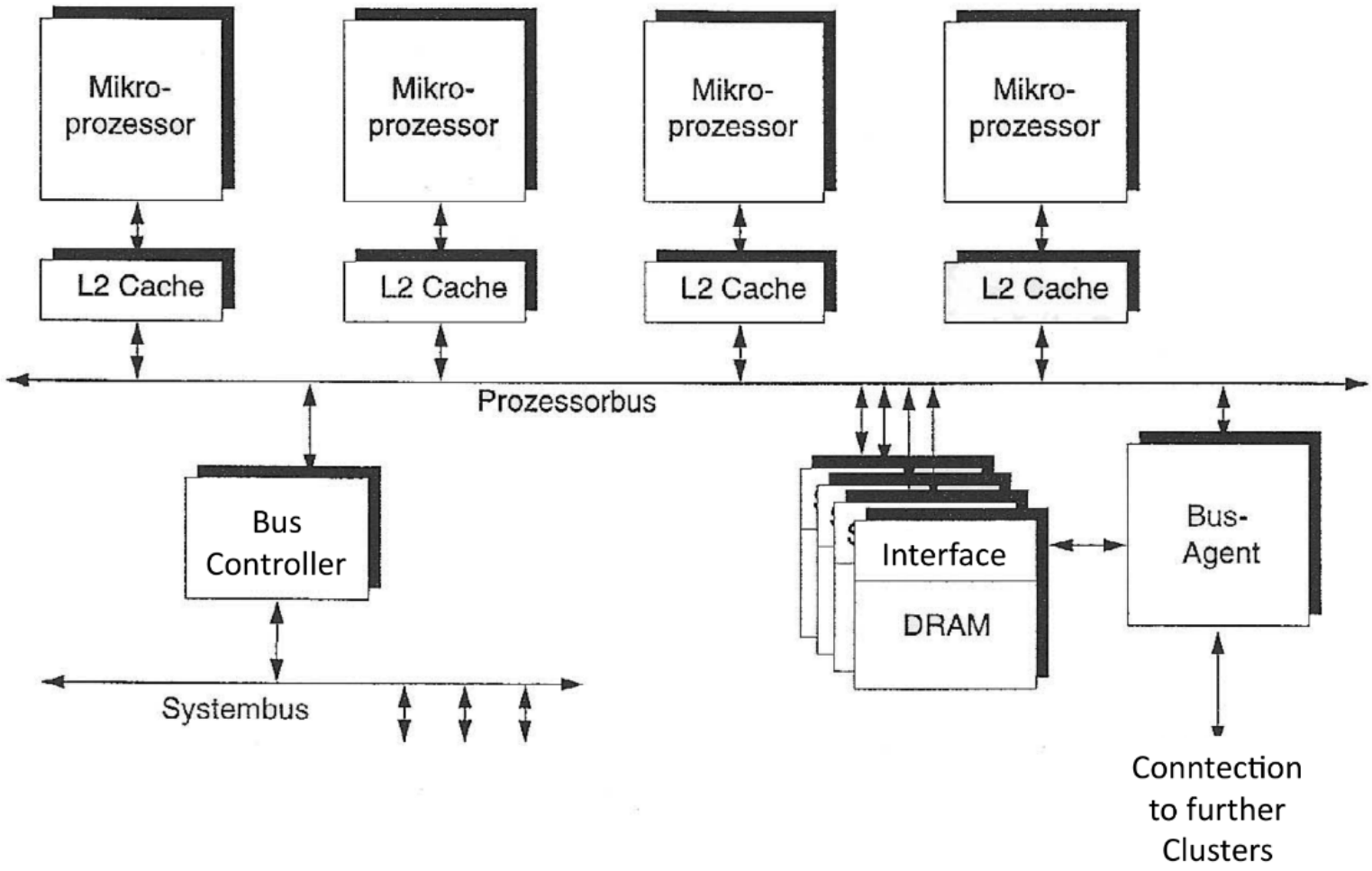
AMD Athlon/Opteron Architecture



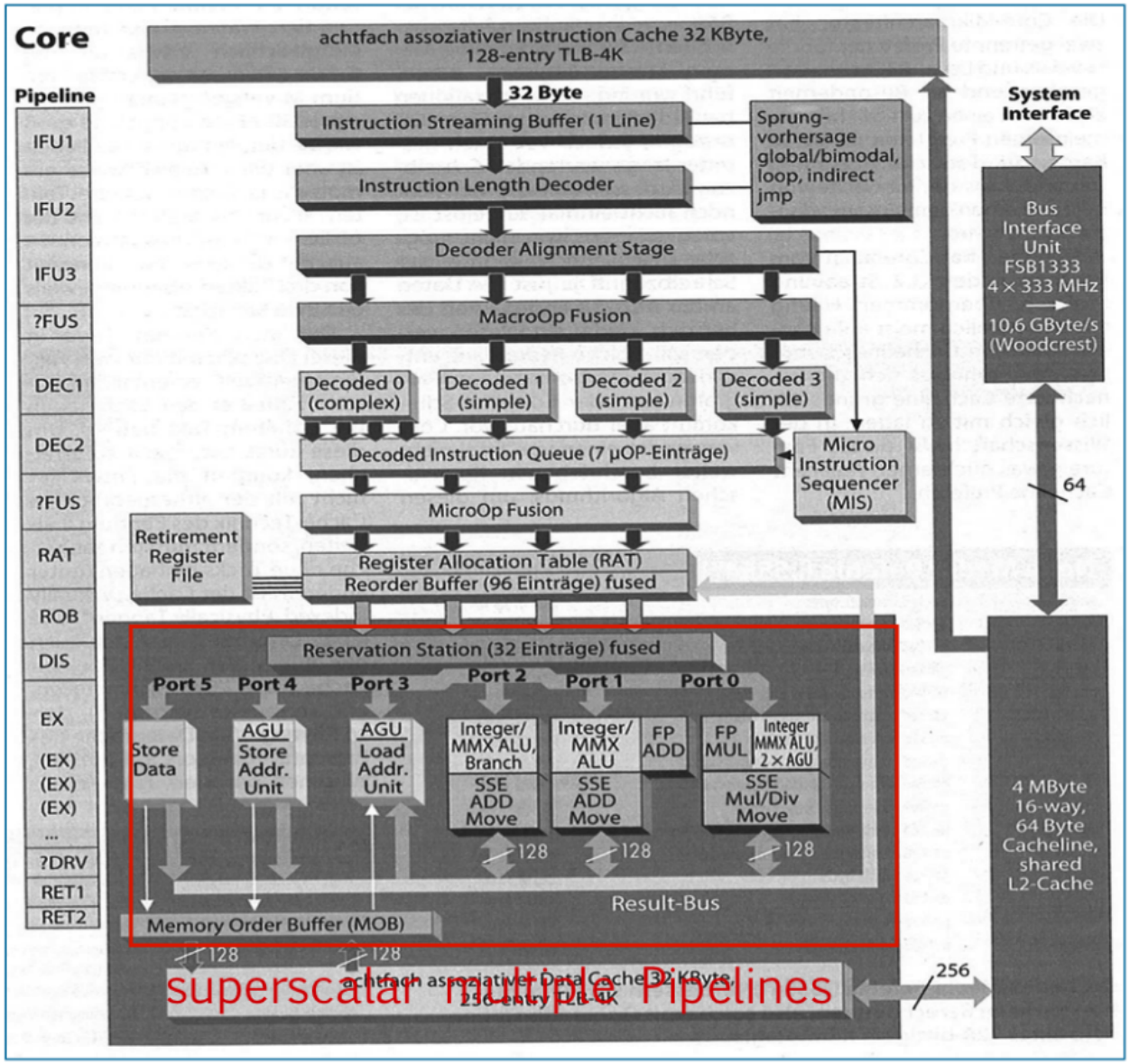
Worksstation Architecture



Multiprocessor Server Architecture

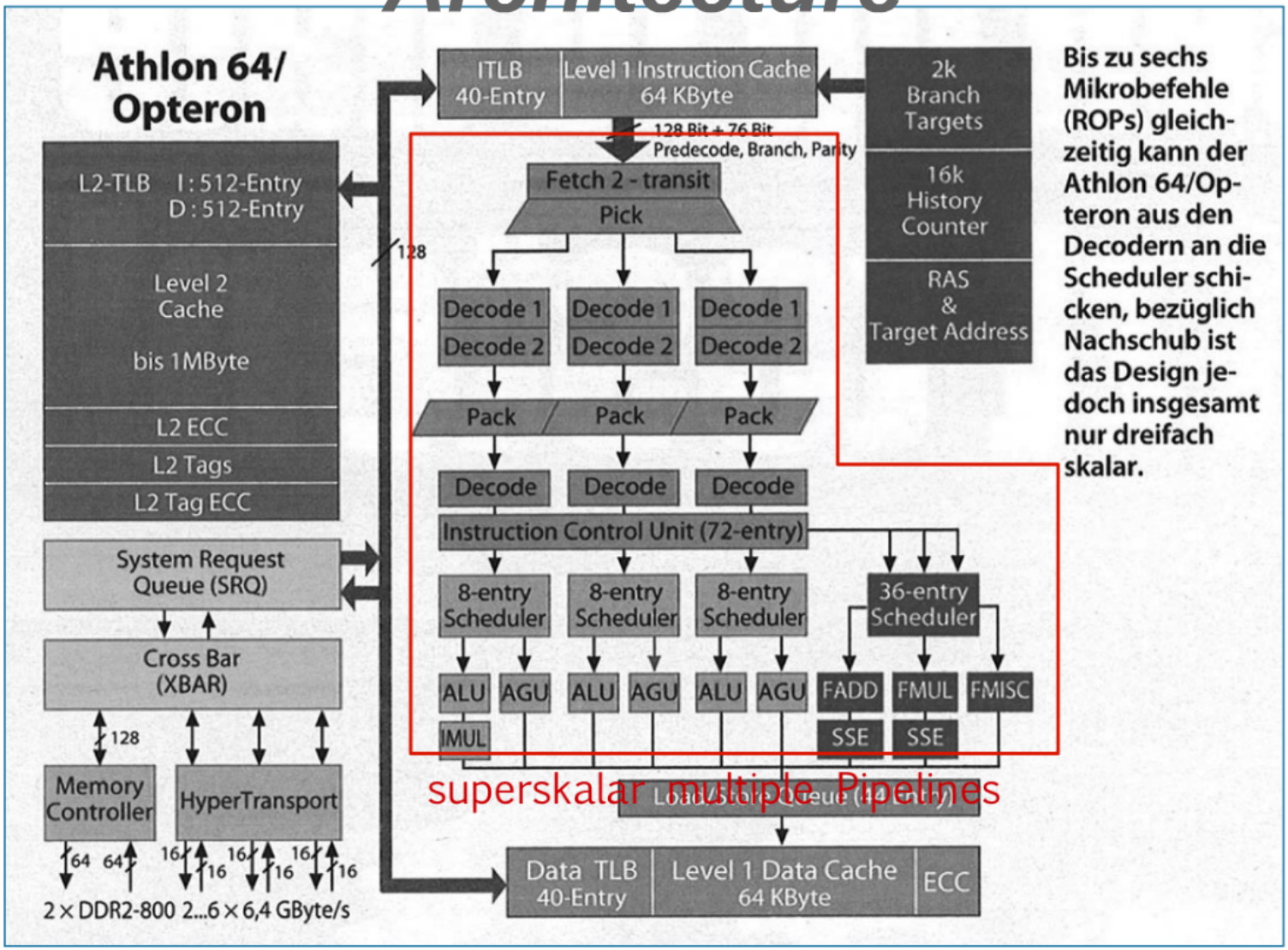


Intel Core Architecture



c't

AMD Athlon/Opteron Architecture



c't

Multi-processor Architectures

A classification according to Flynn (1972)

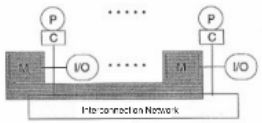
Very coarse, qualitative categorization of multiplicity of instruction and data streams

Examples:
SISD: Monoprocessors, von-Neumann Architecture
MISD: ? (multithreading)
SIMD: Vector processors
MIMD: multi-processors, clusters, etc.

MIMD: Distributed Memory Multi-Computer

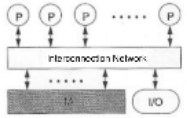
- Memory and periphery distributed
- No remote memory access
- Example: "Beowulf cluster"

MIMD: Virtual Shared Memory Multi-Processor



- Distributed memory accessible by all procs.
- Non-Uniform memory access (local access faster)
- CC-NUMA: Cache Coherent Non-Uniform Memory Access

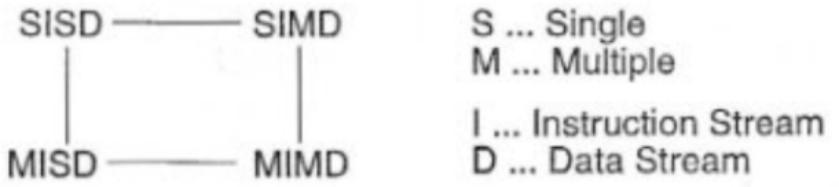
MIMD: Shared Memory Multi-Processor



- Memory and periphery accessible by all procs.
- Uniform Memory Access (UMA)
- Network: Bus or Crossbar

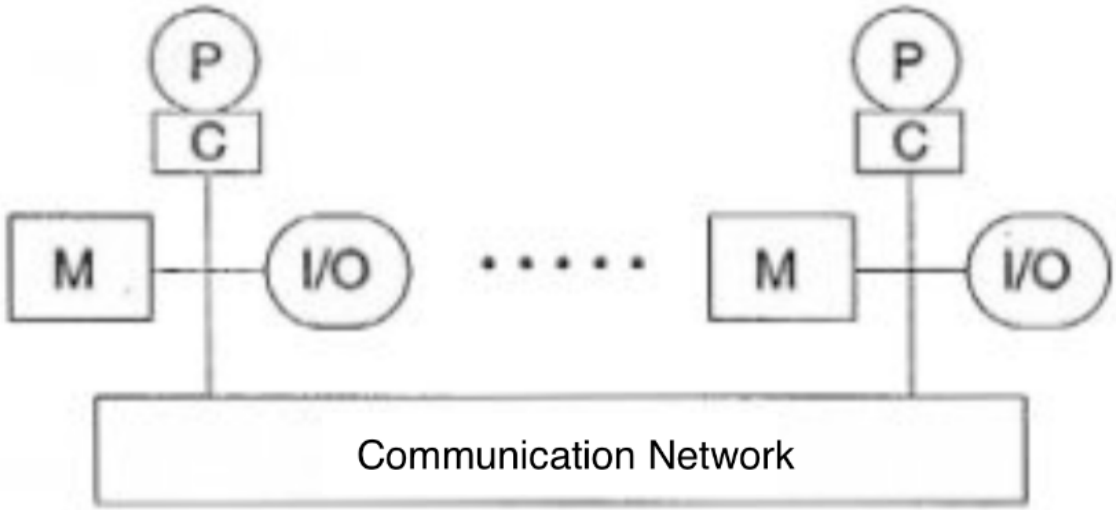
A classification according to Flynn (1972)

Very coarse, qualitative categorization of multiplicity of instruction and data streams



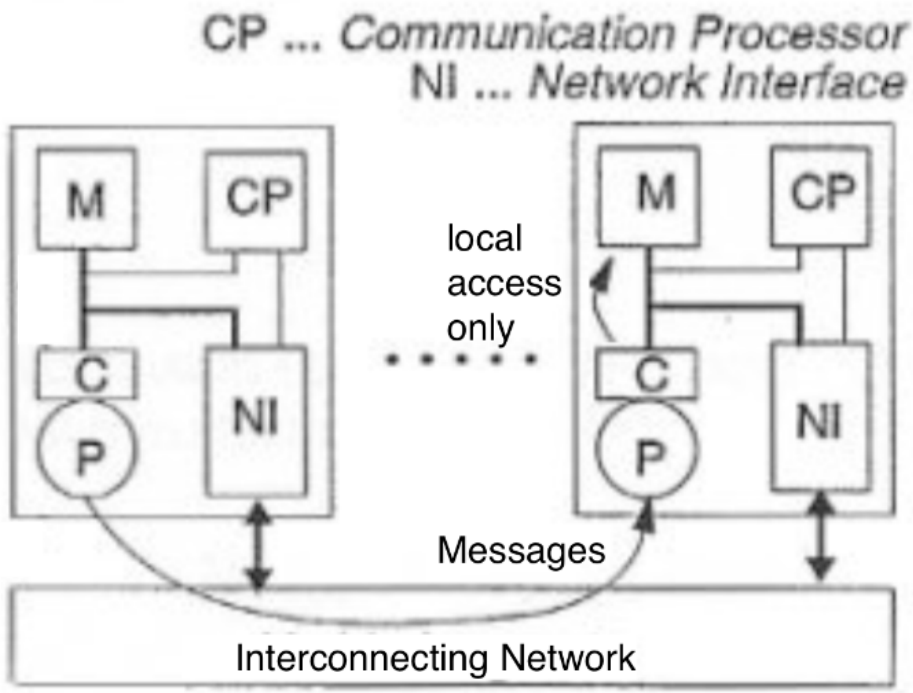
Examples:
SISD: Monoprocessors, von-Neumann Architecture
MISD: ? (multithreading)
SIMD: Vector processors
MIMD: multi-processors, clusters, etc.

MIMD: Distributed Memory Multi-Computer



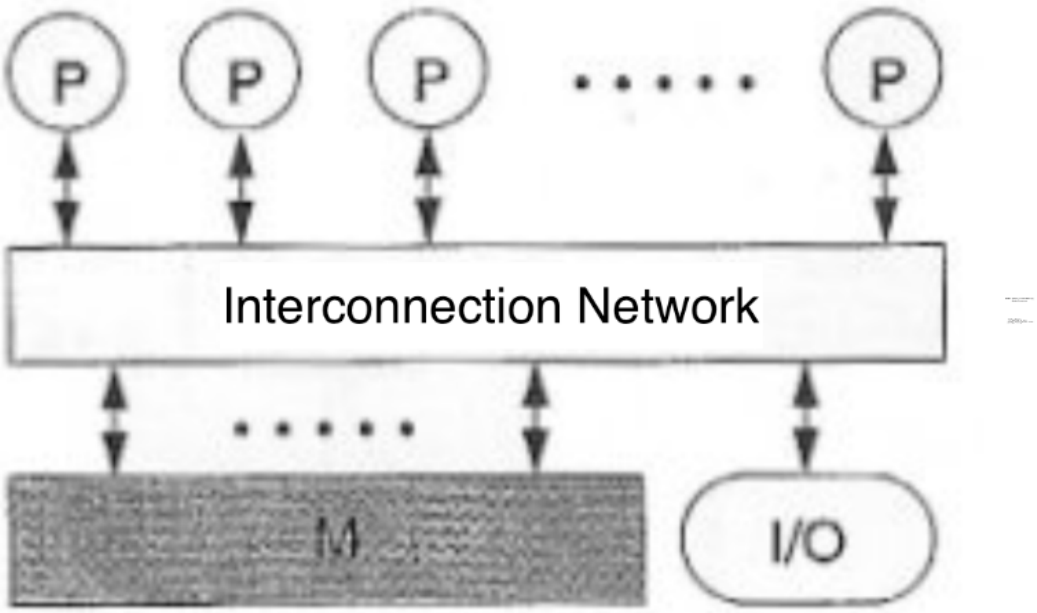
- Memory and periphery distributed
- No remote memory access
- Example: "Beowulf cluster"

MIMD: Distributed Memory Multi-Computer



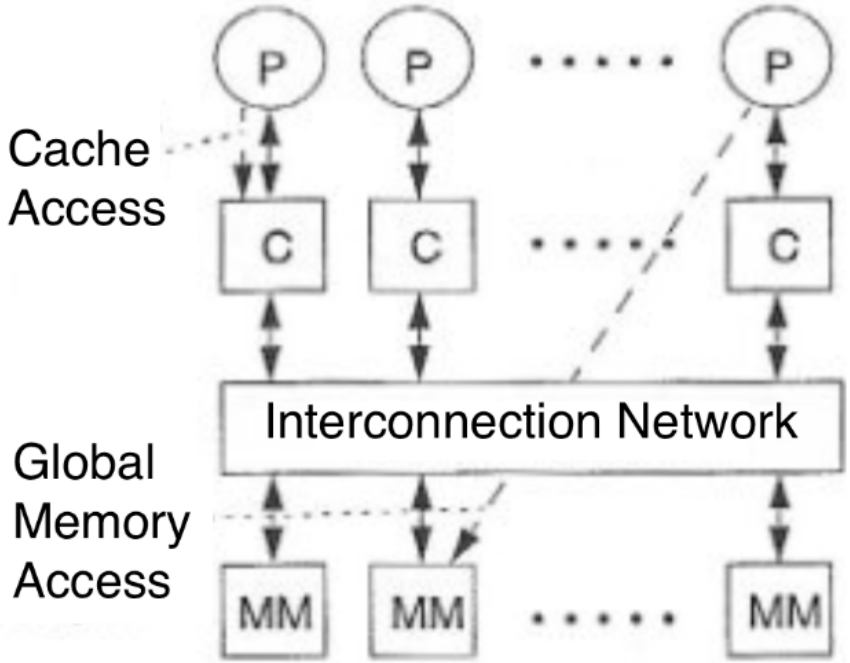
- Higher Bandwidth
- Low latency to local memory
- Dedicated message passing
- Custom Design

MIMD: Shared Memory Multi-Processor



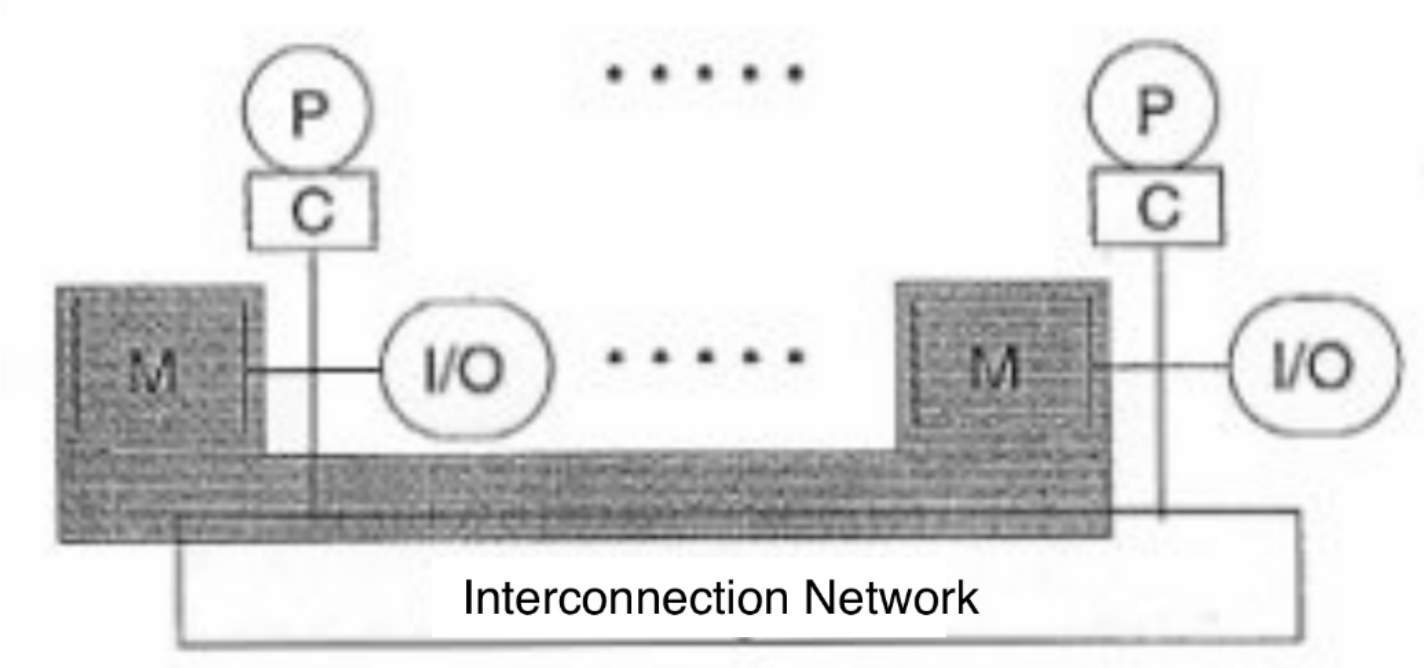
- Memory and periphery accessible by all procs.
- Uniform Memory Access (UMA)
- Network: Bus or Crossbar

MIMD: NUMA Shared Memory Multi-Processor



- Memory hierarchy (caches)
- Non-Uniform Memory Access (NUMA)
- Less Network traffic, faster (local) memory access
- Cache coherency required!

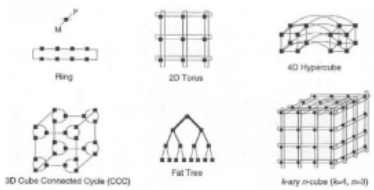
MIMD: Virtual Shared Memory Multi-Processor



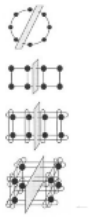
- Distributed memory accessible by all procs.
- Non-Uniform memory access (local access faster)
- CC-NUMA: Cache Coherent Non-Uniform Memory Access

Network Architectures

Topologies of static networks



Determining the performance of a network: bisection bandwidth



Ring:

$$B_{\text{Ring}} = 2$$

2D Mesh of size (nxm):

$$B_{\text{2D mesh}} = \min(n, m)$$

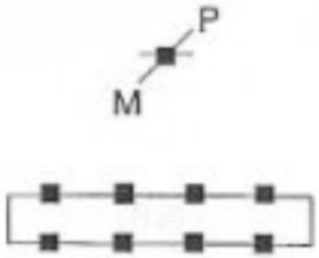
2D Torus of size (nxm):

$$B_{\text{2D torus}} = 2 \cdot \min(n, m)$$

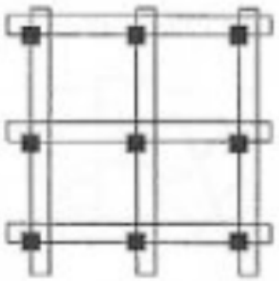
3D Torus of size (lxnxm):

$$B_{\text{3D torus}} = 2 \cdot [l \cdot n + n \cdot m - \max(l, n, m) - \min(l, n, m)] \cdot \min(l, n, m)$$

Topologies of static networks

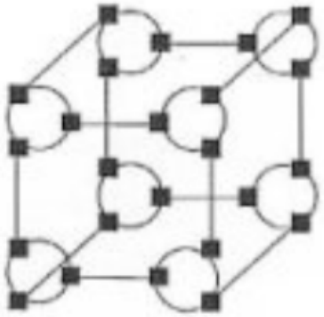


Ring



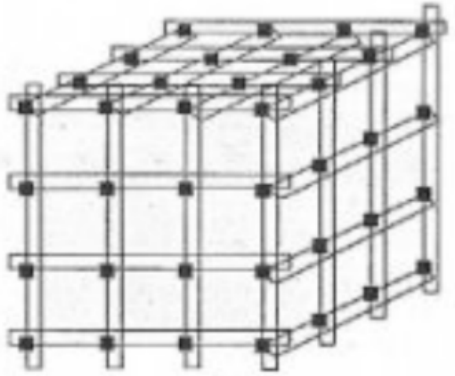
2D Torus

4D Hypercube



3D Cube Connected Cycle (CCC)

Fat Tree

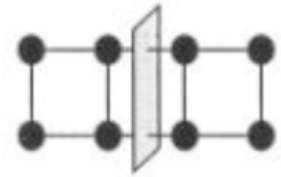


k-ary *n*-cube (*k*=4, *n*=3)

Determining the performance of a network: bisection bandwidth

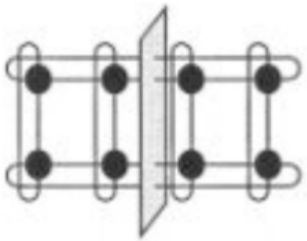
Ring:

$$B_{\text{ring}} = 2$$



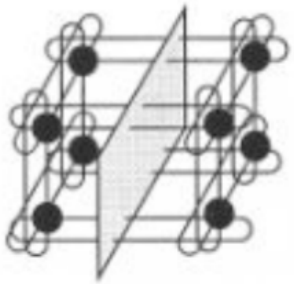
2D Mesh of size (nxm):

$$B_{\text{2D mesh}} = \min(m, n)$$



2D Torus of size (nxm):

$$B_{\text{2D torus}} = 2 \cdot \min(m, n)$$



3D Torus of size (lxnxm):

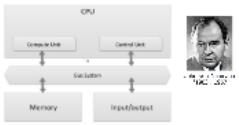
$$B_{\text{3D torus}} = 2 \cdot [l + m + n - \max(l, m, n) - \min(l, m, n)] \cdot \min(l, m, n)$$

Network Topology Characteristics

Topology	# Nodes	Degree (# ports)	min. path	bisection width
Ring	N	2	$\frac{N}{2}$	2
2D Torus	$N = k^2$	4	$2\frac{k}{2}$	$2k$
Hypercube (d-dim.)	$N = 2^d$	d	d	$\frac{N}{2}$
Tree	N	3	$2(\log_2 N - 1)$	1
k-ary n-cube	$N = k^n$	$2n$	$n\frac{k}{2}$	$2k^{n-1}$

Basics

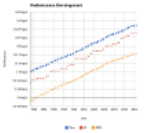
Von Neumann Architecture



Introduction

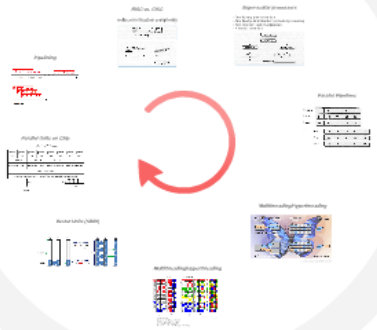
Why deal with Computer Architectures?

- Most programs achieve 10-20% of peak performance, why?
- How does a computer work, anyways?
- What makes the performance?



Acceleration Features

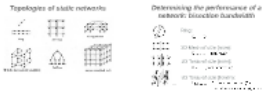
Parallelism on the processor level



Computer Architectures

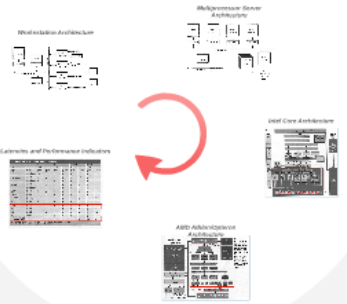
Scientific Computing
Jörn Behrens

Network Architectures



System Architectures

Replicating functional units



Multi-processor Architectures