Computer Architecture

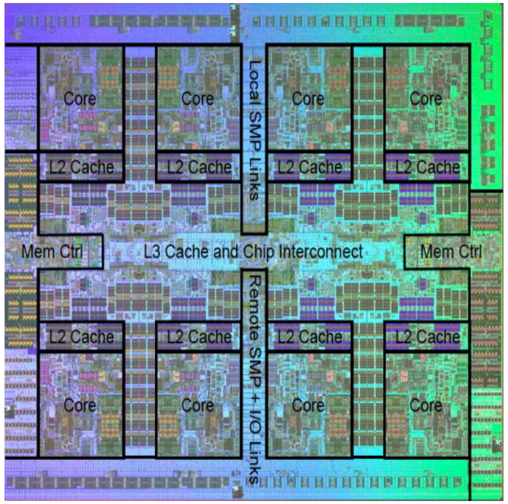
Per Stenström

Agenda

- Historical perspective and driving forces
- What is computer architecture?
- Parallelism in instruction execution
- Trends moving forward
- Course offerings

Evolution of computers: A 65-year perspective

IBM Power 7 (2010) ~ 1 billion additions/sec



Human imagination

Improved technologies

Innovative design principles

~ 10 m

~ 1 cm

~ 1 million times faster,
smaller, and more power-efficient

ENIAC (1946) ~ 1000 additions/sec

Moore’s Law

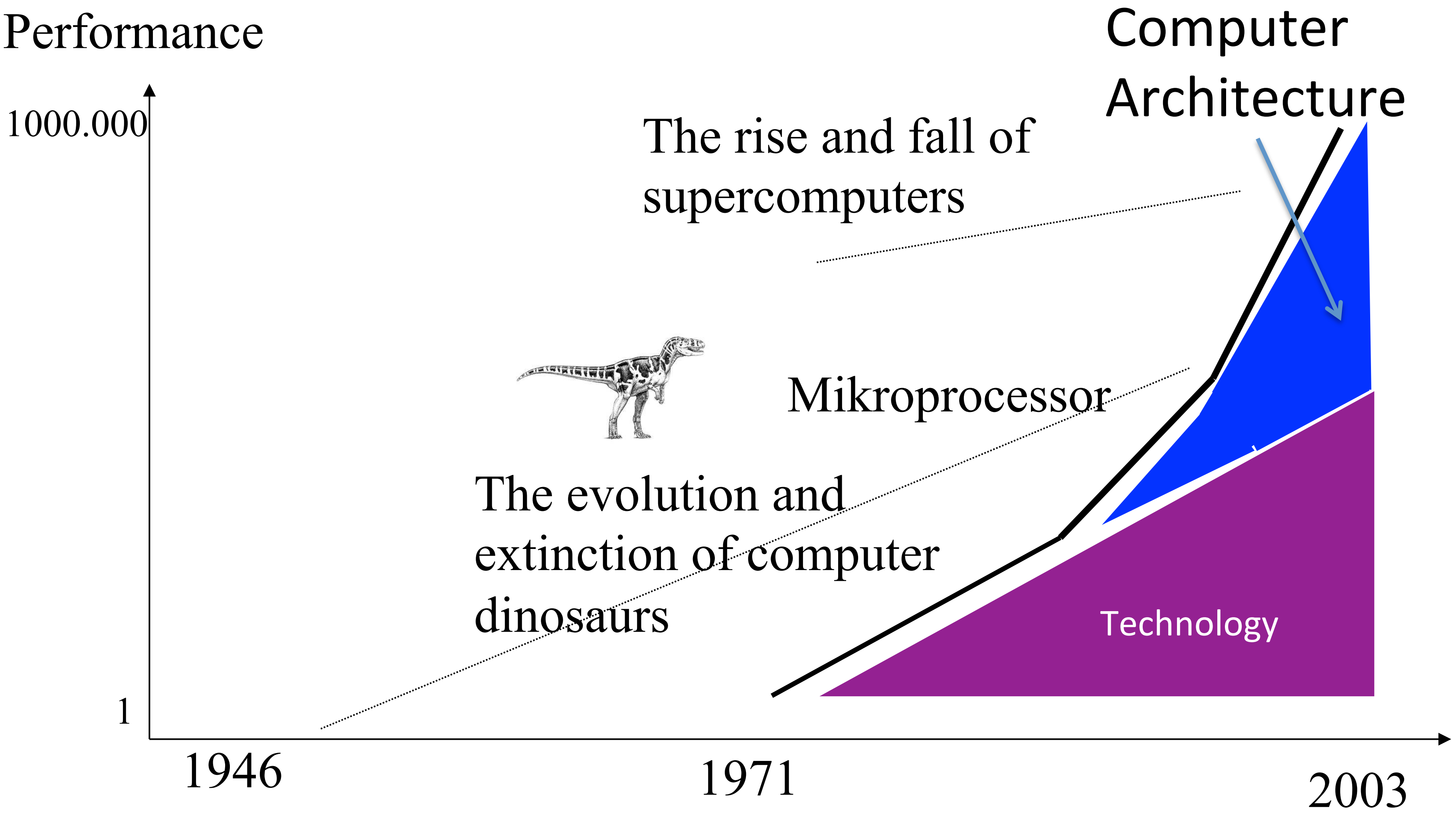
In 1965 Gordon Moore, Intel:

- Number of transistors on a die will double biannually

Miniaturization led to a 35% annual improvement of clock speed.

Moore’s Law has come to dictate performance goals in the computing industry

The Killer Microprocessor



Computer Architecture

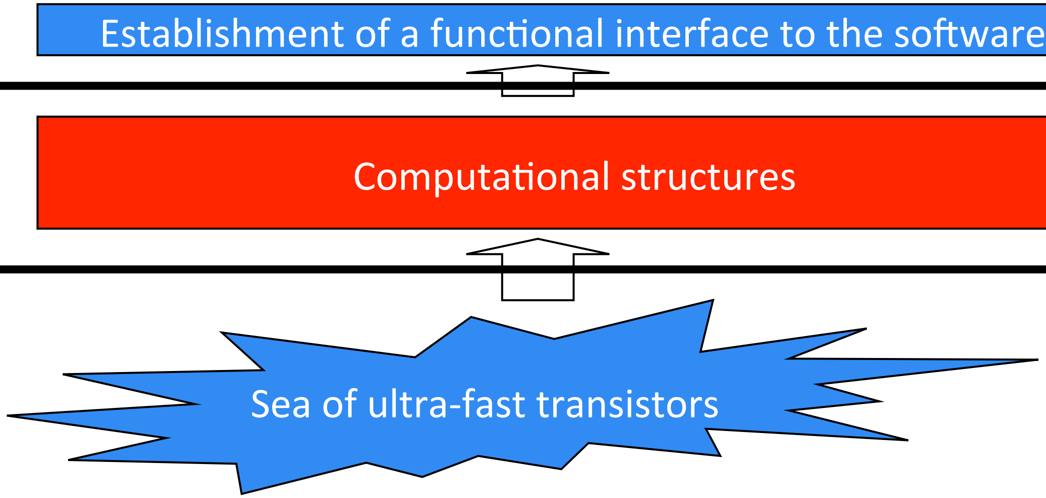
The rise and fall of supercomputers

Mikroprocessor

The evolution and extinction of computer dinosaurs

Computer Architecture

- The engineering discipline of computer design
- The hardware/software interface
  - Instruction Set Architecture (ISA)
  - Computer organization
  - Hardware design



Establishment of a functional interface to the software

Computational structures

Sea of ultra-fast transistors

Parallelism and locality

Software exhibits, in varying degrees:

- *Parallelism* – individual operations are independent and can be carried out in parallel
- *Locality* – different operations reuse earlier computed values

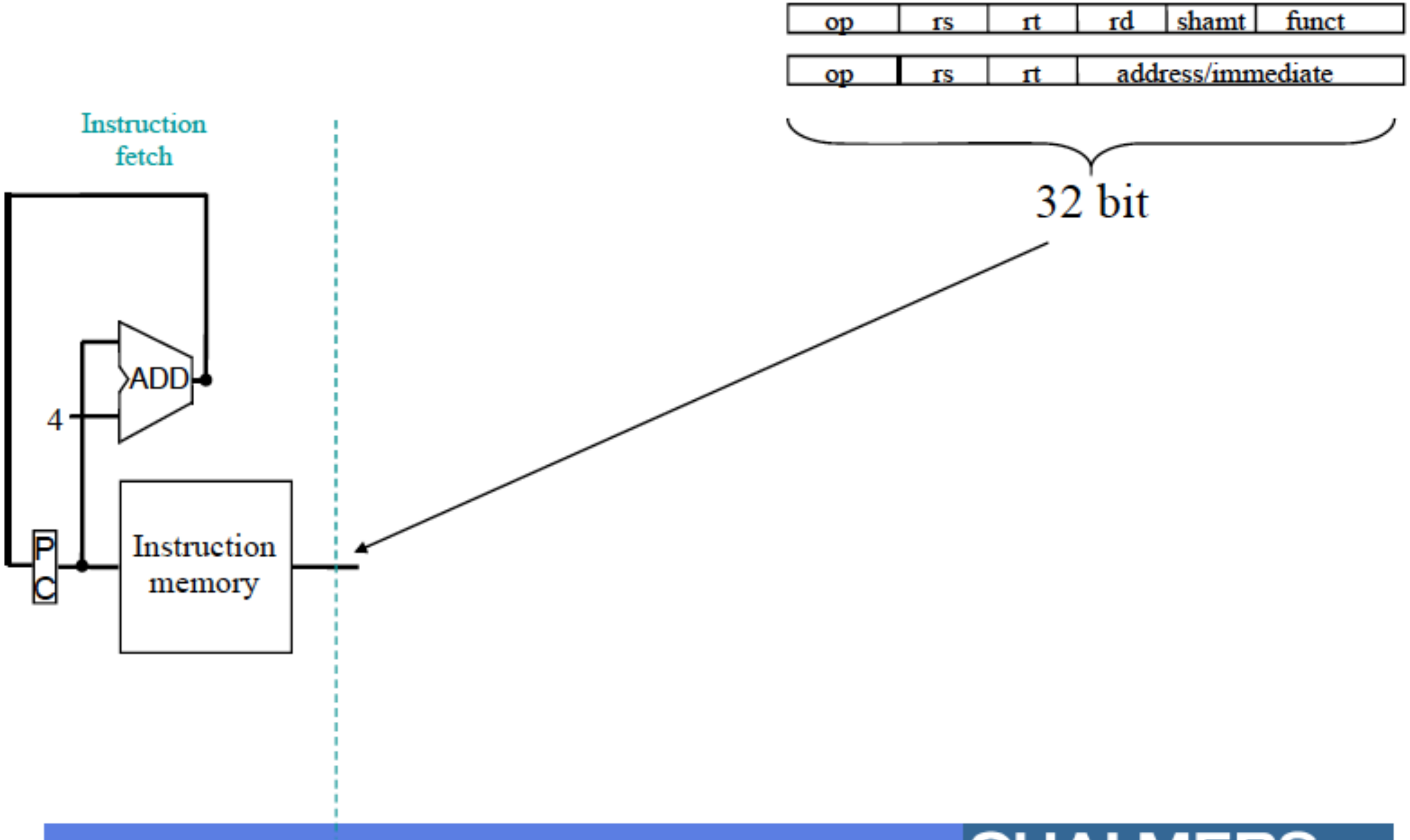
These fundamental properties have led to numerous innovations in **computer architecture**

Instruction Execution

For each instruction:

1. Instruction fetch (IF)
2. Instruction decode, operand fetch (ID)
3. Execute computations (EX)
4. Memory access (MEM)
5. Write back results to registers (WB)

Single Cycle Implementation



op	rs	rt	rd	shamt	funct

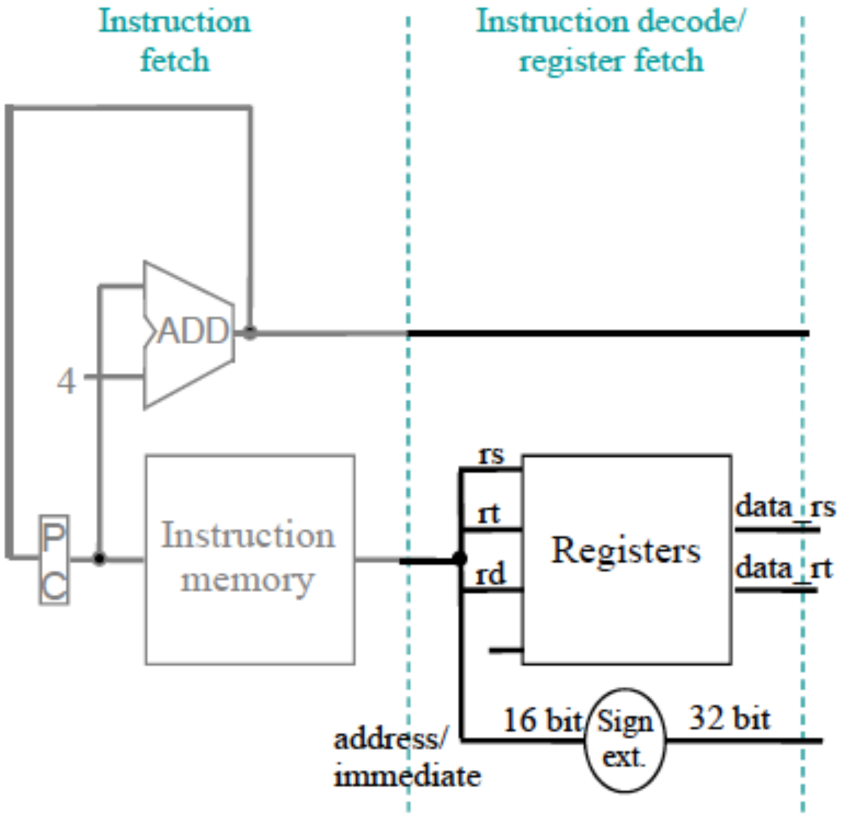
op	rs	rt	address/immediate

32 bit

Single Cycle Implementation

op	rs	rt	rd	shamt	funct

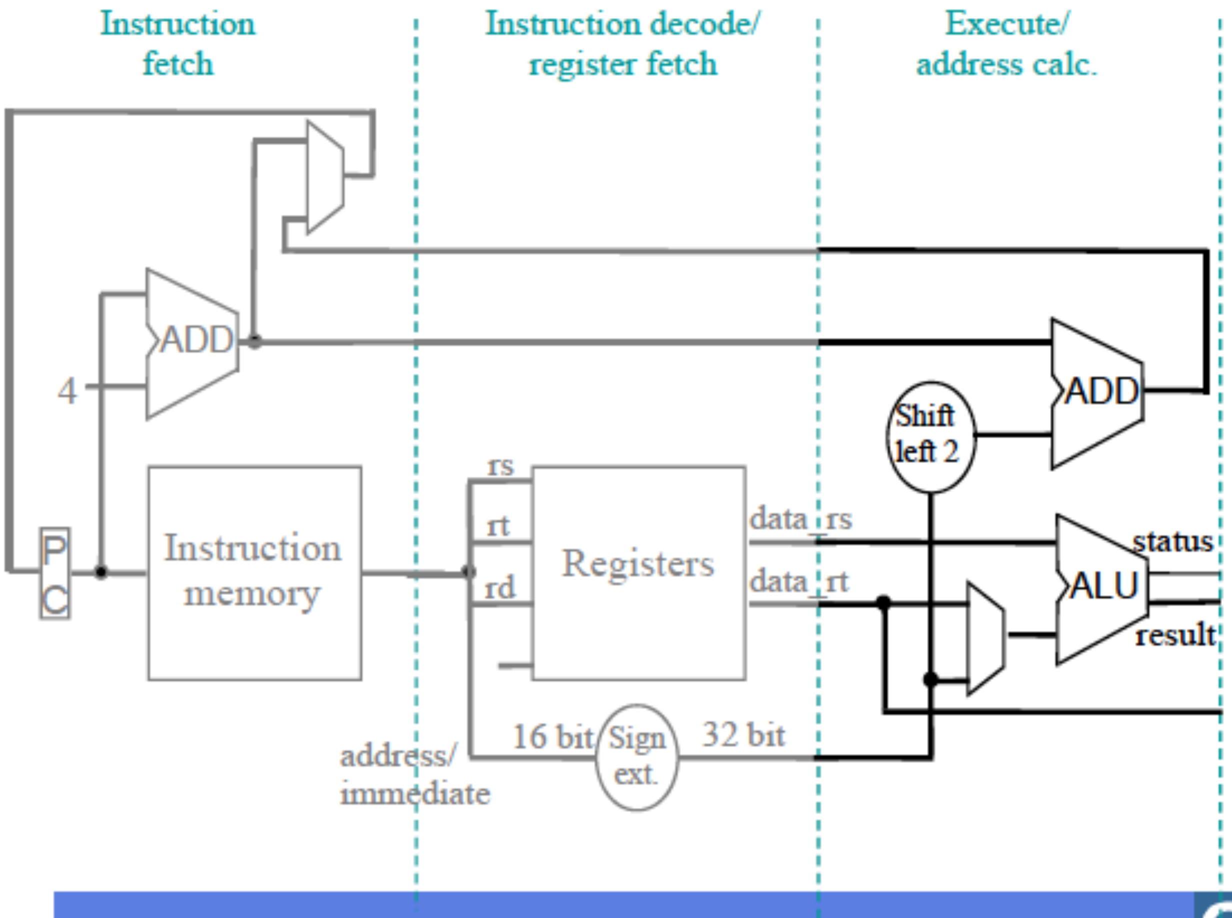
op	rs	rt	address/immediate



Single Cycle Implementation

op	rs	rt	rd	shamt	funct

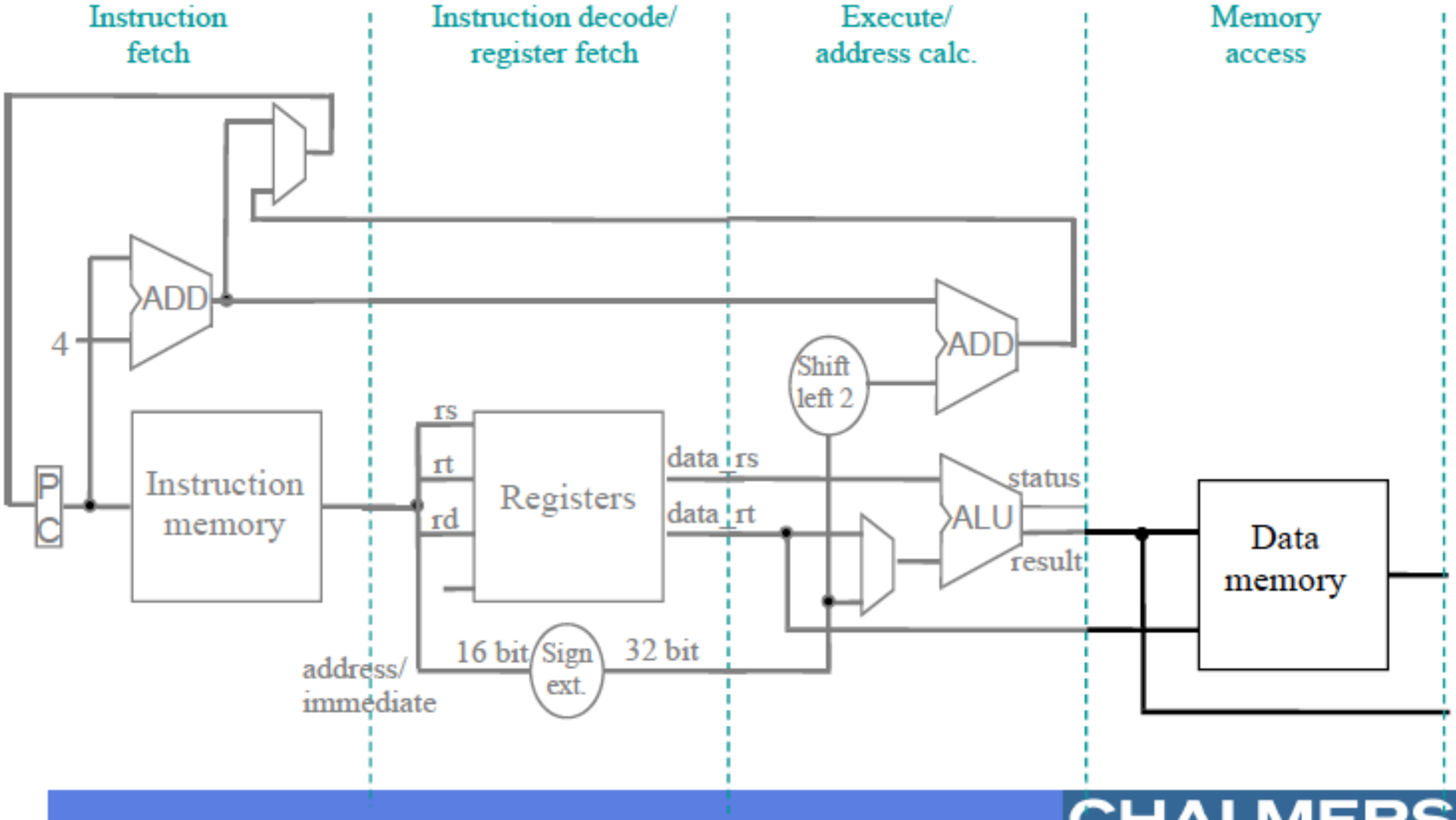
op	rs	rt	address/immediate



Single Cycle Implementation

op	rs	rt	rd	shamt	funct

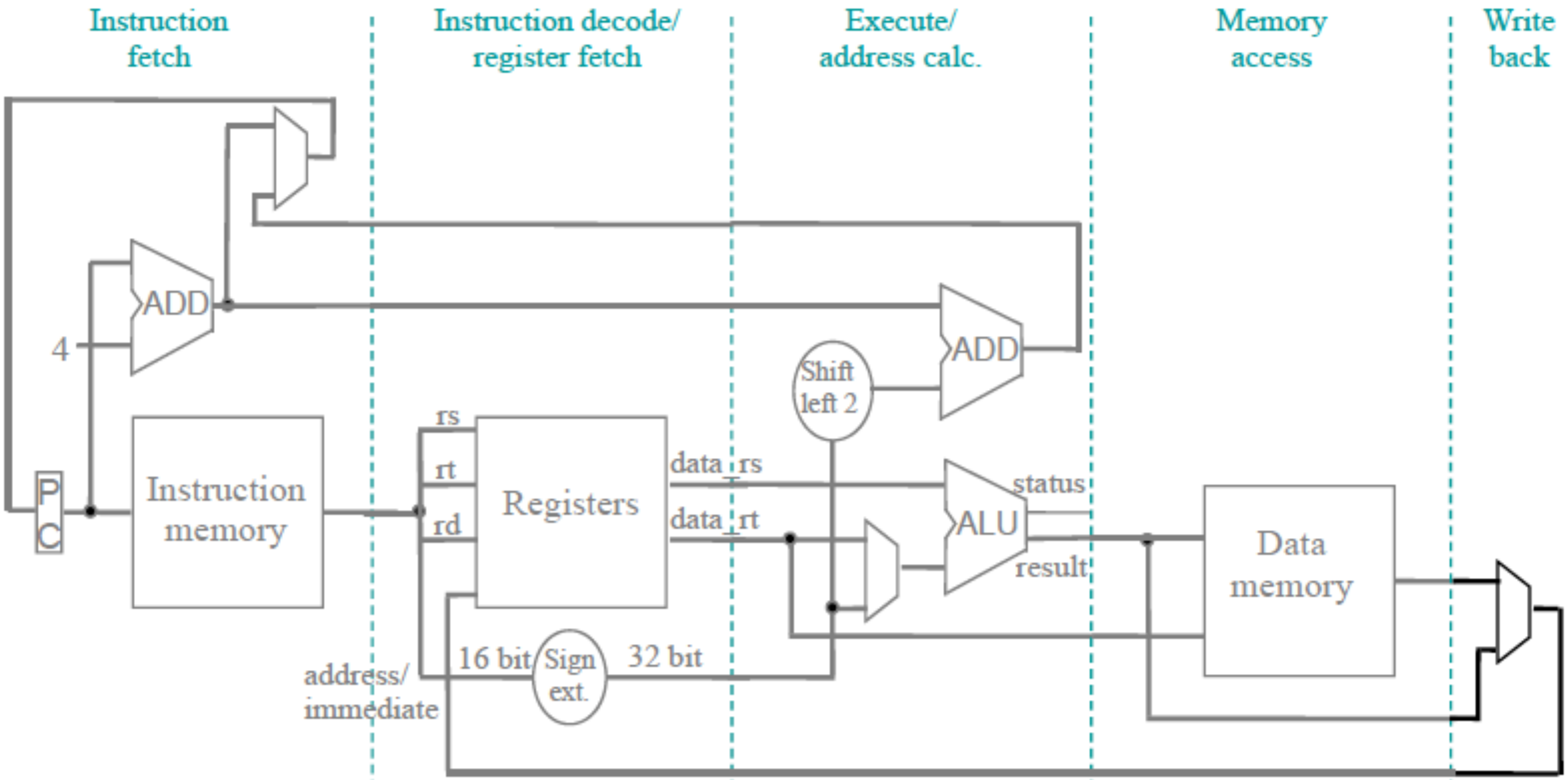
op	rs	rt	address/immediate



Single Cycle Implementation

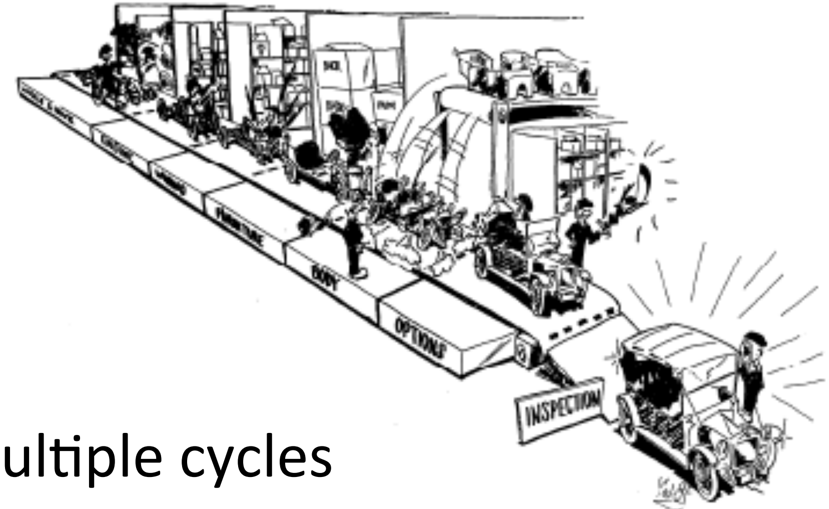
op	rs	rt	rd	shamt	funct

op	rs	rt	address/immediate



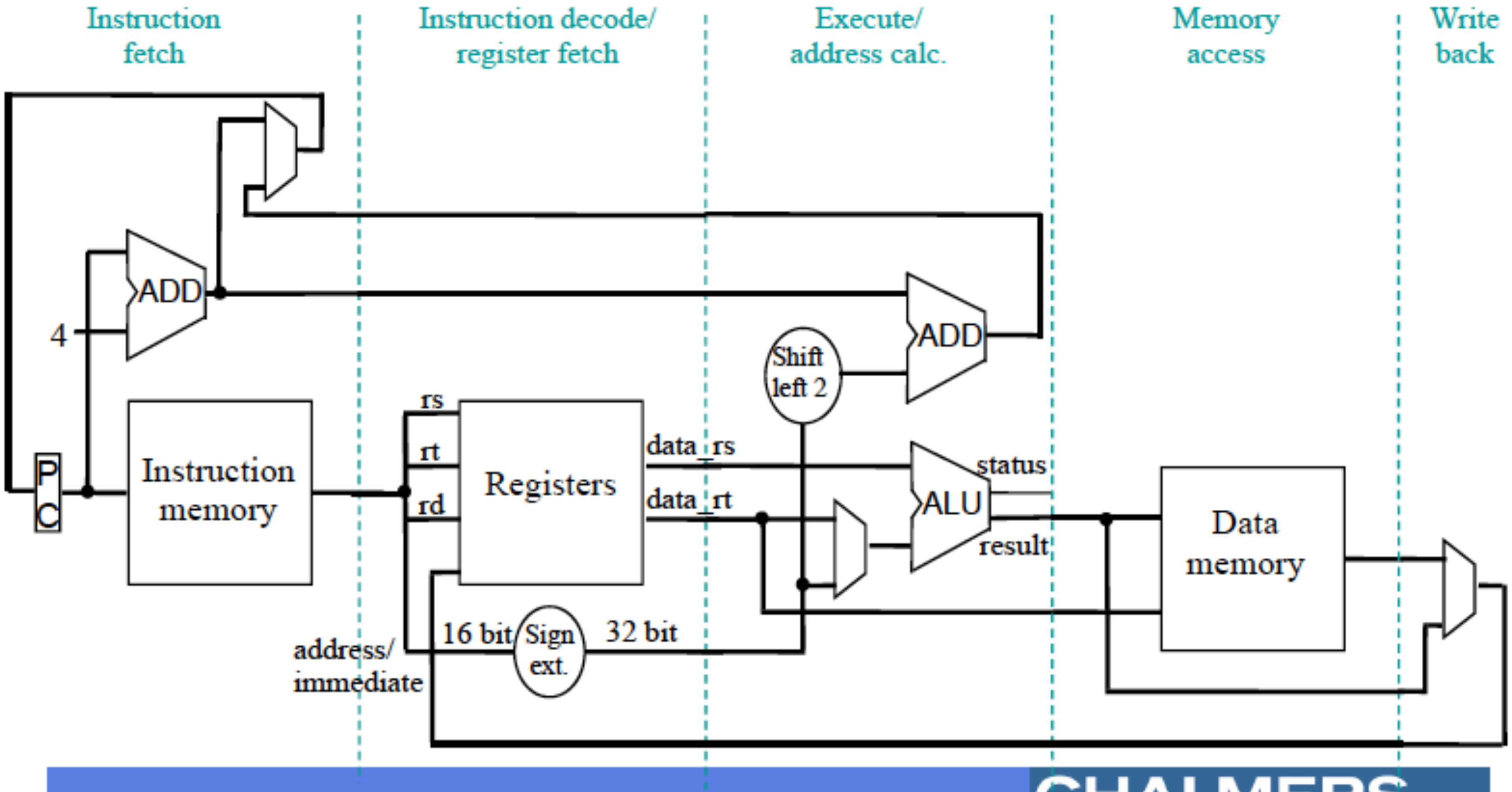
The Assembly Line Concept

- A pipelined processor is based on the assembly line concept
- One station for each stage in the instruction execution
- At any moment there is one instruction at each station
- One new instruction every cycle => CPI=1



Observation:
While each instruction takes multiple cycles to complete, one instruction is completed each cycle!

Pipeline



Instruction fetch
Instruction decode/ register fetch
Execute/ address calc.
Memory access
Write back
ADD
4
PC
Instruction memory
rs
rt
rd
Registers
address/ immediate
16 bit
Sign ext.
32 bit
data_rs
data_rt
Shift left 2
ADD
ALU
status
result
Data memory

Pipelining Example

```
⋮
add $5, $2, $3
lw $4, 100($5)
sw $4, 400($7)
beq $8, $9, 800
⋮
```

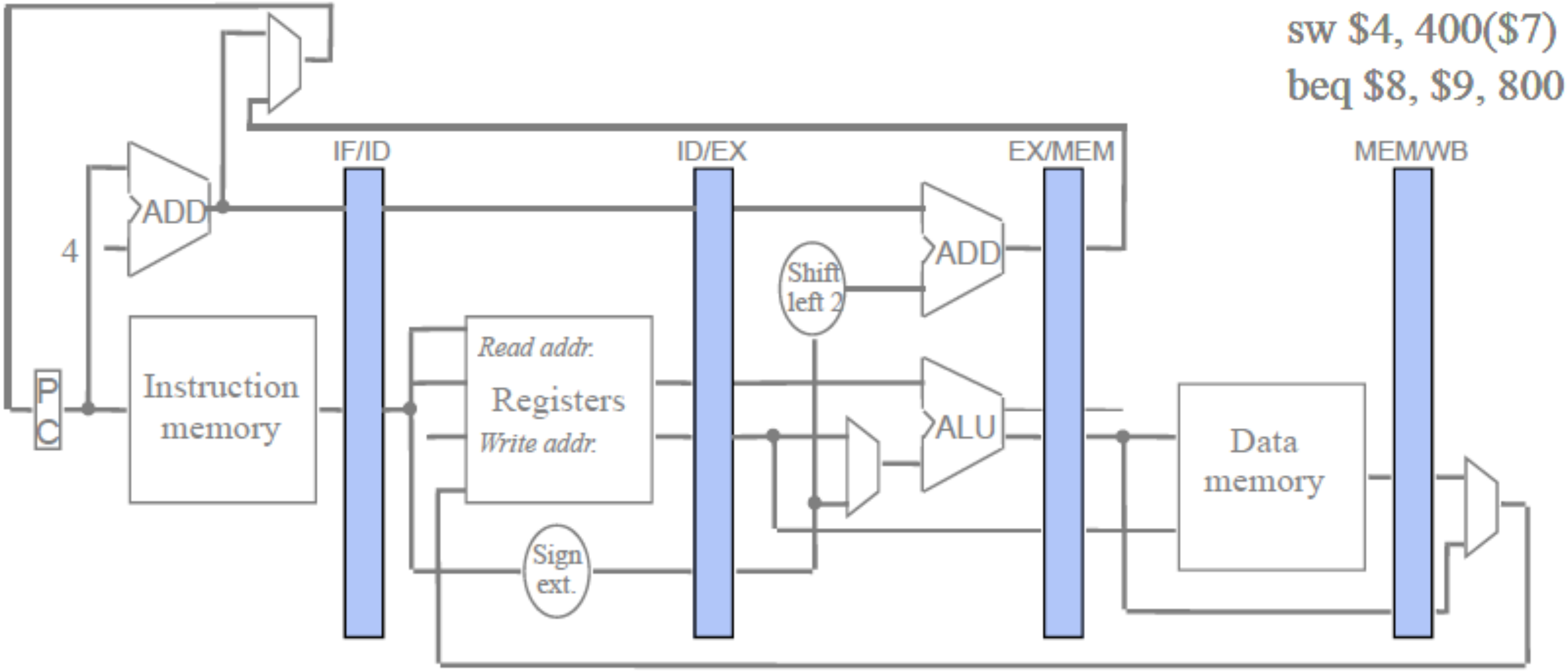
Pipelining Example

PC ⟶ add $5, $2, $3

lw $4, 100($5)

sw $4, 400($7)

beq $8, $9, 800



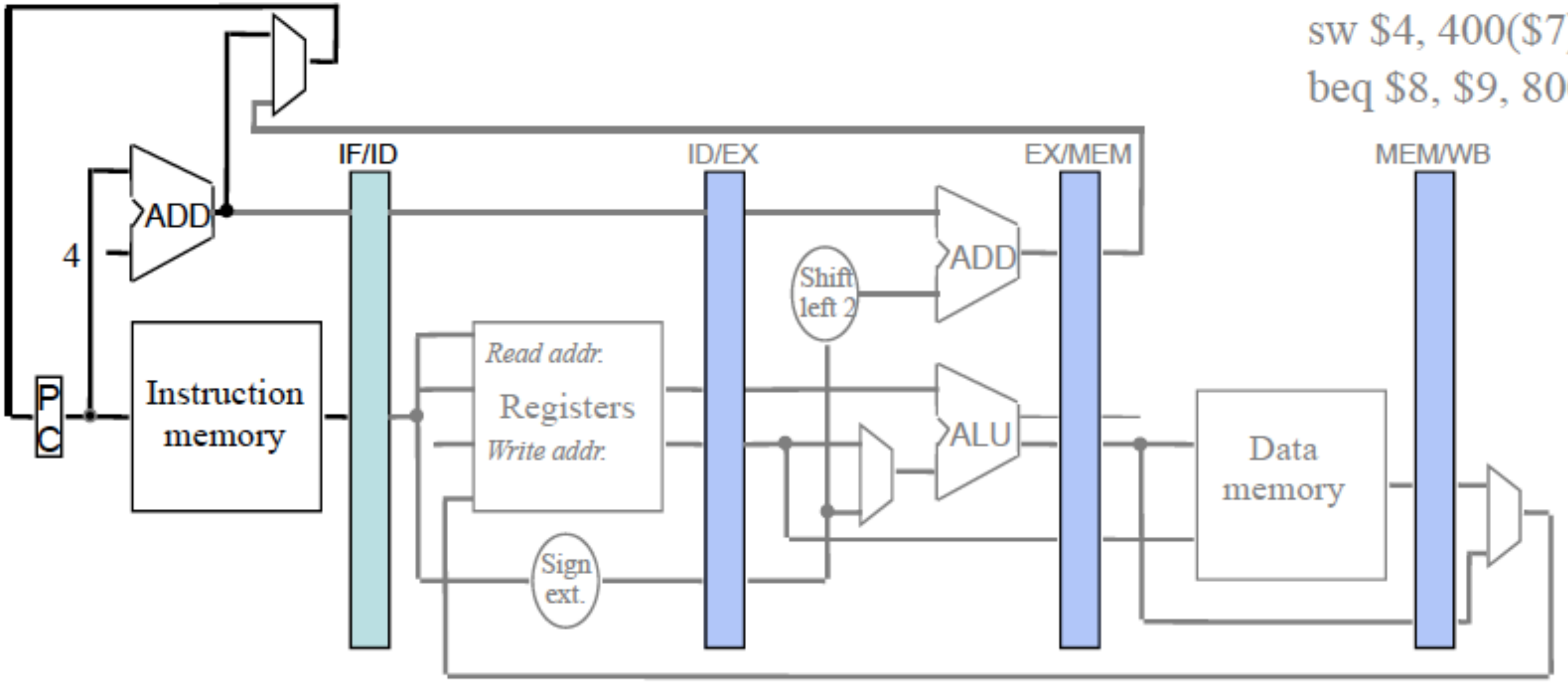
Pipelining Example

PC ⟶ add $5, $2, $3

lw $4, 100($5)

sw $4, 400($7)

beq $8, $9, 800



add $5, $2, $3

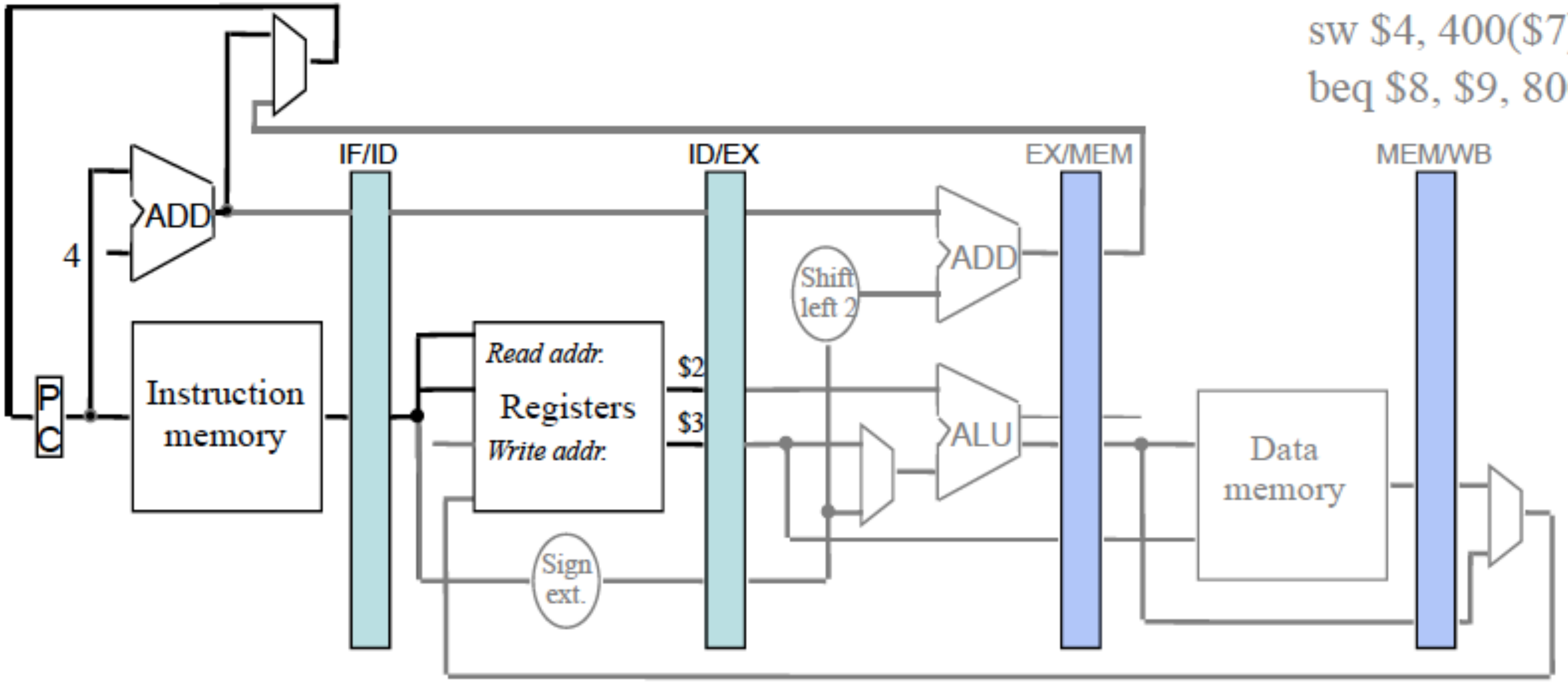
Pipelining Example

add $5, $2, $3

PC ⟶ lw $4, 100($5)

sw $4, 400($7)

beq $8, $9, 800



lw $4, 100($5) add $5, $2, $3

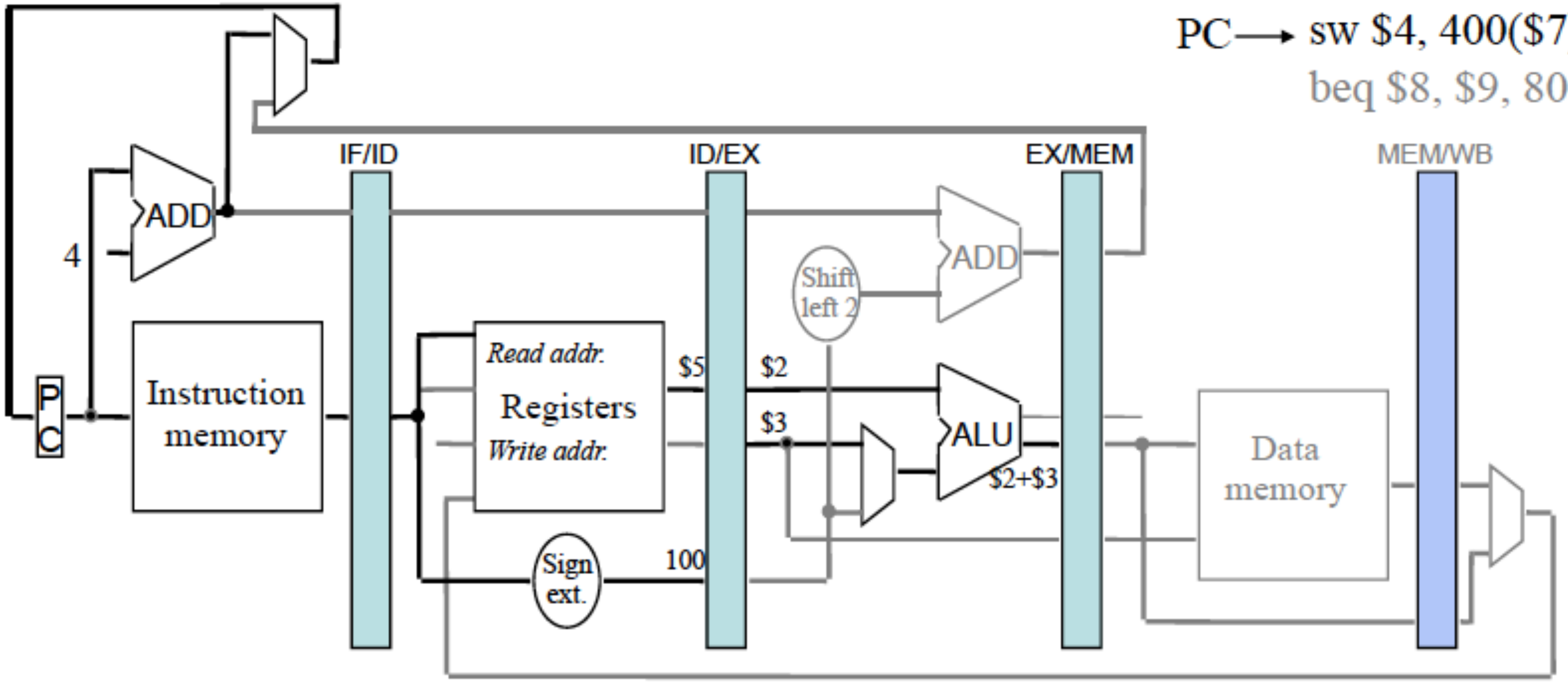
Pipelining Example

add $5, $2, $3

lw $4, 100($5)

PC→ sw $4, 400($7)

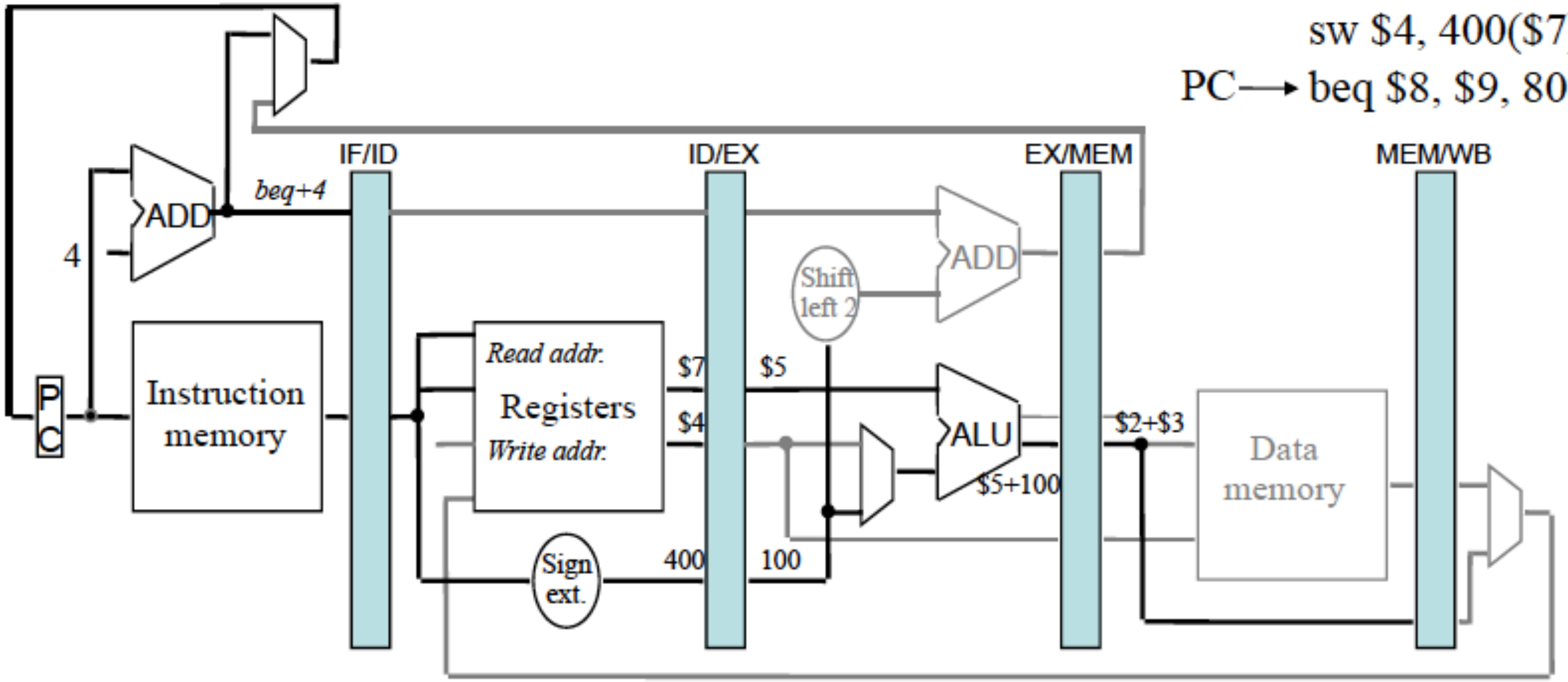
beq $8, $9, 800



sw $4, 400($7)   lw $4, 100($5)   add $5, $2, $3

Pipelining Example

add $5, $2, $3
lw $4, 100($5)
sw $4, 400($7)
PC → beq $8, $9, 800

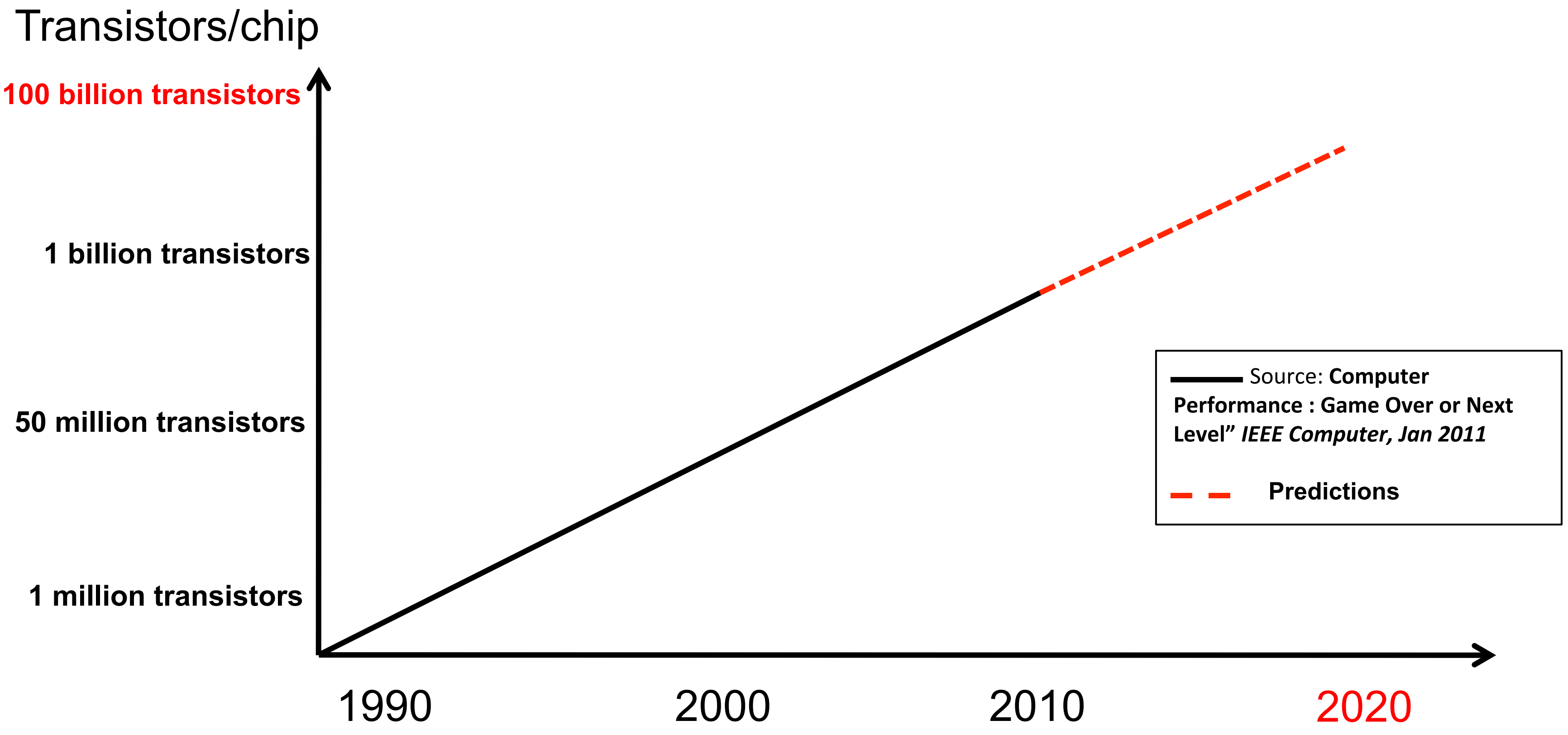


beq $8, $9, 800 sw $4, 400($7) lw $4, 100($5) add $5, $2, $3

CHALMERS
Chalmers University of Technology

Technology Trends

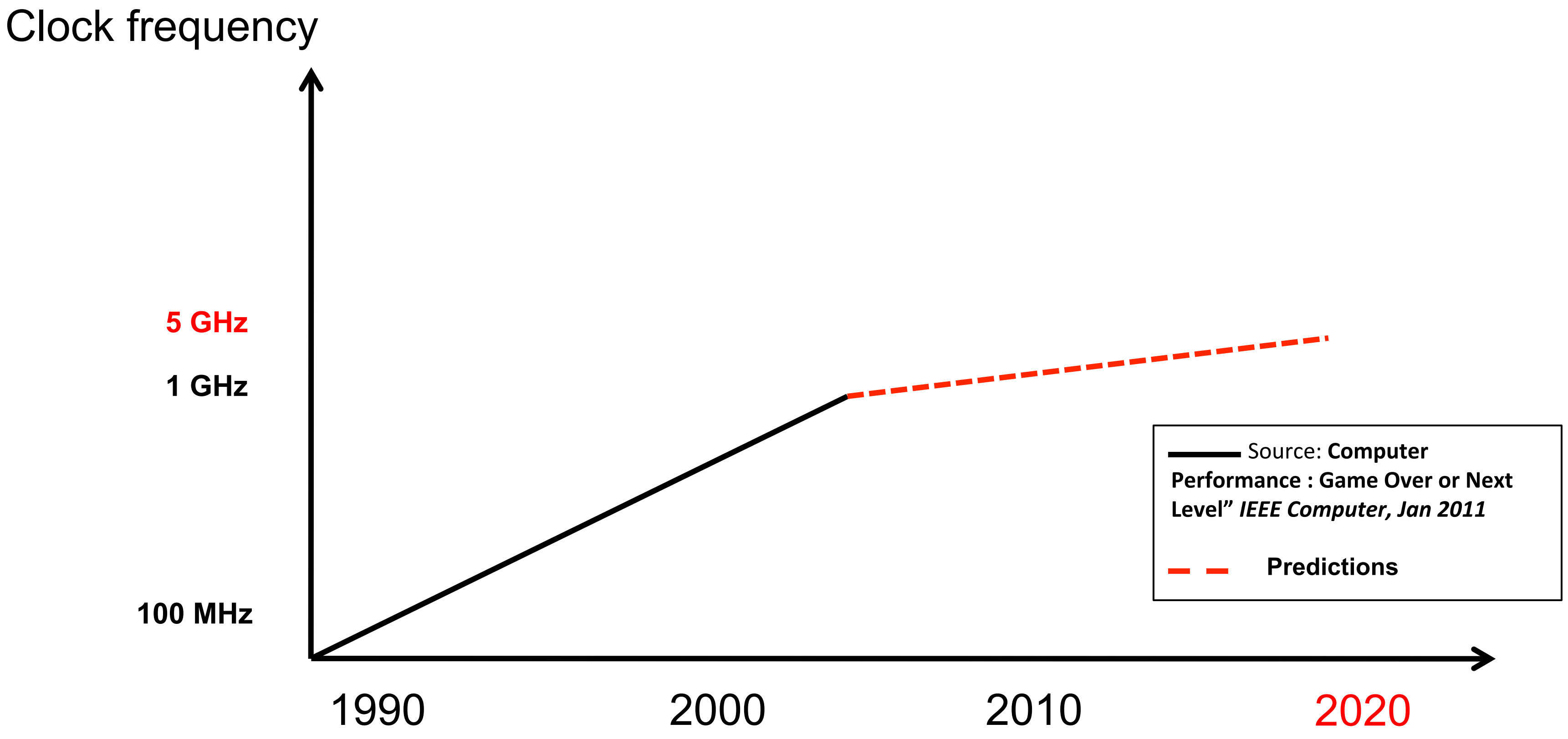
Technology Scaling



Source: Computer Performance : Game Over or Next Level" *IEEE Computer, Jan 2011*

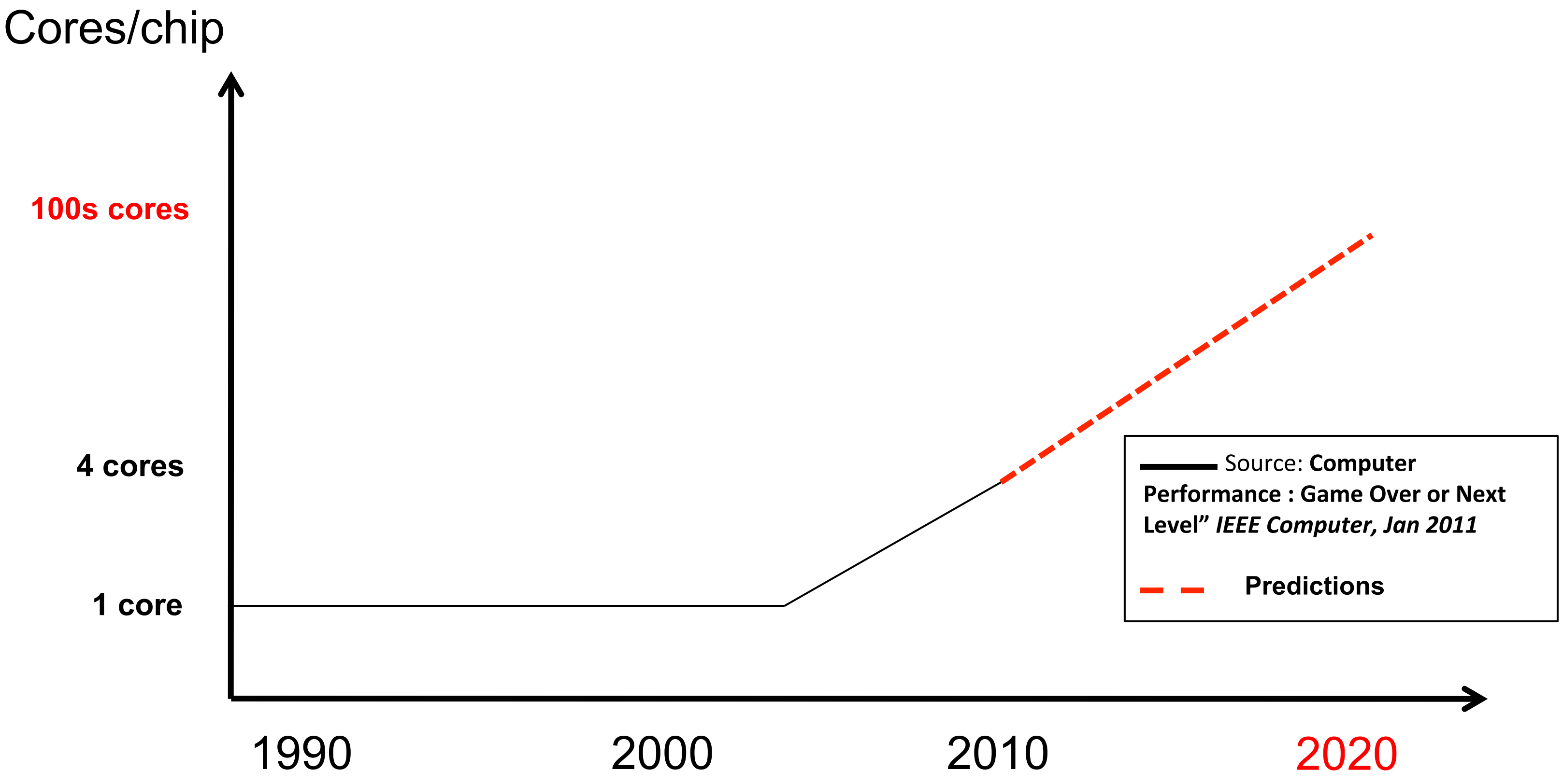
- Good news: Technology scaling will continue

Clock Frequency Scaling



Bad news: Clock frequency will increase slowly at best

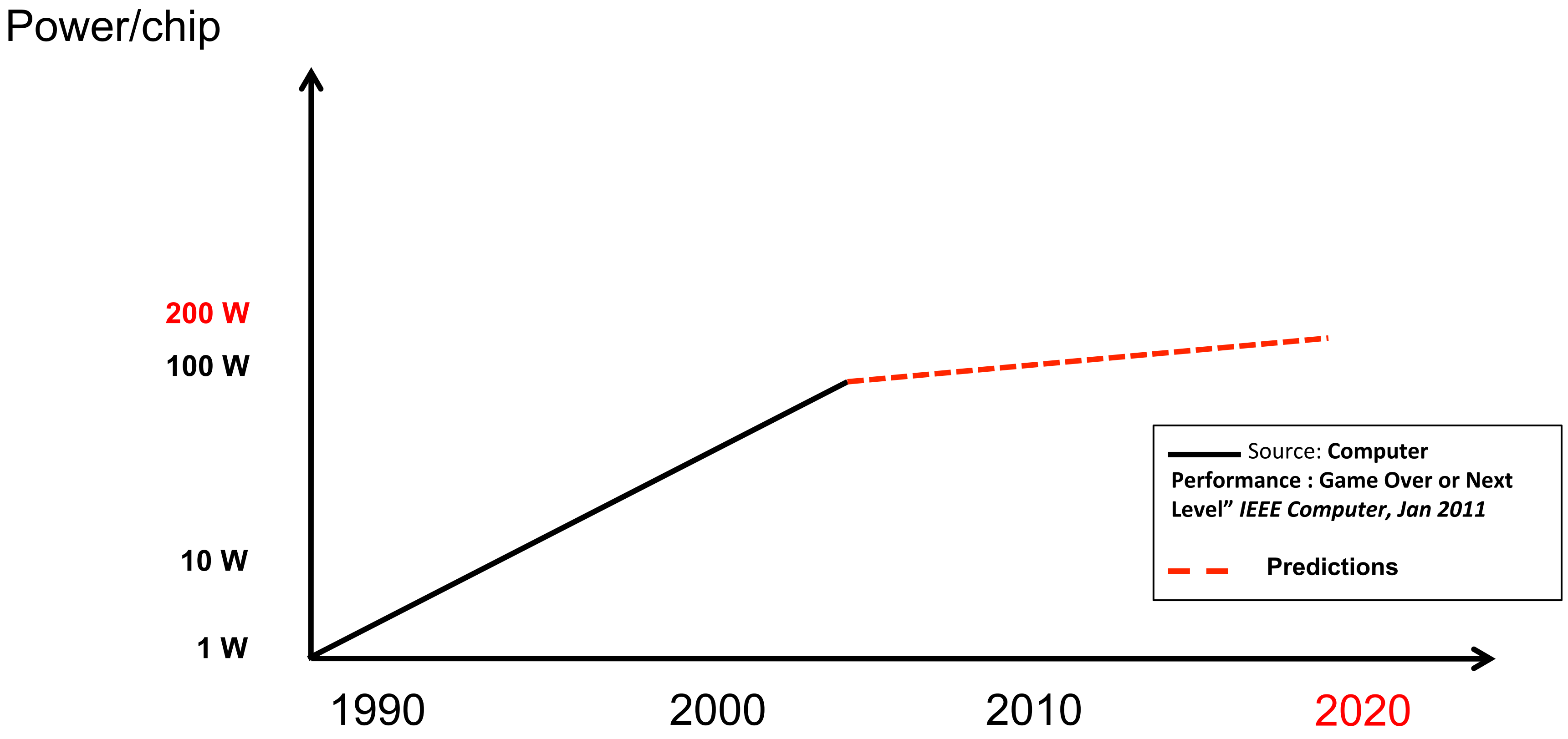
Multicore Scaling



Source: **Computer Performance : Game Over or Next Level" *IEEE Computer, Jan 2011***

By 2020, several hundreds of cores/chip possible

Power Budget per Chip



Source: Computer Performance : Game Over or Next Level" *IEEE Computer, Jan 2011*

Predictions

Bad news: Power budget will increase slowly at best
Power budget: <1W/core!

Trends (summary)

- Technology scaling will **continue**
- Clock-frequency scaling has **discontinued**
- Power budget growth has **discontinued**

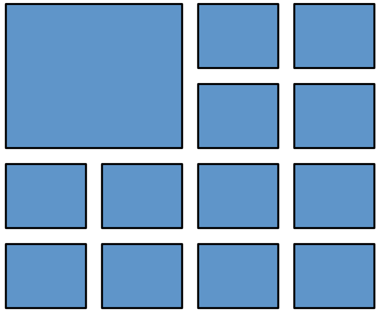
There is considerable room for innovation

The Road Forward

- Parallelism (any form) is our only hope
- Power efficiency is a first-order concern
- Using memory resources efficiently is key

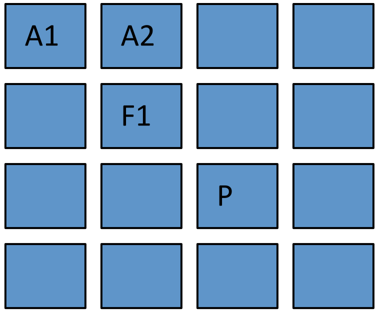
-> **Heterogeneous multicore architectures**

Capability heterogeneous (single ISA)



Capabilities and functionalities can be applied over time and space

Functionally heterogeneous (multi ISA)



Courses offered

- **Computer architecture (DAT105), LP2**
- **Parallel computer organization and design (EDA 282), LP1**
- **Energy aware computing (DAT275), LP4**

Can be studied in any sequence.

Computer Architecture

To master

1. fundamental concepts in computer design to follow advancement in the field
2. design principles of processors (cores) in multicore systems:

   Goal: Uncover parallelism between instructions
3. design principles of memory hierarchies

   Goal: Keep reused data close to the processor
4. design principles of storage systems

   Goal: Retrieve data fast and reliably from a huge repository
5. design exploration techniques: simulation-based

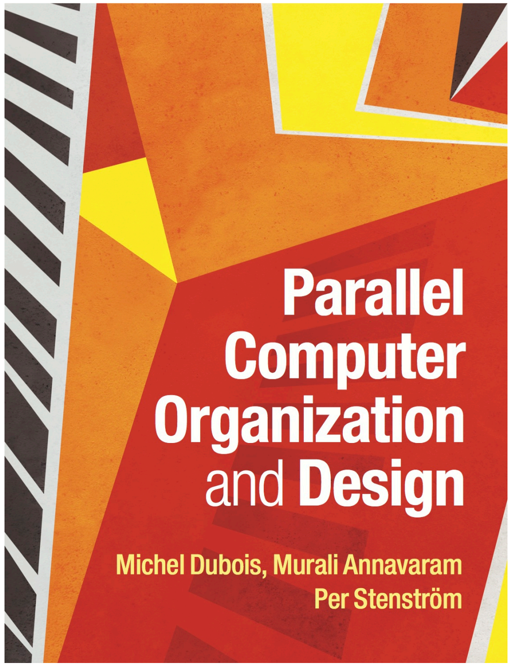
Course Organization

- 7 normal lectures & 2 optional repetition lectures
- 5 problem solution sessions
- 1 design exploration project

Textbook:

Parallel Computer Organization and Design

Dubois, Annavaram, Stenström


Parallel Computer Organization and Design
Michel Dubois, Murali Annavaram
Per Stenström

Parallel Computer Organisation and Design (EDA 282)

To master

1. fundamental concepts in parallel computer architecture to follow advancements in the field
2. parallel programming models and issues involved in designing parallel software
3. design principles of the communication substrate to support parallel programming models including
   1. Message passing systems
   2. Shared memory multiprocessors
   3. Interconnection networks
   4. Memory coherence and consistency

Energy Aware Computing (DAT275)

To master

- why energy aware computing is important
- electrical mechanisms that cause power and energy to be dissipated
- strengths and weaknesses of different classes of computers w.r.t. energy efficiency
- computer architecture techniques to reduce energy
- simulation tools to estimate energy usage for computer applications