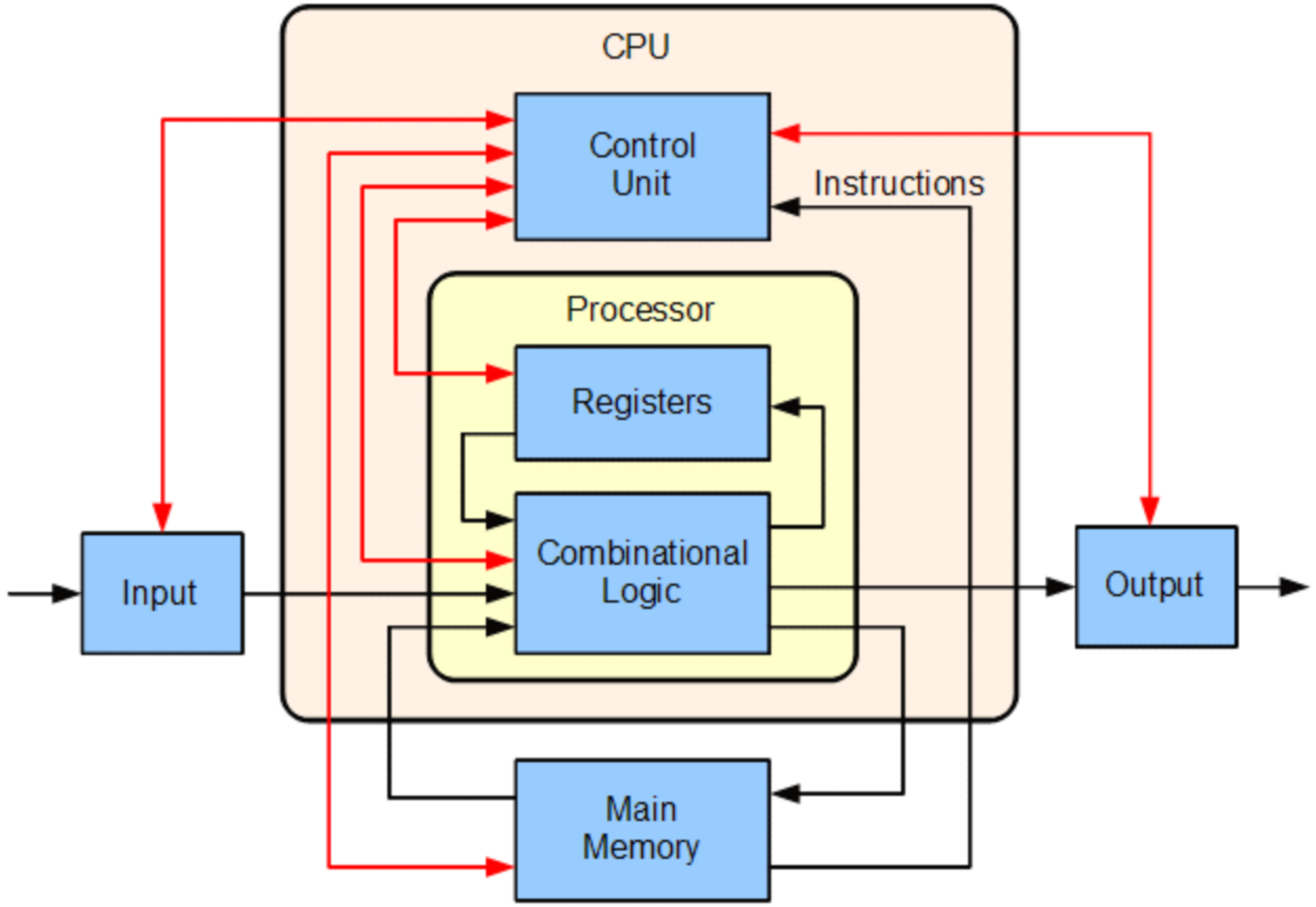
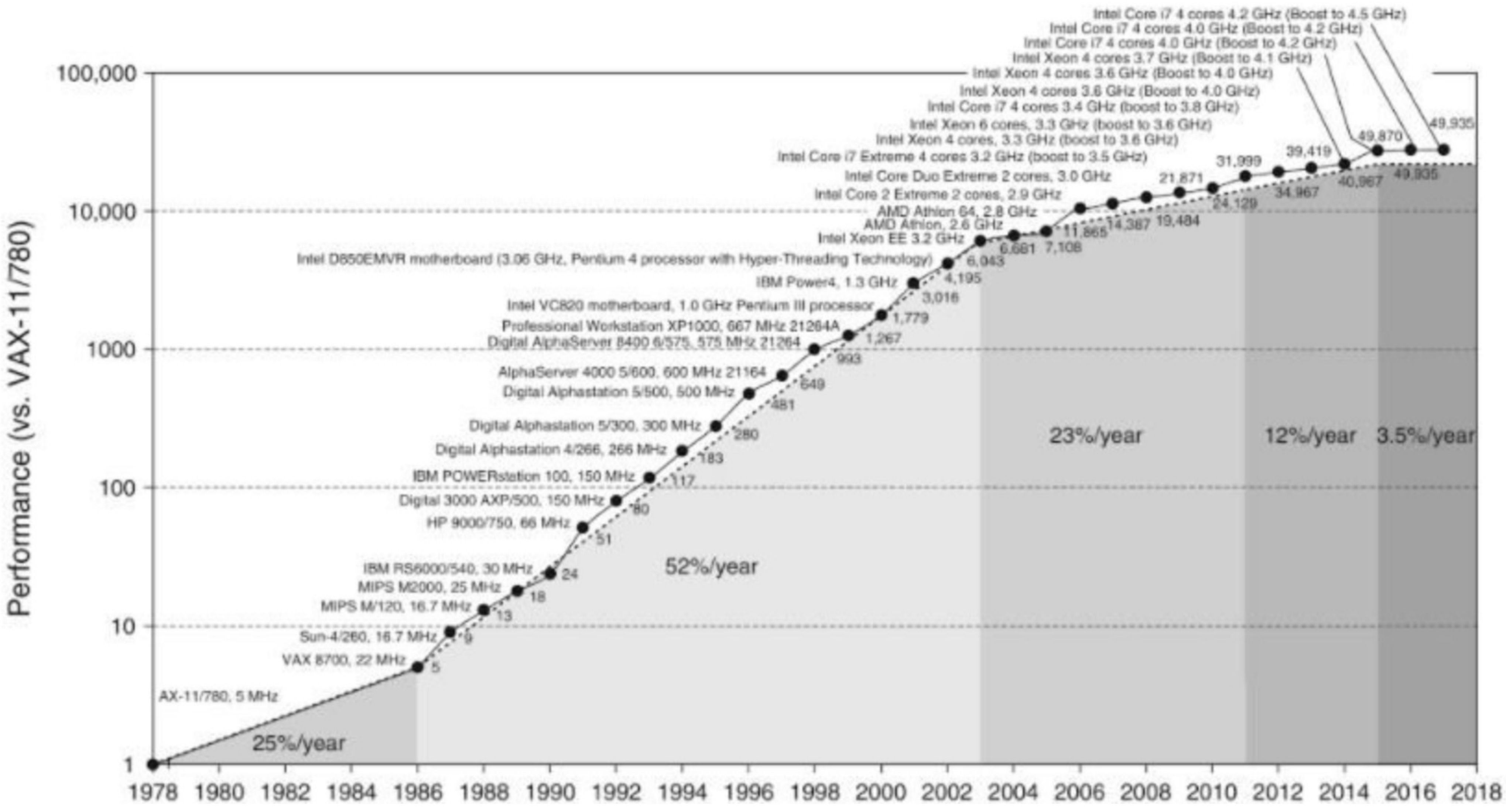
Chapter 2

CPU Architecture

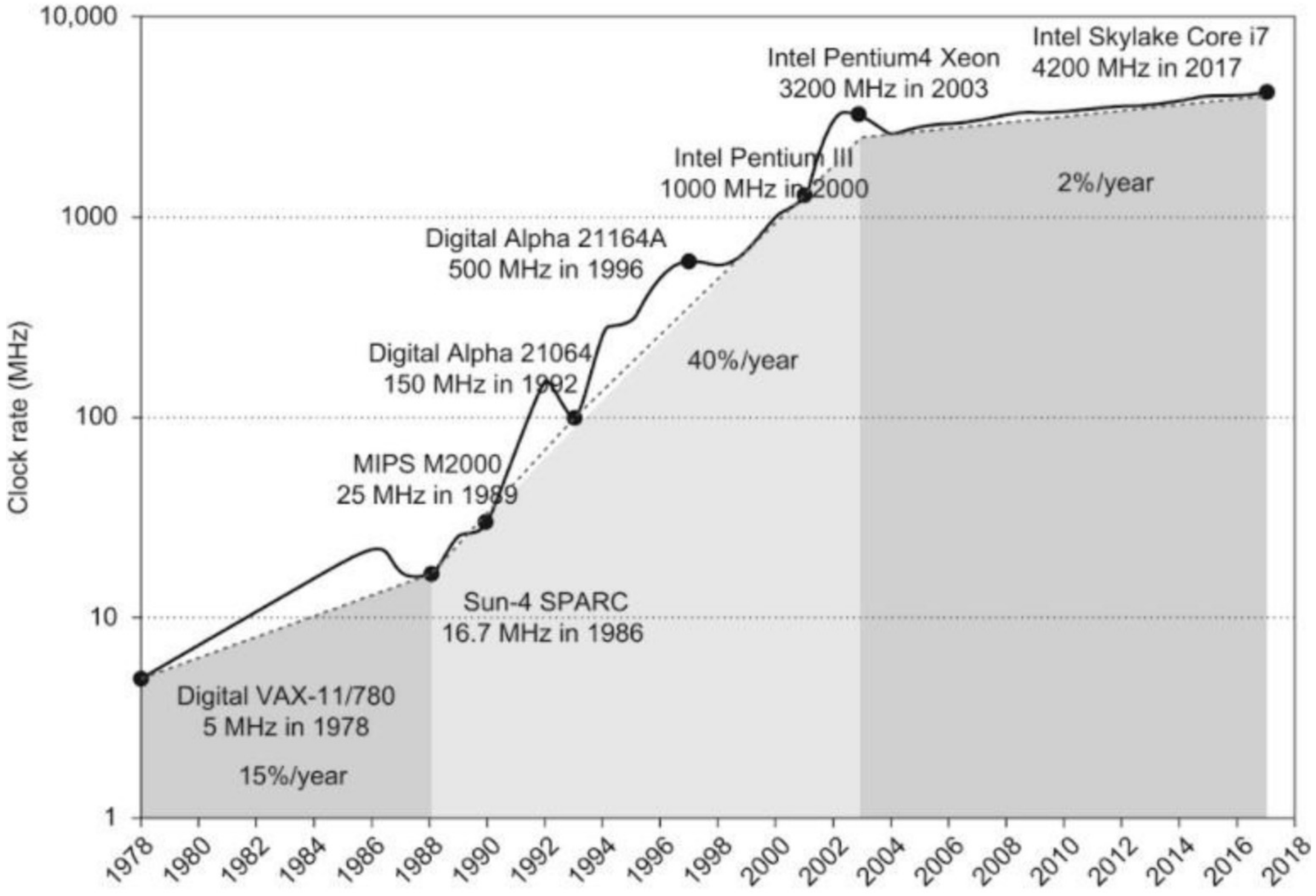
A Basic Uniprocessor CPU Computer



Processor Performance

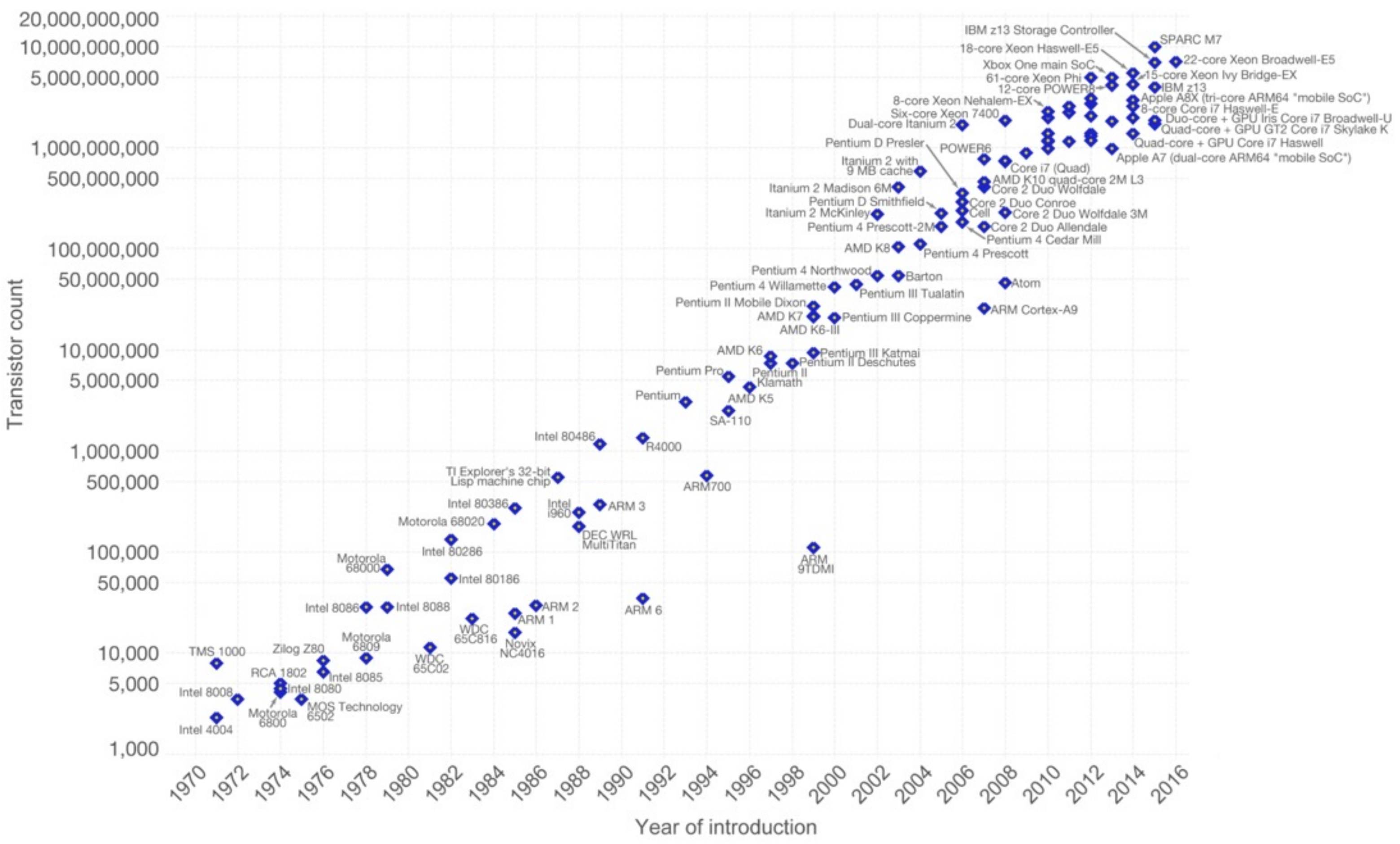


Clock Rate of Microprocessors



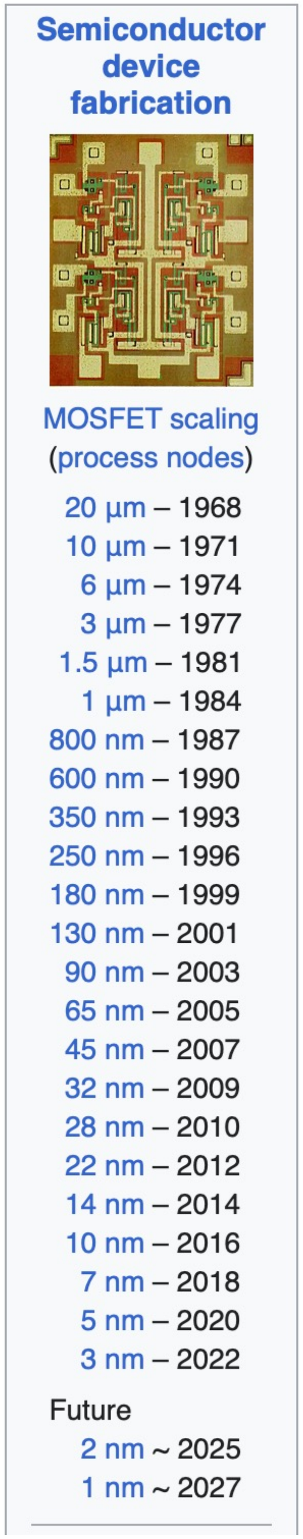
Moore’s Law

- **An observation made by Intel co-founder Gordon Moore in 1965:**
  - The number of transistors per square inch on integrated circuits had doubled every 18 months



Roadmap for Semiconductors

- **Key issue:**
  - **How to use the newly available transistors?**
  - **<u>The whole class is about this question</u>**



Semiconductor device fabrication

MOSFET scaling (process nodes)

20 µm	– 1968
10 µm	– 1971
6 µm	– 1974
3 µm	– 1977
1.5 µm	– 1981
1 µm	– 1984
800 nm	– 1987
600 nm	– 1990
350 nm	– 1993
250 nm	– 1996
180 nm	– 1999
130 nm	– 2001
90 nm	– 2003
65 nm	– 2005
45 nm	– 2007
32 nm	– 2009
28 nm	– 2010
22 nm	– 2012
14 nm	– 2014
10 nm	– 2016
7 nm	– 2018
5 nm	– 2020
3 nm	– 2022
Future	
2 nm	~ 2025
1 nm	~ 2027

Performance: Parallelism

- **Instruction-Level Parallelism (ILP)**
  - Implicit/transparent to users/programmers
  - Instruction pipelining
  - Superscalar execution
  - Out-of-order execution
  - Register renaming
  - Speculative execution
  - Branch prediction
- **Task-Level Parallelism (TLP)**
  - Explicit to users/programmers
  - Multiple threads or processes executed simultaneously
  - Multi-core processors
- **Data Parallelism**
  - Vector processors and SIMD

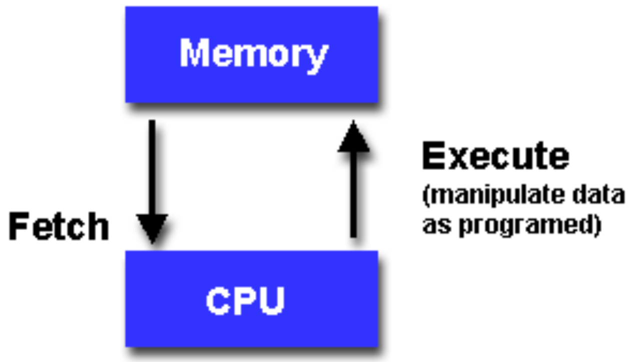
CPU Computing

CPU performance is the product of many related advances

- **Increased transistor density**
- **Increased transistor performance**
- **Wider data paths**
- **Pipelining**
- **Superscalar execution**
- **Speculative execution**
- **Caching**
- **Chip- and system-level integration**

von Neumann Architecture

- **Computer model : von Neumann computer (Named after the Hungarian mathematician John von Neumann)**
- **A von Neumann computer uses the stored-program concept**
  - The CPU executes a stored program that specifies a sequence of read and write operations on the memory



- **Three components: processor, memory, and datapath**
  - Bottlenecks
  - Solution: multiplicity (implicit parallelism)

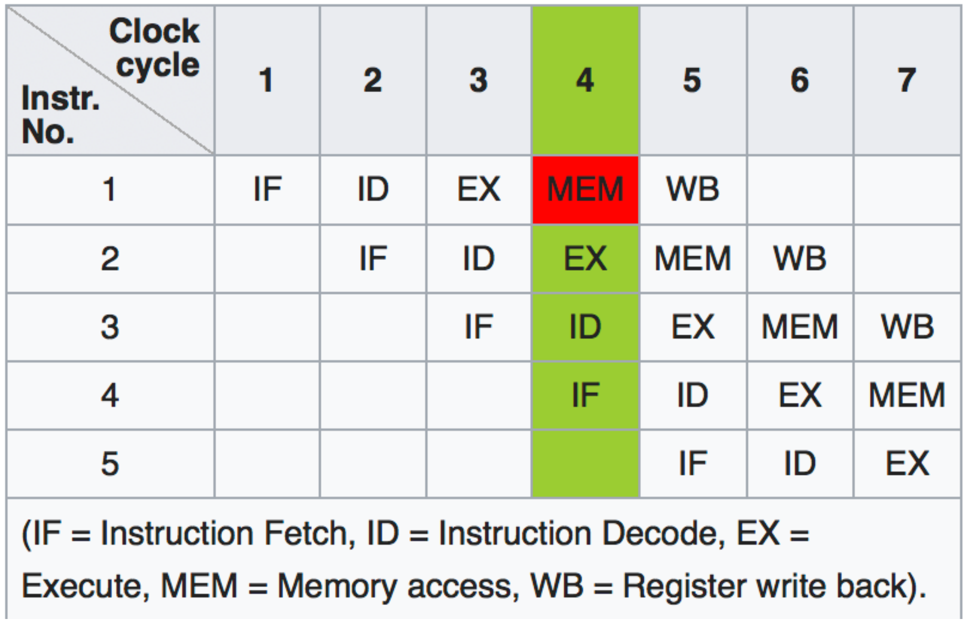
Bandwidth: Gravity of Modern Computer Systems

- **The Bandwidth between key components ultimately dictates system performance**
  - Especially true for massively parallel systems processing massive amount of data
  - Tricks like buffering, reordering, caching can temporarily defy the rules in some cases
  - Ultimately, the performance goes falls back to what the “speeds and feeds” dictate
    - Perform well themselves
    - Cooperate well too

ILP: Instruction Pipelining

- Pipelining attempts to keep every part of the processor busy with some instruction by dividing incoming instructions into a series of sequential steps performed by different processor units with different parts of instructions processed in parallel
- It allows faster CPU throughput than would otherwise be possible at a given clock rate, but may increase latency due to the added overhead of the pipelining process itself

Basic five-stage pipeline

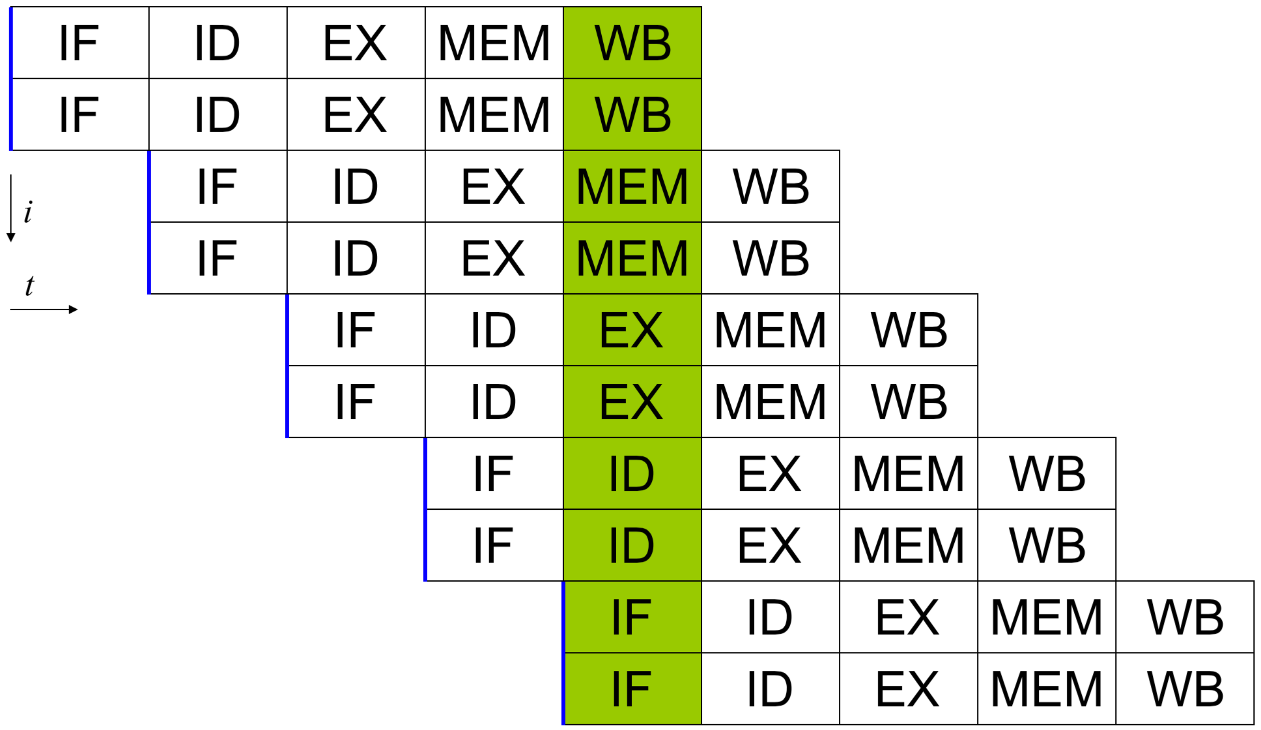


Clock cycle Instr. No.	1	2	3	4	5	6	7
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX

(IF = Instruction Fetch, ID = Instruction Decode, EX = Execute, MEM = Memory access, WB = Register write back).

ILP: Superscalar Execution

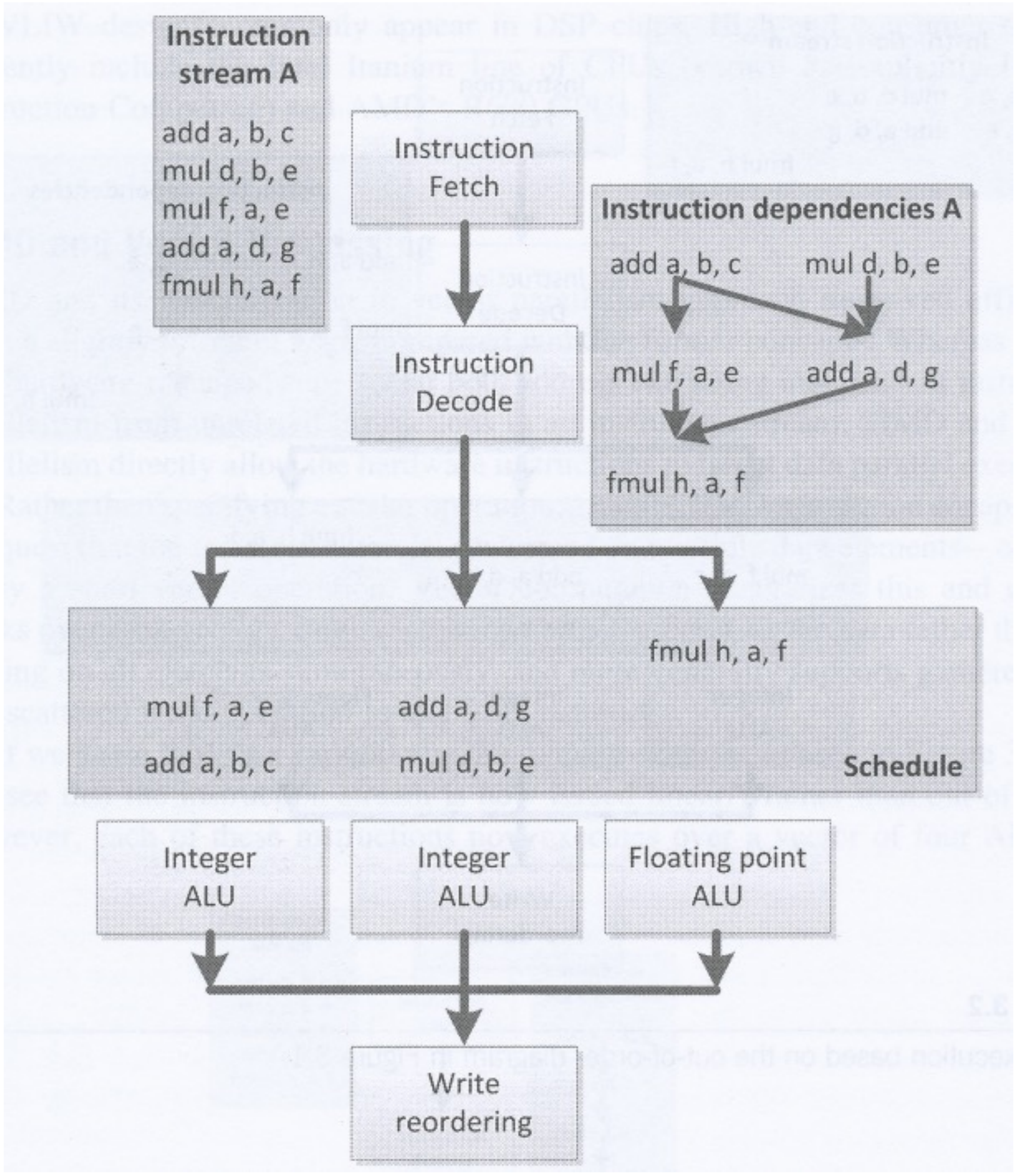
- **A superscalar processor can execute more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to different execution units on the processor**
  - Each execution unit is not a separate processor, but an execution resource within a single CPU such as an arithmetic logic unit



Superscalar Execution

- **Superscalar and, by extension, out-of-order execution is one solution that has been included on CPUs for a long time**
  - Out-of-order scheduling logic requires a substantial area of the CPU die to maintain dependence information and queues of instructions to deal with dynamic schedules throughout the hardware
  - Speculative instruction execution necessary to expand the window of out-of-order instructions to execute in parallel results in inefficient execution of throwaway work

Out-of-order Execution



ILP: Very Long Instruction Word (VLIW)

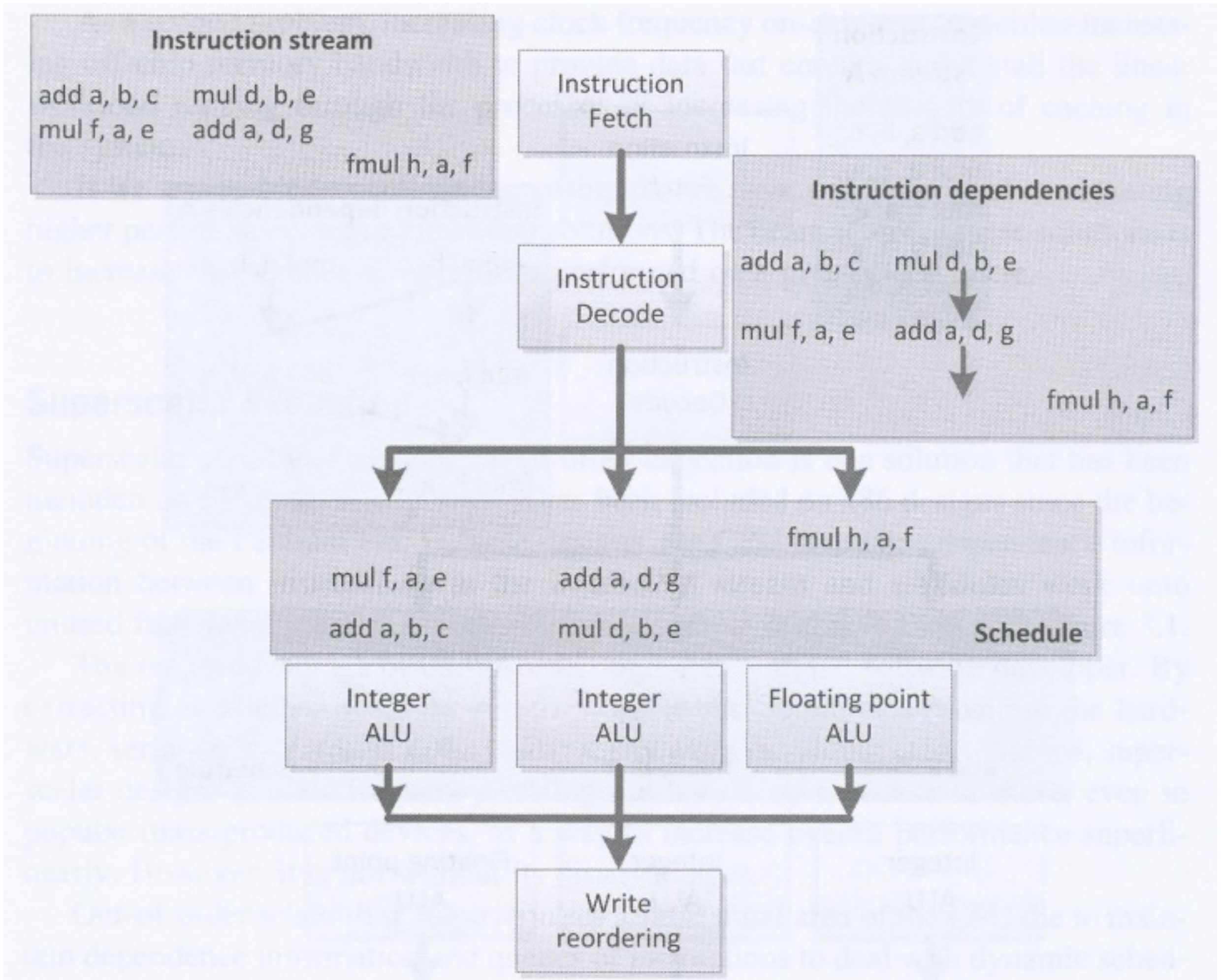
- **A VLIW processor allows programs to explicitly specify instructions to execute at the same time, concurrently, in parallel**
  - Instructions are scheduled by the compiler
  - A fixed number of operations are formatted as one big instruction (called a bundle)
  - Goal: Reduce hardware complexity
    - Less multiple-issue hardware
    - Simpler instruction dispatch
    - No structural hazard checking logic
  - Compiler figures all this out

Example Instruction format (5-issue):

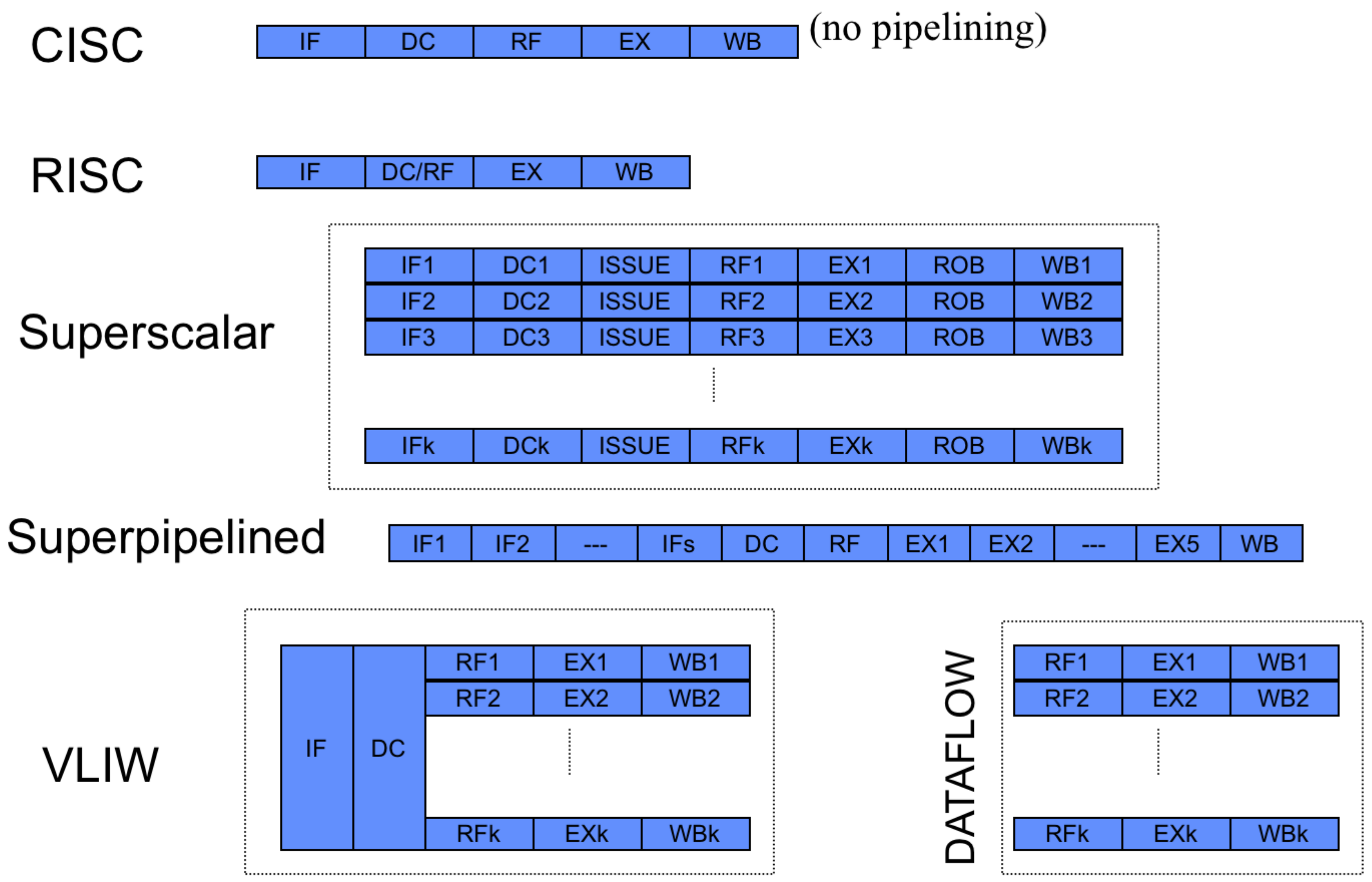
operation 1	operation 2	operation 3	operation 4	operation 5

VLIW

- **VLIW is a heavily compiler-dependent method for increasing instruction-level parallelism in a processor**
  - VLIW moves the dependence analysis work into the compiler



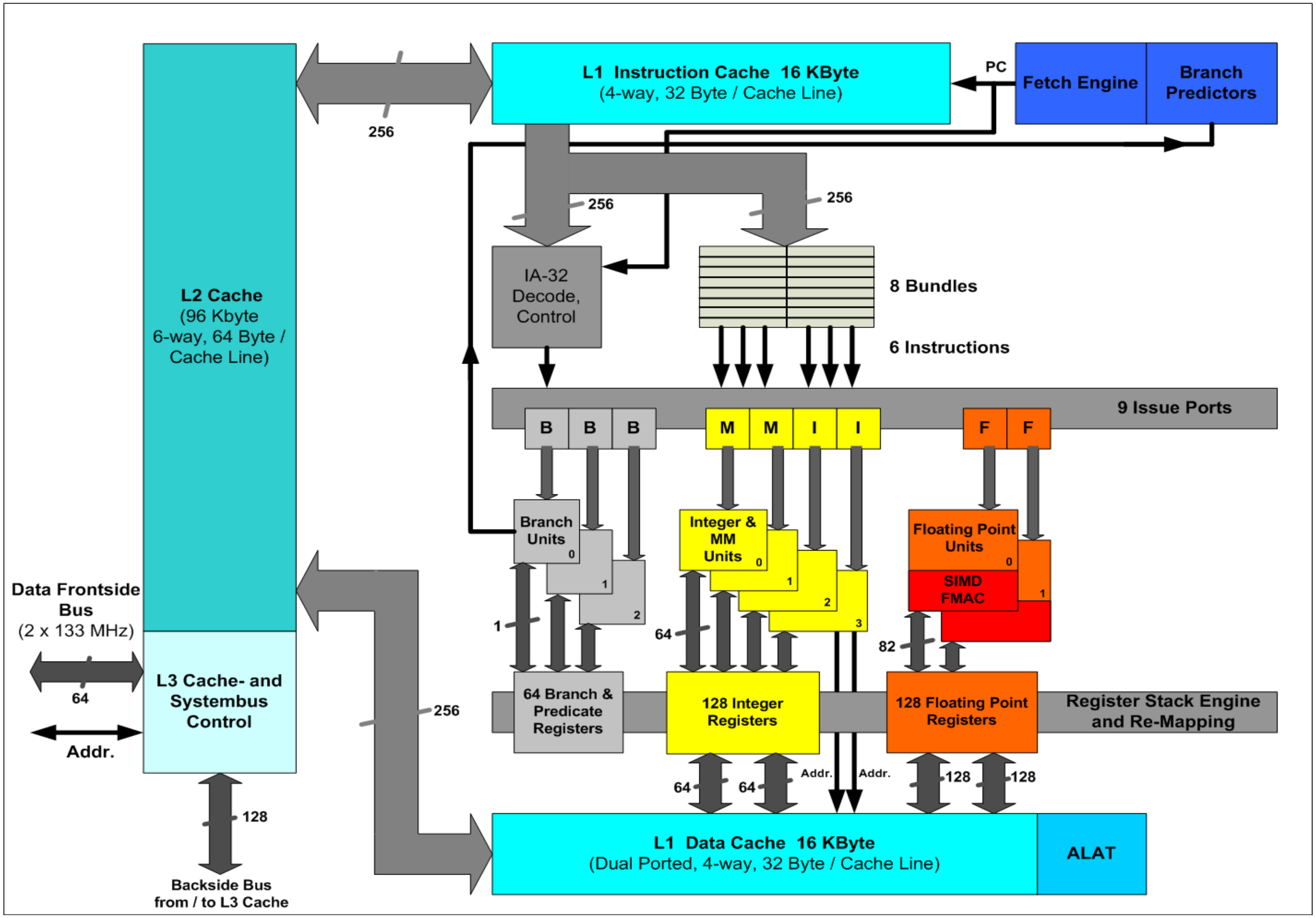
Instruction Pipeline Overview



ILP: Explicitly Parallel Instruction Computing (EPIC)

- **EPIC architecture evolved from VLIW architecture**
- **Move the complexity of instruction scheduling from the CPU hardware to the software compiler**
- **Multiple operations are encoded in every instruction, and then processed by multiple execution units**
- **Moving beyond VLIW:**
  - Each group of multiple software instructions is called a *bundle*
    - Each of the bundles has a stop bit indicating if this set of operations is depended upon by the subsequent bundle
  - A software prefetch instruction is used as a type of data prefetch
  - A speculative load instruction is used to speculatively load data
  - A check load instruction aids speculative loads by checking whether a speculative load was dependent on a later store, and thus must be reloaded

Intel Itanium Organization


L1 Instruction Cache 16 KByte
(4-way, 32 Byte / Cache Line)
PC
Fetch Engine
Branch Predictors
256
256
256
L2 Cache
(96 Kbyte
6-way, 64 Byte /
Cache Line)
IA-32
Decode,
Control
8 Bundles
6 Instructions
9 Issue Ports
B
B
B
M
M
I
I
F
F
Branch
Units
0
1
2
Integer &
MM
Units
0
1
2
3
Floating Point
Units
0
SIMD
FMAC
1
Data Frontside
Bus
(2 x 133 MHz)
64
Addr.
L3 Cache- and
Systembus
Control
1
64
82
256
64 Branch &
Predicate
Registers
128 Integer
Registers
128 Floating Point
Registers
Register Stack Engine
and Re-Mapping
64
64
Addr.
Addr.
128
128
128
L1 Data Cache 16 KByte
(Dual Ported, 4-way, 32 Byte / Cache Line)
ALAT
Backside Bus
from / to L3 Cache