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 system 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

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.

Example 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.

Example on the following slides:
Assembly language programming

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

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

variables, constants, arrays, records in memory, in registers?

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

Instruction fetch-execute cycle

get instr  
PC++

Fetch  
decode instr, calc effect addr

M=0  
read from memory

M>0  
push to memory

M=2  
pushr to memory

M=1  
write to memory

check for interrupts

D=0  
execute instr

Execute  
write from memory

M=2  
push all registers

M=1  
pushr all registers

M=0  
return all registers

Process, process States and Life Time

create  
ready-to-run

running

waiting

completed or killed

When will state change?  
What happens in state change (at instr. level)?  
Who or what causes the state change?
**Processes in Queues, PCB**

- **R-to-R**: 1056 → 1766 ↔ **Running**: 0188
- **Waiting**:
  - Disk: 0036 → 7654 → 9878
  - Timer: 0555
  - Msg from 1345: 2222 → Process 9878 descriptor (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**

- **Memory** → **CPU**
- User process
- Device controller (OS process)
- Direct I/O
- Indirect I/O
- DMA I/O

**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**

- **Compile** From HLL:
  - Compilation unit
  - HLL program or module
  - Symbolic address
- **Link** with other and with library modules:
  - Object module
  - Compiled program (machine code)
  - Linear addresses (per module)
- **Load** into memory as process:
  - Executable
  - Linear addresses (one addr space)
  - some missing (?)
- **Process**
  - Executable program
  - Linear addresses (virt. addr space)
  - PCB (prog)

**Interpretation and Emulation**

- **Java program**
  - Java byte code compilation
  - JVM
  - Java byte-code
  - Pentium II processor
  - (example)
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?

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
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

Transport layer

Transport protocols

Application protocols

Network layer

Network protocols

Ethernet, token ring, PPP

Introduction to Data Communication

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?
- Process communication
  - Shared memory? Messages? Why?
  - Over the network?
- More?
  - Distributed Systems, 4 cr

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

- TAS Ri, L
  (ttk-91 extension)
- Critical section

Address for this instruction

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

An Introduction to Specification and Verification, 4 cr

- 4th 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
- Principles of automatic verification
- Verification of simple protocols
- More?
  - Semantics of Programs, 6 cr (lectured 1999)
  - Automatic Verification, 6 cr (lectured 2002)
Computational Theory ...

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

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

Proven theorems in computational theory and algorithm analysis

- 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 known problems such that we do not yet know how difficult they really are
  
P = NP

---

End of Lecture 12 and End of Course

--

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

http://www.nmtcs.edu/apollo_retrospective.html