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

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.
Assembly language programming

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

…
```

```plaintext
I DC 0
```

```plaintext
LOAD R1, =20
STORE R1, I
```

```plaintext
Loop
```

```plaintext
LOAD R2, =0
LOAD R1, I
STORE R2, T(R1)
```

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

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

variables, constants, arrays (2D), records
in memory, in registers?

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

---

Activation record (Activation record stack)

• 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

```plaintext
int funcA (int x, y);
```

```plaintext
return val
param x
param y
old PC
old FP
local var i1
local var i2
old R1
old R2
```

---

Lecture 12, Summary
Instruction fetch-execute cycle

Fetch
- get instr
- PC++
- M=2
- read from memory
- M>0
- exec. instr
- M=0
- decode instr, calc effect addr
- write to memory
- pushr
- check for interrupts
- D=0 (SR)
- write
- push
- pushr

Execute
- pushr
- push
- M=2
- write
- M=0
- exec. instr
- M>0
- decode instr, calc effect addr
- read from memory
- M=2
- check for interrupts
- D=0 (SR)
- write
- pushr
- push
- M=0
- exec. instr
- M>0
- decode instr, calc effect addr
- read from memory

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

<table>
<thead>
<tr>
<th>+</th>
<th>“15”</th>
<th>“0.1875” = “0.0011”</th>
</tr>
</thead>
<tbody>
<tr>
<td>sign</td>
<td>exponent</td>
<td>mantissa or significand</td>
</tr>
</tbody>
</table>

1/8 = 0.1250
1/16 = 0.0625
0.1875

1) Binary point (.) is immediately after the first bit
2) Mantissa is normalized: leftmost bit is 1
3) Leftmost (most significant) bit (1) is not stored (implied bit)

1/8 = 0.1250
1/16 = 0.0625
0.1875

mantissa exponent
0.0011 “15”

24 bit mantissa!

Process, process States and Life Time

- ready-to-run
- running
- waiting
- completed or killed
- create
- waiting

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

R-to-R → 1056 → 1766 ← Running ← 0188

waiting

Disk1 → 0036 → 7654 → 9878
Timer → 0555
Msg from 1345 → 2222

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

bus

device

device controller

data c/s

Device driver
(OS process)

User 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

<table>
<thead>
<tr>
<th>Compile</th>
<th>Link with other and with library modules</th>
<th>Load Into memory as process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compile From HLL</td>
<td>Object module</td>
<td>Executable</td>
</tr>
<tr>
<td>Compilation unit</td>
<td>Compiled program (machine code)</td>
<td>Linear addresses (one addr space)</td>
</tr>
<tr>
<td>HLL program or module</td>
<td>Linear addresses (per module)</td>
<td>some missing (?)</td>
</tr>
<tr>
<td>symbolic address</td>
<td>Compiled program (machine code)</td>
<td>Linear addresses (virt. addr space)</td>
</tr>
<tr>
<td>prog.c</td>
<td>math.l</td>
<td>PCB (prog)</td>
</tr>
<tr>
<td>prog.o</td>
<td>myprog.obj</td>
<td>myprog.exe</td>
</tr>
</tbody>
</table>
Interpretation and Emulation

k = i+j; Java program
Java byte code compilation
Java byte-code

JVM
Java virtual machine

iload i
iload j
iadd
istore k

load

native environment

Pentium II processor

Java interpreter

Pentium II processor

Java byte code

JIT compiler

dl load module

Java processor

Pentium II processor

22.4.2010 Teemu Kerola, Copyright 2010
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?

![Figure 11.11 Timing Diagram for Instruction Pipeline Operation](image)

![Figure 11.16 Branch Prediction State Diagram](image)
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

- Application
  - Application protocols
  - Transport protocols
  - Network protocols
  - Transfer protocols
  - Ethernet, token ring, PPP
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)
  \[
  \begin{align*}
  Ri := & \text{mem}[L] \\
  \text{if } Ri &= 1 \text{ then} \\
  \{Ri := 0, \text{mem}[L] := Ri, \text{jump } *+2\}
  \end{align*}
  \]
- Critical section

  \[
  \begin{align*}
  \text{LOAD} & \ R1,=1 \\
  \text{STORE} & \ R1,L
  \end{align*}
  \]

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

semaphores
monitors
rendezvous
guarded statements
rpc, messages
Java concurrent progr.
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)
Foundation for Computational Theory

Processor

Memory

Bus

500 million numbers \(\approx 10\) digits

Program P

Data

Memory contents before P’s execution:

Memory contents after P’s execution:

\(X = \) very large integer (500M digits?)

\(Y = \) some other very large integer

P is integer valued function \(P: \mathbb{N} \rightarrow \mathbb{N}\)

Program P representation in memory: large integer, \(P \in \mathbb{N}\)
Computational Theory … (5)

- 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 (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 we do not yet know how difficult they really are

\[ P \neq NP \]
End of Lecture 12 and End of Course

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

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