Lecture 12
Summary

Main topics

What use is this for?

What next?

Next Courses?

Next topics?

Goals

- To understand basic features of a computer system, from the point of view of the executing program
- To understand, how a computer systems executes the program given to it
- To understand the execution time program representation in system
- To understand the role and basic functionalities of the operating system

What use is this course for?

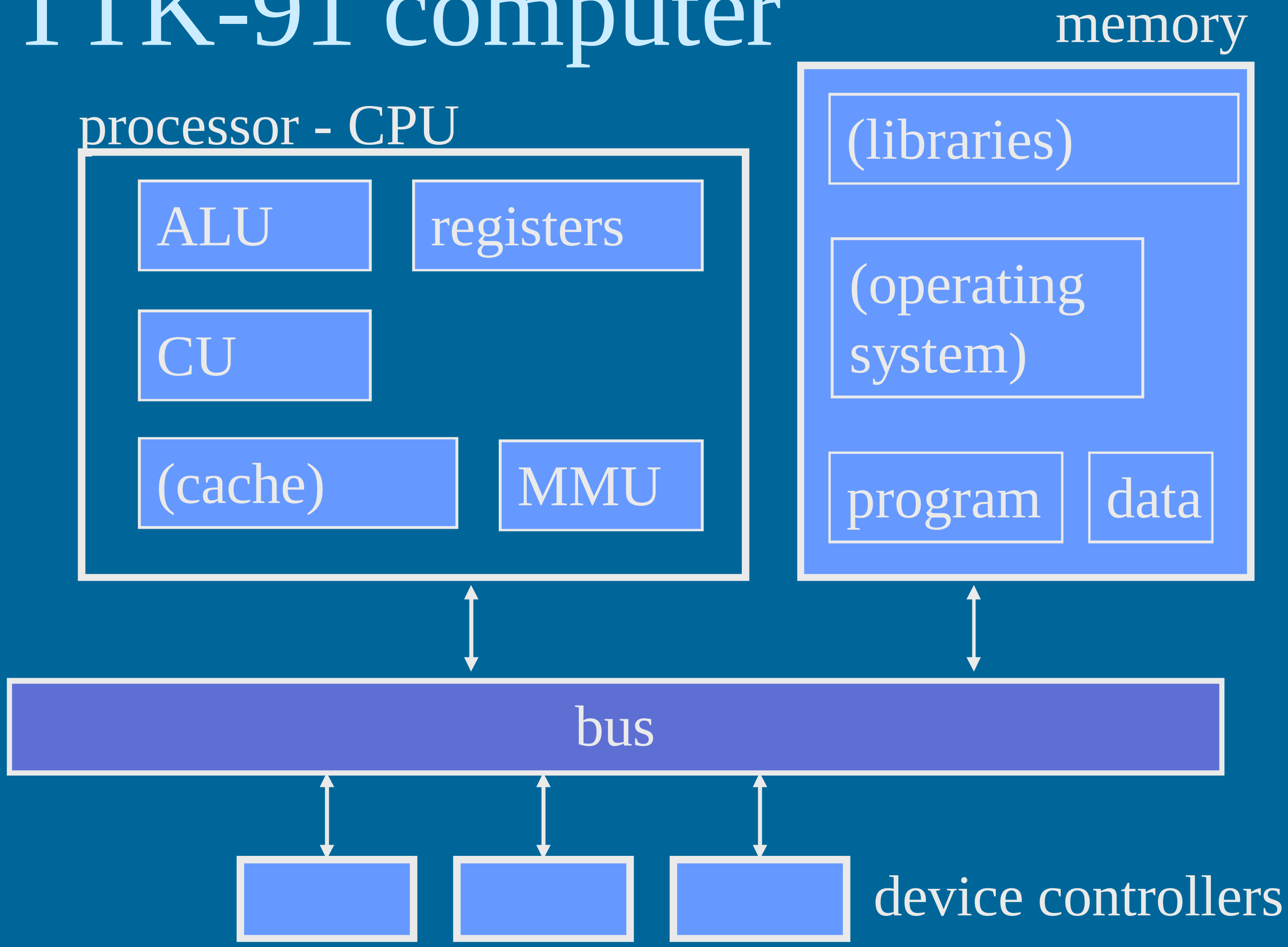
- Program execution speed is based on machine instructions executed by the processor (CPU), and not in the program representation format in high level language
  - High level language representation is still important
- Understanding higher level topics is easier, once one first understands what happens at lower levels of the system

Main Topics

- System as a whole, speed differences
  - Example machine and its use
- Program execution at machine language level
  - Processor, registers, bus, memory
  - Fetch-execute cycle, interrupts
  - Activation record stack, subroutine implementation
- Data representation formats (program vs. hardware)
- I/O devices and I/O implementation
  - Device drivers, I/O interrupts, disk drive
- Operating system fundamentals
  - Process and its implementation (PCB)
  - Execution of programs in the system
  - Compilation, linking, loading
  - Interpretation, emulation, simulation

Examples on the following slides

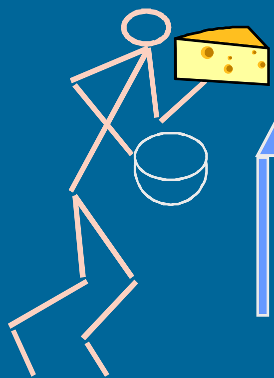
Example architecture: TTK-91 computer



Speed differences: Teemu’s Cheese Cake

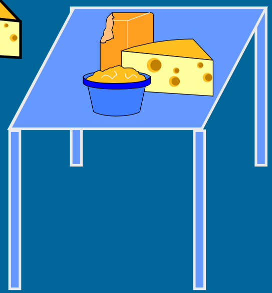
The speed of registers, cache, disk drive and web as compared to finding cheese for cheese cake.

hand



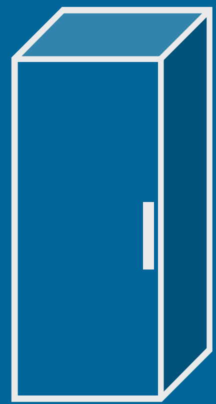
0.5 sec
(register)

table



1 sec
(cache)

Refridge-rator

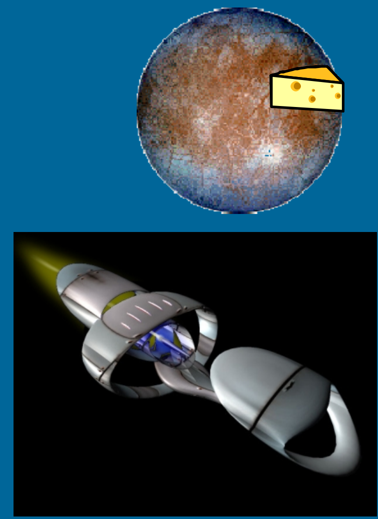


10 sec
(memory)

Moon

12 days
(disk)

Europa (Jupiter)



Michael Crumpton

4 years
(web, human)

2008: 0.5 ns? gap widens 10 ns? gap widens 4 ms? (50 days?) 1 s? (65 yrs?)

Assembly language programming

```
for (int i=20; i < 50; ++i)
        T[i] = 0;
```

variables, constants,
arrays (2D), records

in memory, in registers?

selection, loops,
subroutines, SVC's,
parameters,
local variables

```
I       DC      0
        ...
        LOAD    R1, =20
        STORE   R1, I

Loop    LOAD    R2, =0
        LOAD    R1, I
        STORE   R2, T(R1)

        LOAD    R1, I
        ADD     R1, =1
        STORE   R1, I

        LOAD    R3, I
        COMP    R3, =50
        JLES    Loop
```

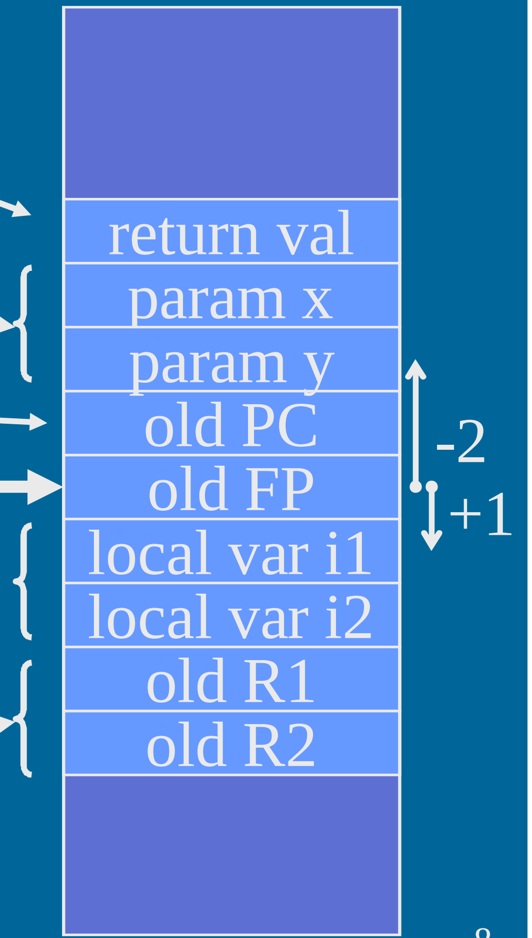
Activation record (Activation record stack)

int funcA (int x,y);

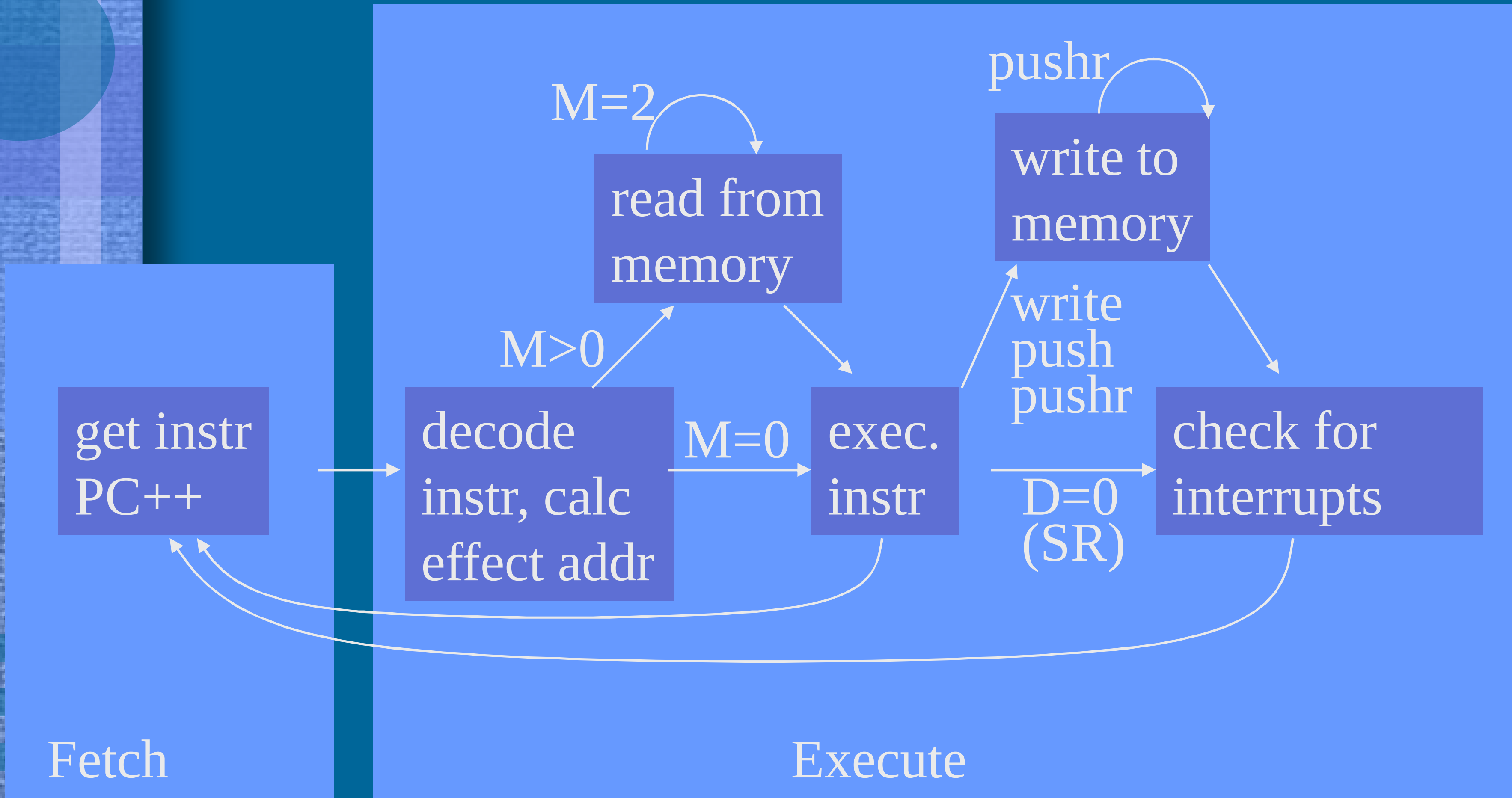
- Subroutine implementation (ttk-91)
  - function return value (or all return values)
  - all (input and output) parameter values
  - return address
  - previous activation record
  - all local variables and data structures
  - saved registers values for recovering them at return

Parameter types?

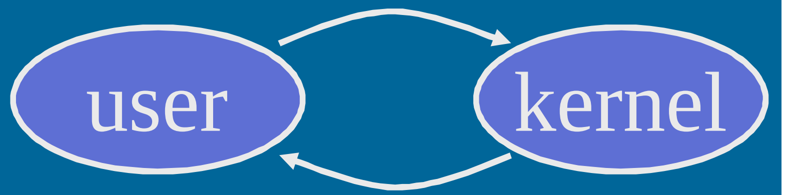
call-by-value, call-by-reference, call-by-name



Instruction fetch-execute cycle



Processor execution mode



- User mode (normal mode)
  - Can use only ordinary instructions
  - Can reference only user's own memory areas (MMU controls)
- Privileged or kernel mode
  - Can use only all instructions, including privileged instructions (e.g., clear_cache, iret)
  - Can reference all memory areas, including kernel memory
    - Can (also) use direct (physical) memory addresses

When and how mode changes?

Data representation formats

1/8 = 0.1250
1/16= 0.0625
0.1875

“+”	“15”	“0.1875” = “0.0011”
sign	exponent	mantissa or significand

integers
floating points
character
character strings
pictures, sounds

non-standard data?

which data is (not) understood by the processor?

so that ...

1 ... ely

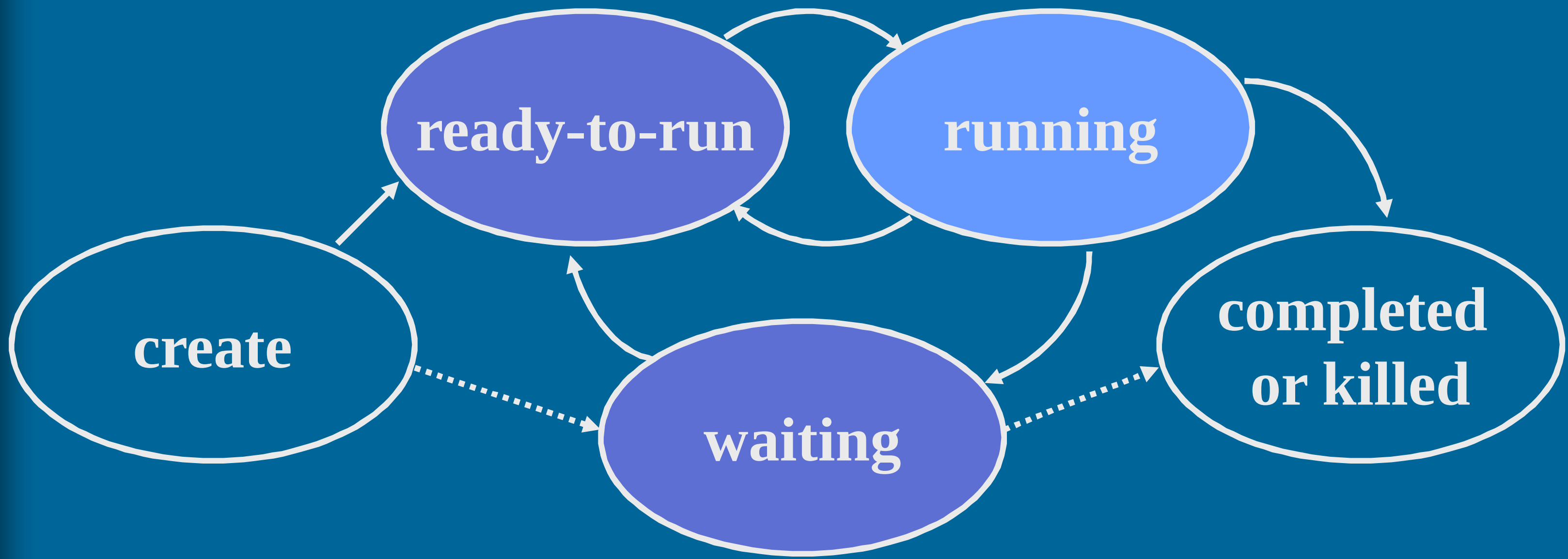
2

3

mantissa	exponent
0.0011	“15”
1.1000	“12”
1000	“12”

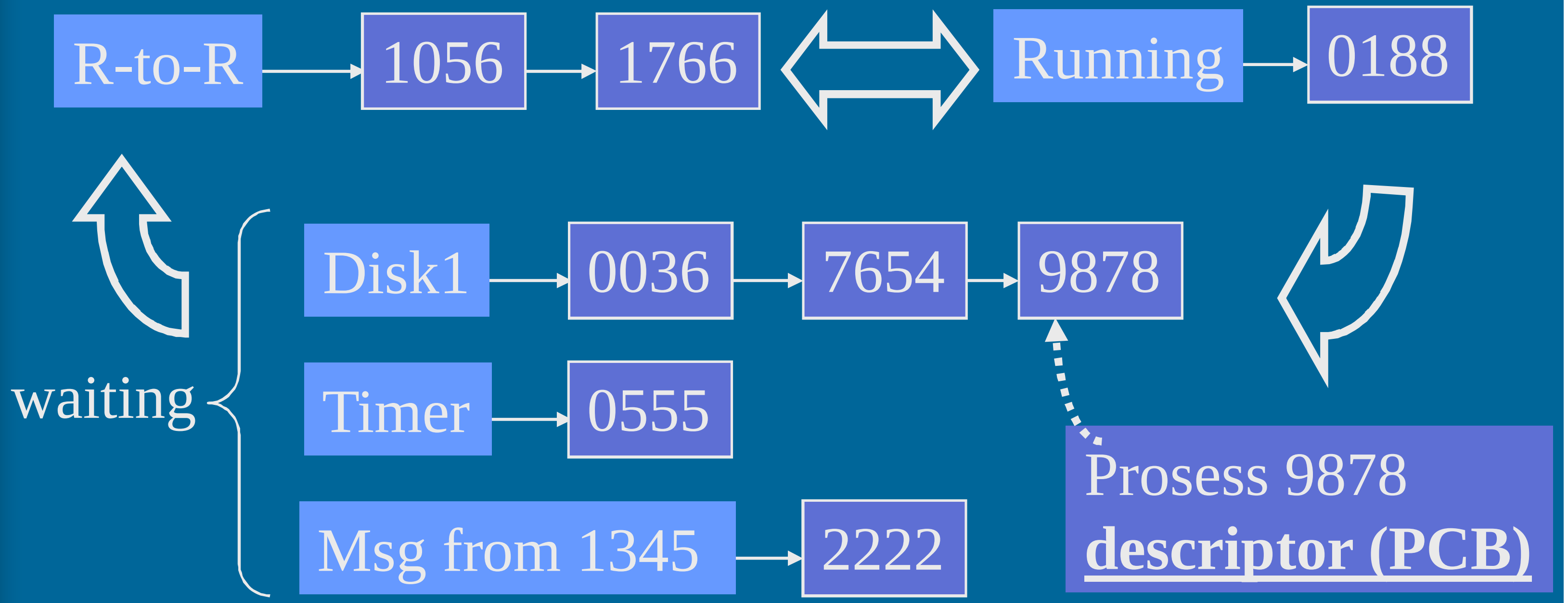
24 bit mantissa!

Process, prosess States and Life Time



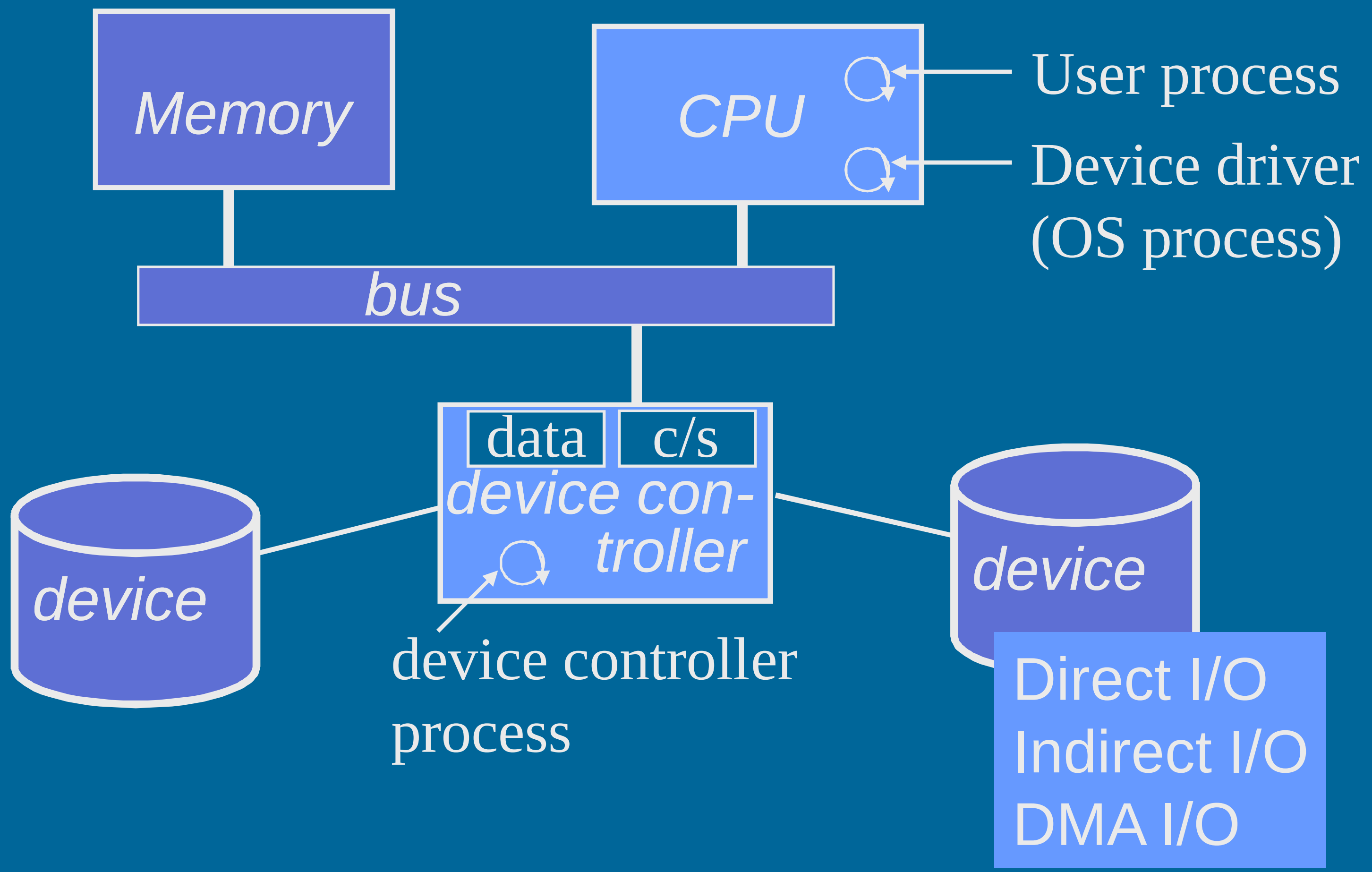
When will state change?
What happens in state change (at instr. level)?
Who or what causes the state change?

Prosesses in Queues, PCB



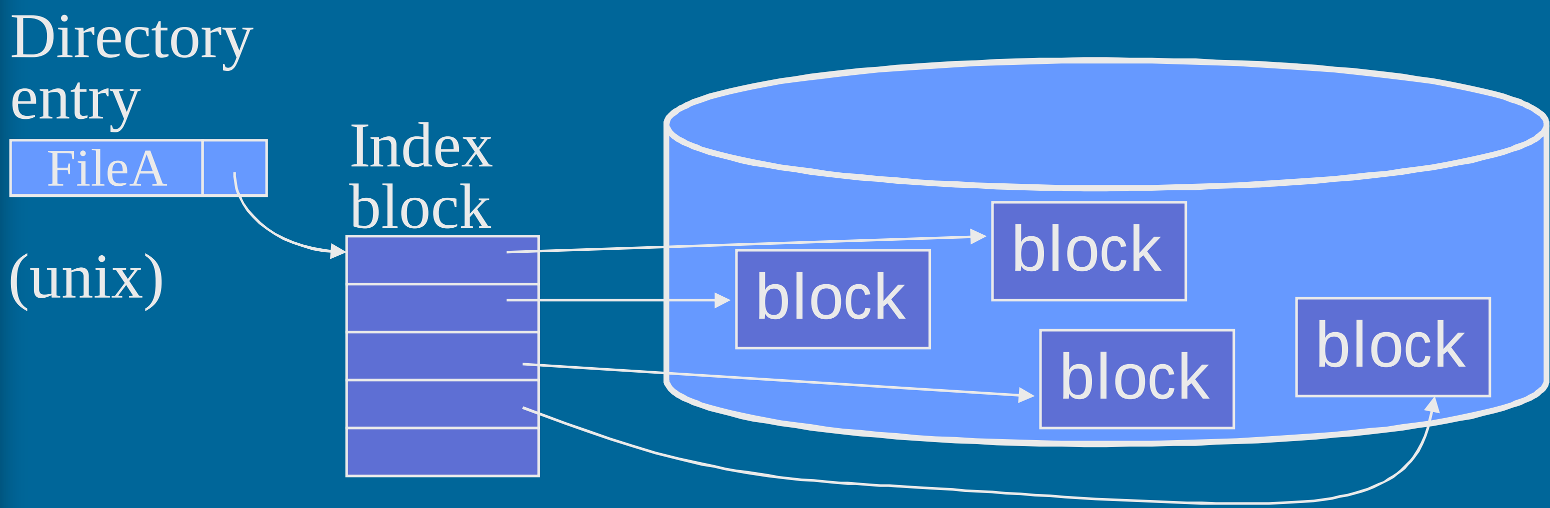
scheduling:
select next process in Ready-to-Run queue and
move it to CPU for execution
(copy processor state for this process into processor registers)

I/O Implementation, Device Controller and Device Driver

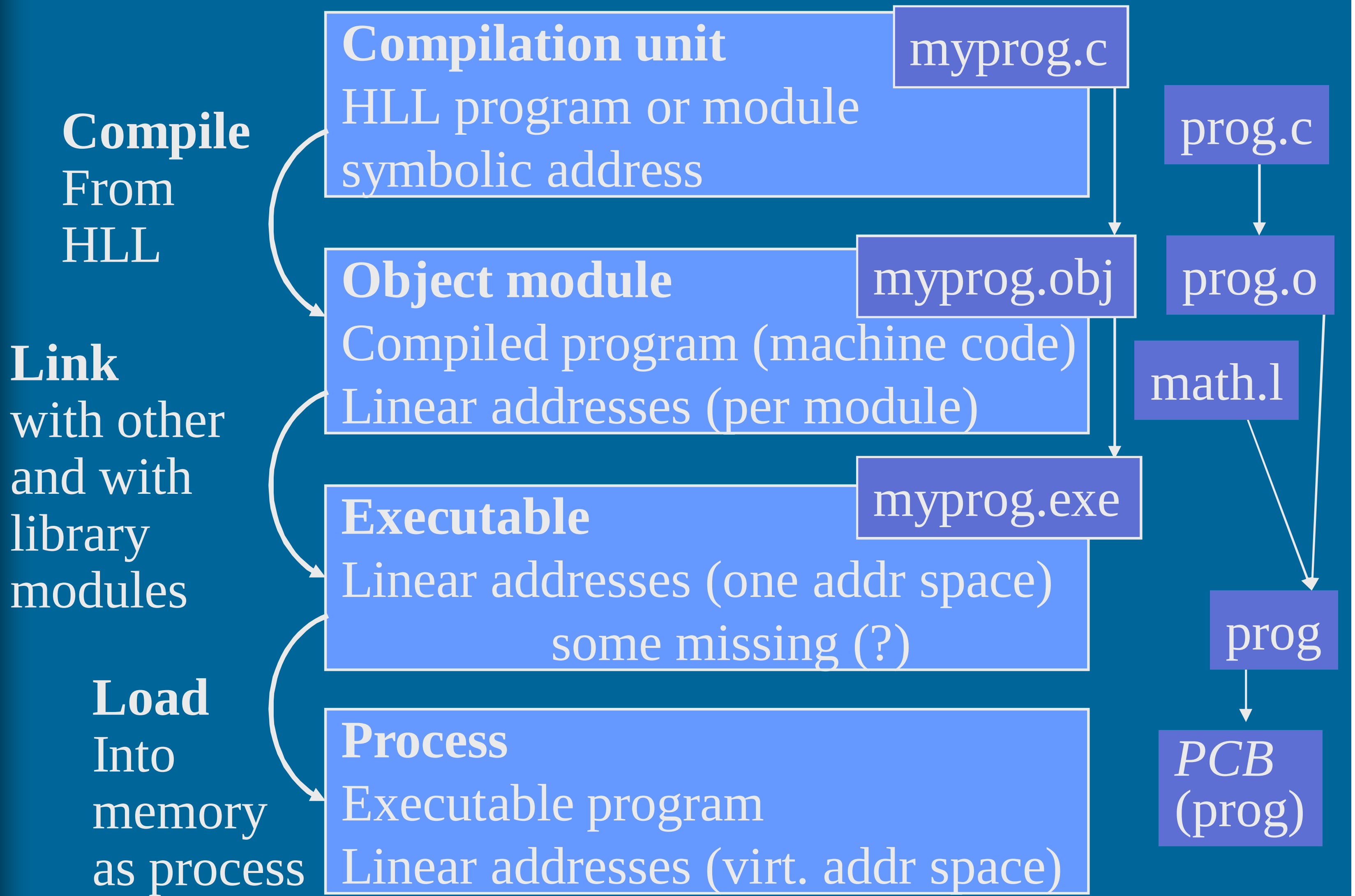


Disk Use

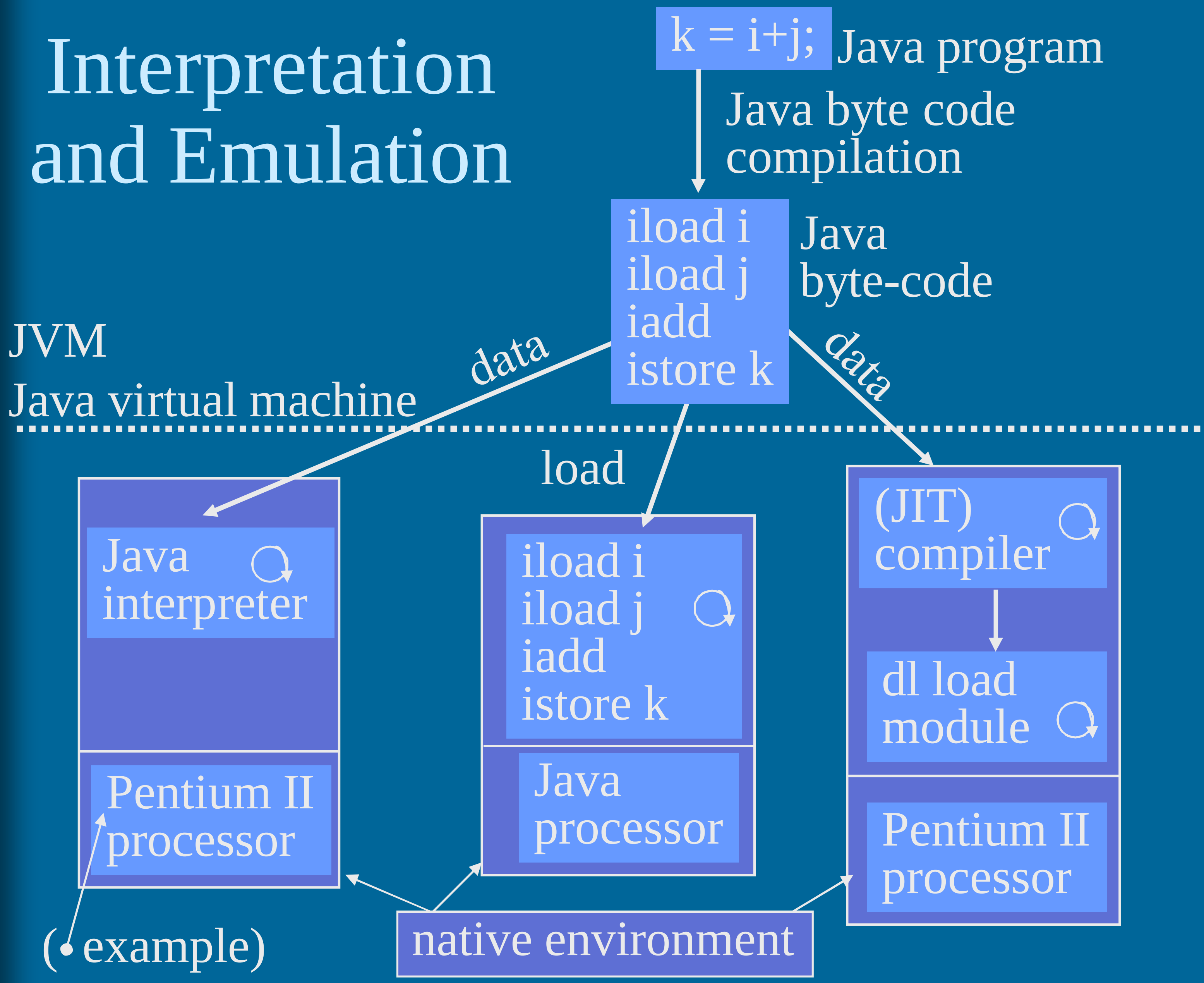
- A file is composed of multiple blocks
  - block per disk sector (2-4 sectors?)
- Disk directory contains information on all blocks used by each file
  - blocks are read in correct order



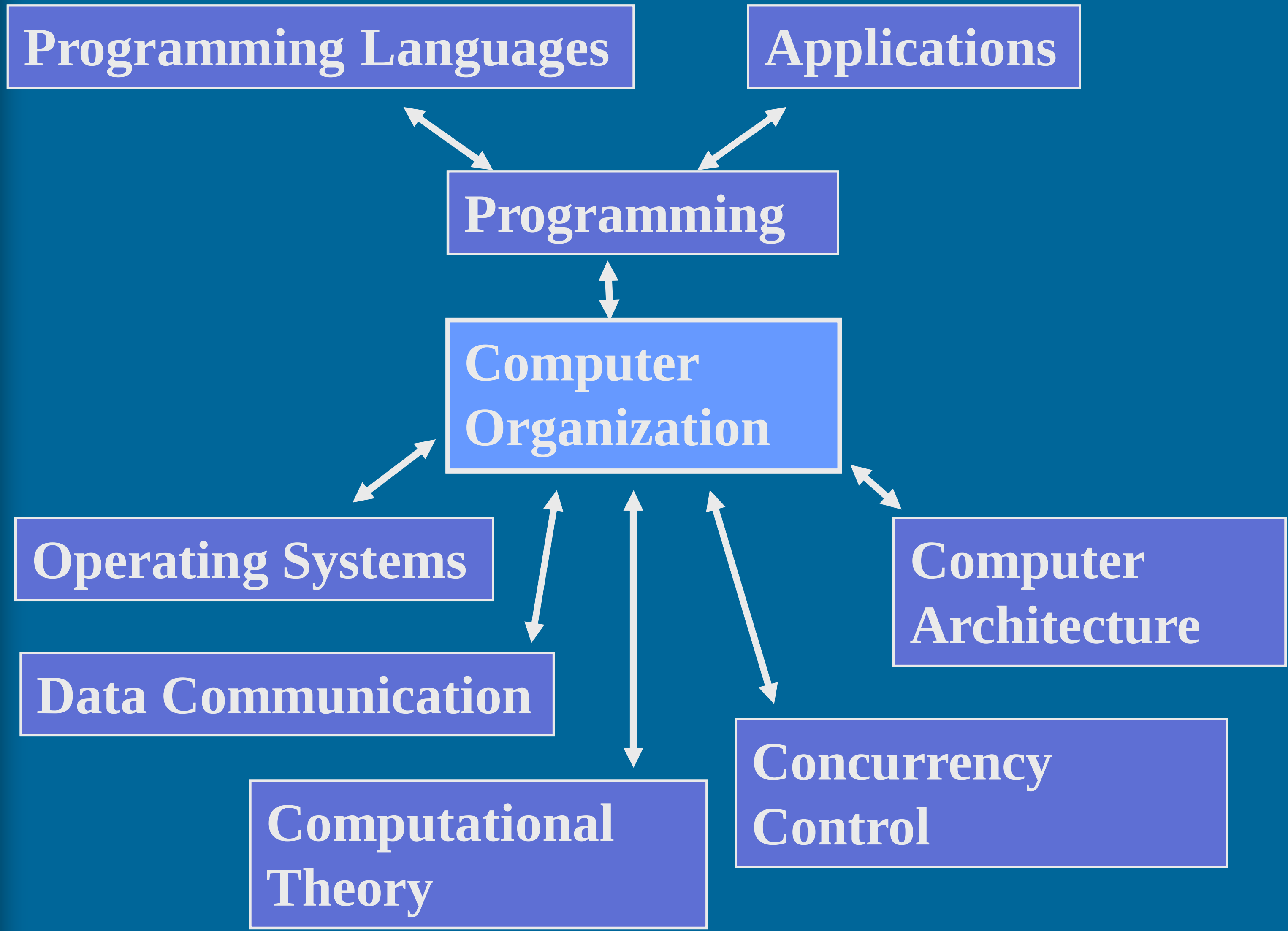
From High Level Language (HLL) to Execution



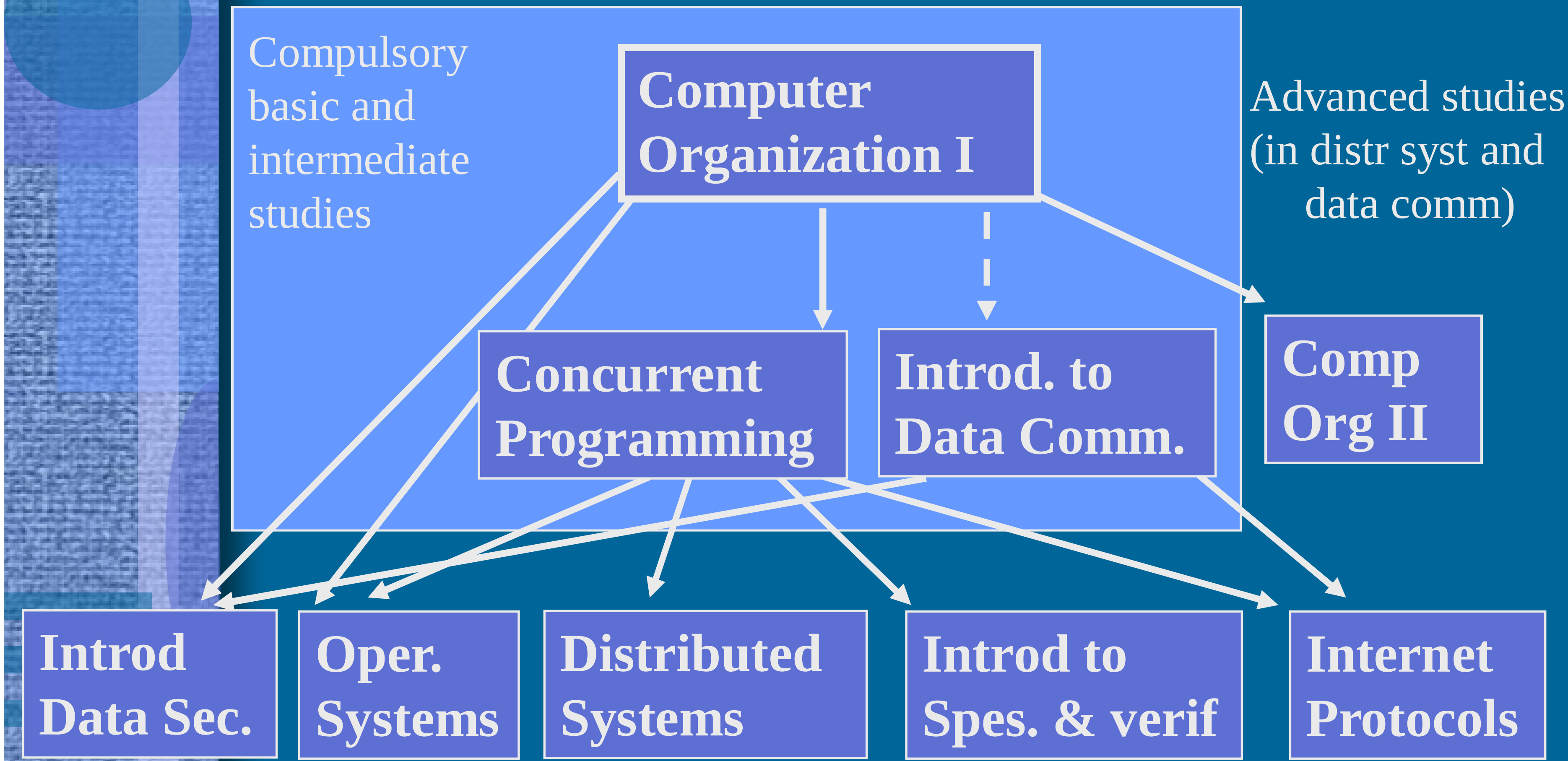
Interpretation and Emulation



Topic Dependencies



Course Dependencies



Computer Organization II, 4 cr

- 2nd year students
  - Elective course in BSc or MSc studies
- Prerequisites: CO-I
- In most universities combined with CO-I
- One level down from CO-I in implementation hierarchy
  - ”How will hardware clock cycle make the processor to execute instructions ?”
  - ”How is processor arithmetic implemented?”
  - Many instructions in execution concurrently (in many ways!)
    - How is this implemented, what problems does it cause, and how are those problems solved?

CO-II ….

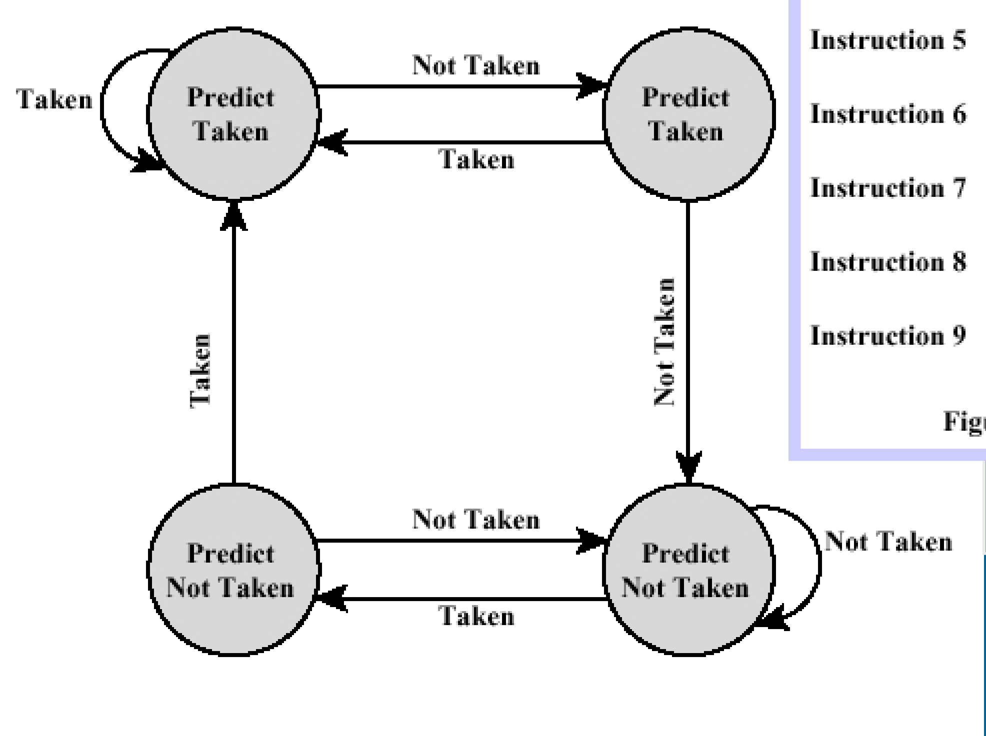


Figure 11.16 Branch Prediction State Diagram

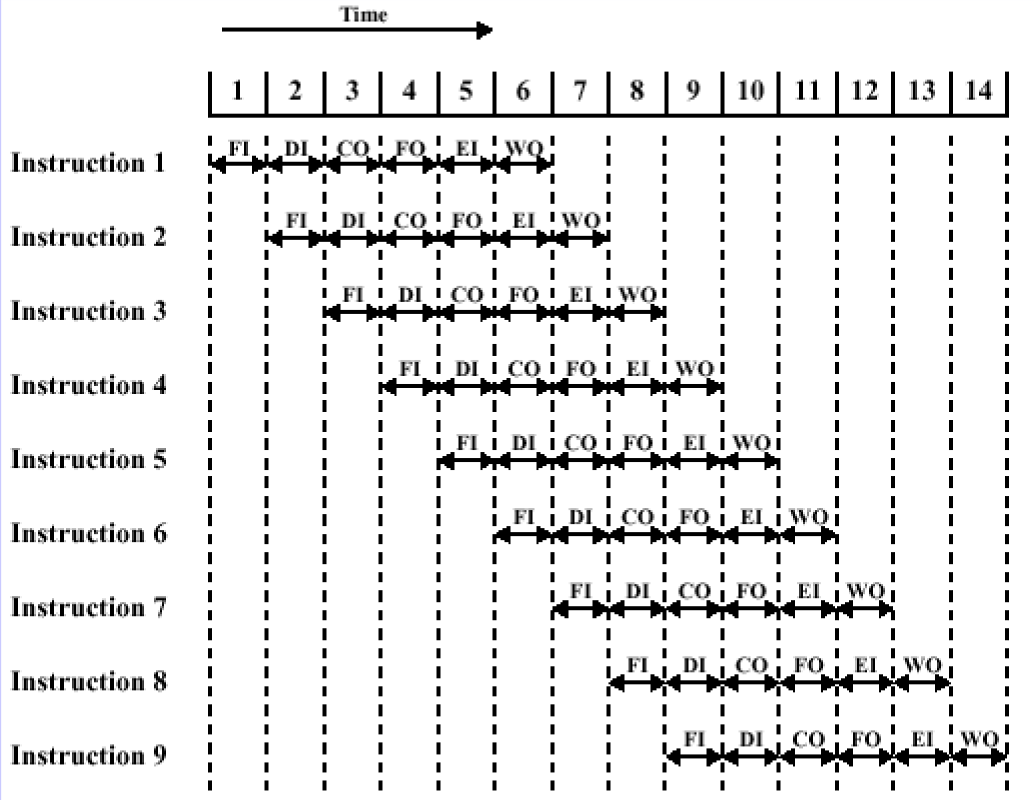


Figure 11.11 Timing Diagram for Instruction Pipeline Operation

[Stal99]

Operating Systems (OS), 4 cr

- 4th year students
  - Compulsory for graduate (M.Sc.) students of the distributed systems and telecommunication specialisation area
- Prerequisites
  - CO-I
  - Concurrent Programming
  - Introduction to Data Communication
- OS role as process and resource controller
- Concurrent processes using shared resources
- Use of system resources
- Process scheduling
- More?
  - Distributed Systems, 4 cr

OS ...

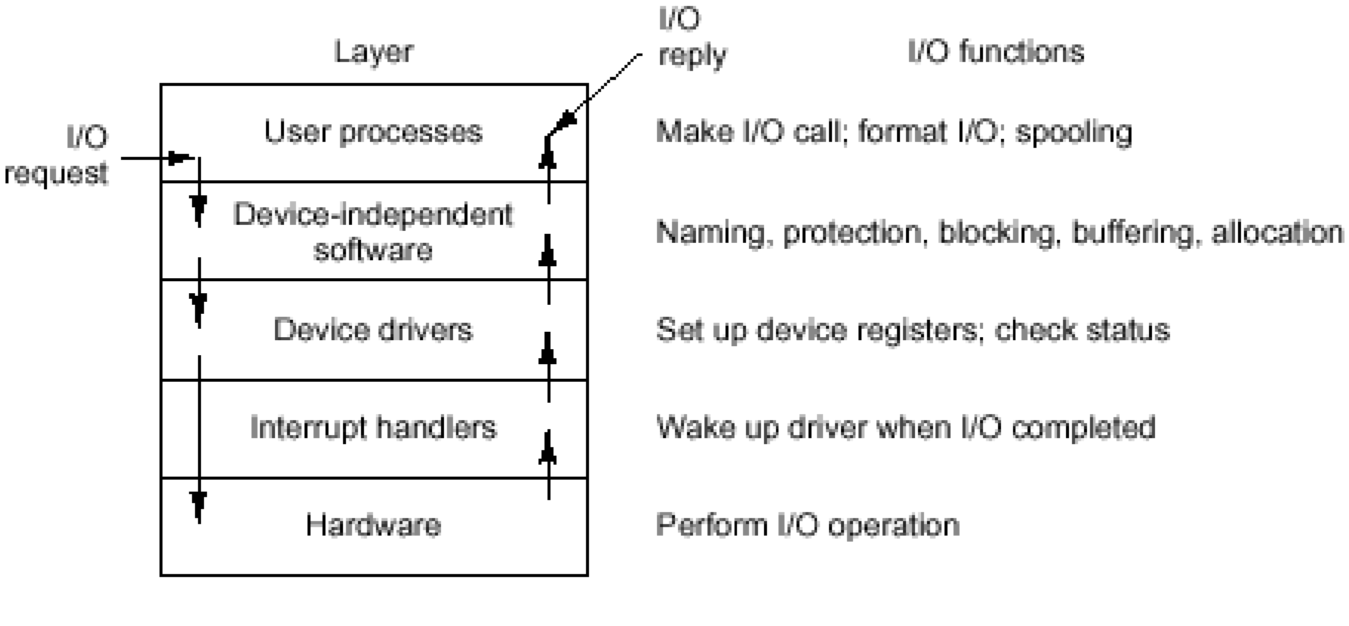


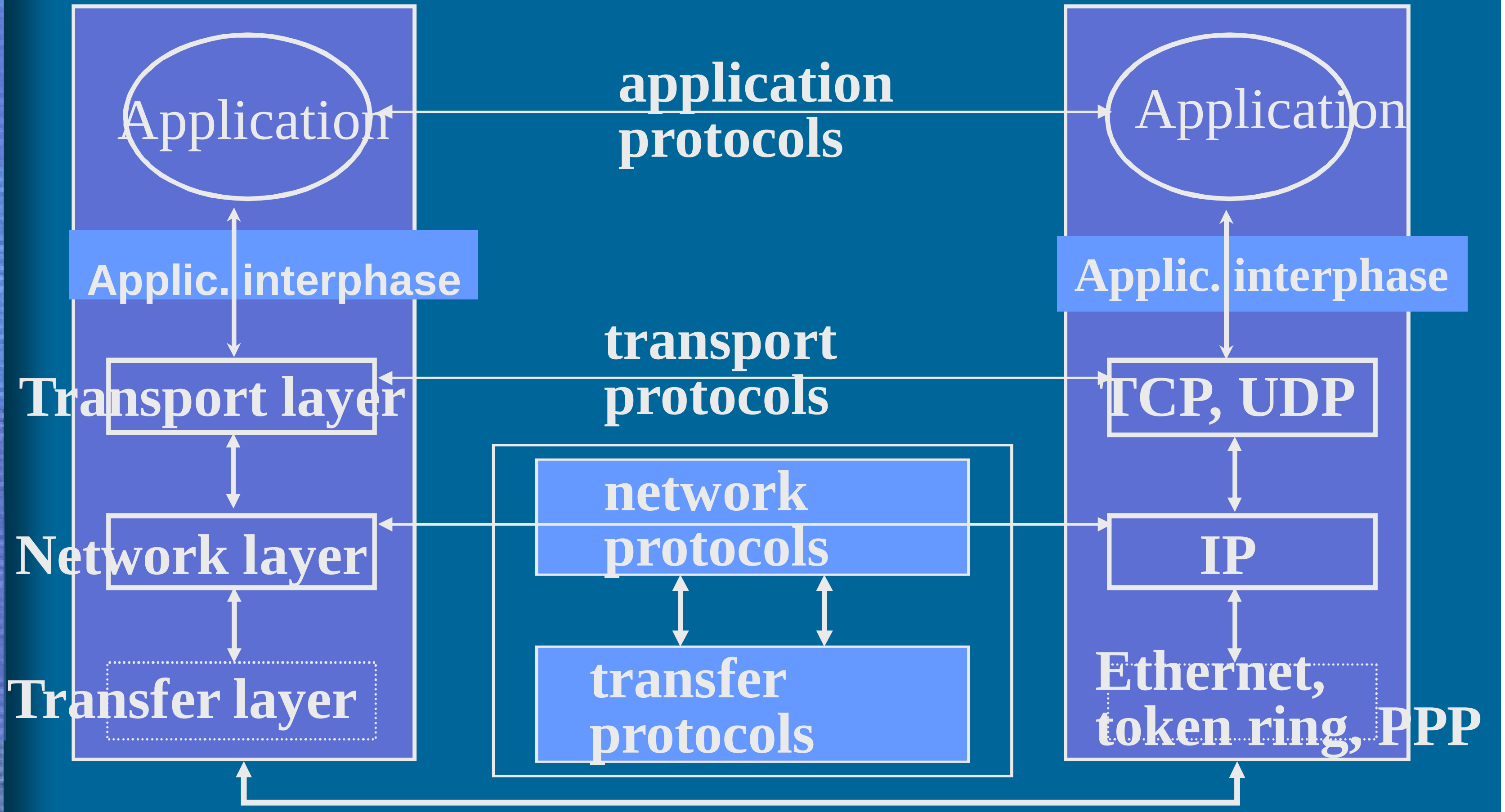
Figure 3-6. Layers of the I/O system and the main functions of each layer.

Intro to Data Communication, 4 cr

- 2nd year students
  - Obligatory undergraduate course
- Computer network basic services to users and applications
- Basic tools for data communication
- Network architecture layer structure and services at each layer
- More?
  - Internet-protocols, 2 cr

Introduction to Data Communication

TCP/IP -layers



Concurrent Programming (CP), 4 cr

- 2nd year students
  - Obligatory undergraduate course
- Prerequisites: CO-I
- Problems caused by concurrency
  - System just freezes … why?
- Concurrency requirements for system
- Process synchronization
  - Busy wait or process switch? Why?
- Prosess communication
  - Shared memory? Messages? Why?
  - Over the network?
- More?
  - Distributed Systems, 4 cr

semaphores
monitors
rendezvous
guarded statements
rpc, messages
Java concurrent progr.

CP - Synchronization Problem Solution with Test-and-Set Instruction

- TAS Ri, L (ttk-91 extension)

Ri := mem[L]
if Ri==1 then
*{Ri := 0, mem[L] := Ri, jump *+2}*

address for this instruction

- Critical section

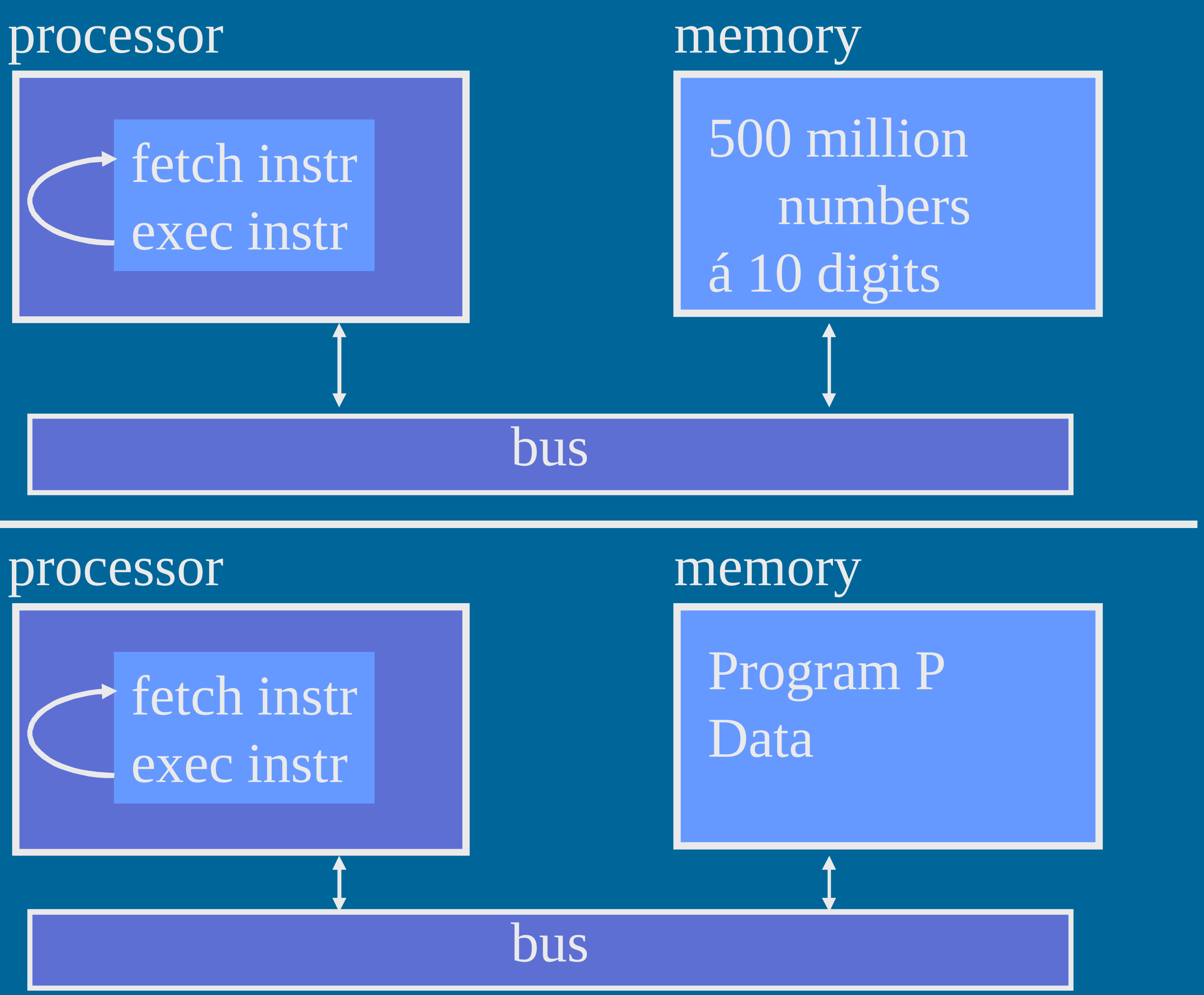
```
LOOP: TAS     R1, L      # L:  1 (open)   0 (locked)
      JUMP    LOOP       # wait until lock open
      ...                # lock is locked for me
      critical section: one process at a time
      ...
      LOAD    R1,=1      # open lock L
      STORE   R1,L
```

- Will it work, if interrupt occurs at ”bad spot”?
  - What is a “bad spot”?

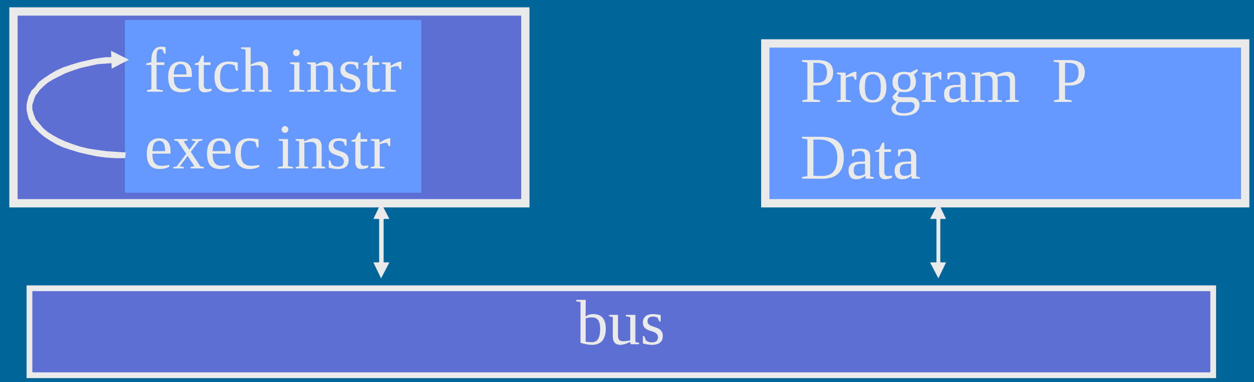
An Introduction to Specification and Verification, 4 cr

- 4^{th} year students
  - Elective graduate level (M.Sc.) course
- Prerequisites
  - Understanding the problematics of distribution and concurrency
  - Introduction to Data Communication, Concurrent Programming
- Model processes with transitional systems
  - step: machine instruction? Method? Transaction? Program?
- Principles of automatic verification
- Verification of simple protocols
- More?
  - Semantics of Programs, 6 cr (lectured 1999)
  - Automatic Verification, 6 cr (lectured 2002)

Foundation for Computational Theory (2)



Computational Theory … (5)



Memory contents
before P's execution:

X = very large integer
(500M digits?)

Memory contents
after P's execution:

Y = some other
very large integer

P is integer valued function P: □→□

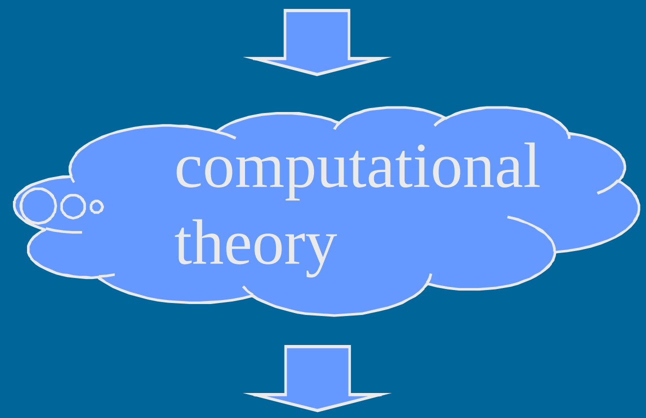
P: N → N

Program P representation in memory: large integer, P∈□

P ∈ N

Computational Theory … (5)

- Properties of any programs can be deduced from properties of integers or integer valued functions



- Proven properties of programs (<u>any</u> programs)
  - valid for <u>all</u> computers
  - valid <u>always</u>: now and in future

Proven theorems in computational theory and algorithm analysis (4)

- With any preselected time span or memory size, there exists a problem such that
  - (1) it has a solution, and
  - (2) all programs solving it will take more time or space than those preselected maximum limits
- There exists programs that can never be solved with any computer
- There exists a large class of know problems such that that we do not yet know how difficult they really are

$$P \stackrel{?}{=} NP$$

http://www.retroweb.com/apollo_retrospective.html

--

End of Lecture 12 and End of Course

--

http://study.for.exam.edu/intime.html