Overview Content

- Structure
- OS view point
- Buses
- I/O-controller and memory-mapped I/O
- Memory hierarchy

I/O layers
- Privileged mode
- Instruction cycle
- Interrupt handling

Goal:
- Remember what has already been covered on Comp. Org I

Structure of a computer

Hardware vs Software

Control, Processing, Storage, Data movement

Operating System's view point

Applications
Shell
System programs

System calls
Resource management
Protection
File system
Memory management

I/O management
Interrupt handling

Devices and device drivers

Buses
- Local (Sisäinen), System, I/O expansion
- Device controllers (Laiteohjaimet, NOTE: Sta10: "I/O module"

I/O controller and memory-mapped I/O

- Device driver (ajuri) controls the device via controller's registers
- Driver refers to these registers as regular memory locations
- Common memory references, like in load/store-instructions
- Controller (ohjain) detects its own memory addresses on the bus
- Device controller 'intelligent' memory location
Memory hierarchy
- Access time (saantiaika) (not?) dependent of the location
- Registers, cache, main memory
- Block buffering (lohkopuskurointi) (OS functionality)
- Magnetic and optical storage devices
- File servers (tiedostopalvelut)
- Network Attached Storage (NAS) - files
- Storage Area Network (SAN) - blocks

Which new common technology is missing from picture?

CPU execution modes
- Instruction privileges
  - Privileged (etuoikeutettu) and normal
  - User mode (käyttäjätila)
  - Kernel mode (etuöikeutettu tila)
- Memory protection
- User mode (käyttäjätila)
- Kernel mode (etuöikeutettu tila)

Mode change
- User mode, normal mode → kernel mode, privileged mode
  - Interrupt or special SVC instructions (service request)
  - Kernel mode → User mode
  - Privileged instruction, for example IRET (return from interrupt, interrupt return)
  - Returns the control and mode as they were before the mode change
    - Very similar with return from a subroutine

Layers of the I/O system

Teemu’s cheese cake
- Register, on-chip cache, memory, disk, and tape speeds relative to times locating cheese for the cheese cake you are baking...

Europa (Jupiter)

Which now common technology is missing from picture?

Direct I/O
- Device driver (laitajärjestelmä)
- Input of a block of data

Device driver
- Normalization of I/O interrupts
- Normalize to CPU address
- Execute I/O instructions
- Notify waiters
- Fetch device status
- Get status
- Clear status
- Perform I/O operation
- Notify waiters

DMA
- Direct Memory Access
- Transfer data
- Fetch DMA status
- Execute DMA command
- Notify waiters

DMA
- Input of a block of data
- Device driver (laitajärjestelmä)
Review Questions

- Main parts of a computing system?
- DMA: principles and functionalities?
- Obligatory hardware and its features?
- How to make CPU to execute normal user program? Operating system?

Digital Logic

Stallings:
Online Chapters 20.1-3
Boolean Algebra
Simplification
Gates
Combination Circuits

Boolean Algebra

- George Boole
  - ideas 1854
  - Claude Shannon (MSc thesis: "Logic"
  - apply to circuit design, 1938
  - "father of information theory"

Topics:
- Describe digital circuitry function
  - programming language?
- Optimise given circuitry
  - use algebra (Boolean algebra) to manipulate (Boolean) expressions into simpler expressions

Boolean Algebra

- Variables: A, B, C
- Values: TRUE (1), FALSE (0)
- Basic logical operations:
  - binary: AND (·) $A \cdot B = AB$
  - OR (+) $B + C = B + C$
  - unary: NOT (¯) $\overline{A}$
- Composite operations, equations
  - precedence: NOT, AND, OR
  - parenthesis
  - $D = A + \overline{B} \cdot C = A + (\overline{B}C)$

Digital Logic

Hardware
- Decentralised or other system hardware
- memory
- Process address execution of current instruction
- Process signals advancement of interrupt
- Process signals enable PBO and PC state control
- Return processor state information
- Return old PBO and PC

Software
- Take decisions of process state information
- Process interrupt
- Enter privilege mode vs. user mode
- Interrupt disabling vs. enabling
- Scheduling, dispatching
- Enter mode
- Enable interrupts?
**Boolean Algebra**

- Other operations
  - XOR (exclusive-or)
  - NAND
  - NOR

- Truth tables
  - What is the result of the operation?

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>NOT P</th>
<th>P AND Q</th>
<th>P OR Q</th>
<th>P NAND Q</th>
<th>P NOR Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Postulates and Identities**

- How can I manipulate expressions?
  - Simple set of rules?

| Boolean Postulates | Complementary Laws
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A + \overline{A} = 1</td>
<td>A \cdot \overline{A} = 0</td>
</tr>
<tr>
<td>A + B = B + A</td>
<td>Associative Laws</td>
</tr>
<tr>
<td>A + (B + C) = (A + B) + C</td>
<td>(A \cdot B) \cdot C = A \cdot (B \cdot C)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Identities</th>
<th>Earth Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 \cdot A = 0</td>
<td>A \cdot \overline{A} = 0</td>
</tr>
<tr>
<td>1 + A = 1</td>
<td>A + \overline{A} = 1</td>
</tr>
</tbody>
</table>

**Gates, circuits, combination circuits**

- Implement basic Boolean algebra operations
- Gates - fundamental building blocks
  - 1 or 2 inputs, 1 output
- Combine to build more complex circuits
  - memory, adder, multiplier, …
- Gate delay in combination circuits
- Change inputs, after (combined) gate delay new output available
  - 1 ns? 10 ns? 0.1 ns?

**Describing the Circuit**

- a) Boolean equations
- b) Truth table
- c) Graphical symbols (next slide)

**Graphical Symbols, Sum-of-Products, Product-of-sums**

- Sum-of-Products
- Product-of-sums

**Simplification of Circuits**

- Algebraic Simplification
- Simplification with Karnaugh Maps
  - E.g., see Kerola slides 2003/appa
Using Karnaugh Maps to Minimize Boolean Functions

Original function

Canonical form (now already there)

Karnaugh Map

Find smallest number of circles, each with largest number \(2^i\) of 1’s

Select parameter combinations corresponding to the circles

Get reduced function

Discussion?

Multiplexers

- Select one of many possible inputs to output
- truth table
- implementation

Each input/output “line” can be many parallel lines
- select one of three 16 bit values
  - CPU, IR, ALU
- simple extension to one line selection
  - lots of wires, plenty of gates …
- Used to control signal and data routing
  - Example: loading the value of PC

Encoders/Decoders

- Exactly one of many Encoder input or Decoder output lines is 1
- Encode that line number as output
  - hopefully less pins (wires) needed this way
  - optimise for space, not for time
- Example:
  - encode 8 input wires with 3 output pins
  - route 3 wires around the board
  - decode 3 wires back to 8 wires at target

Decoder

Read-Only-Memory (ROM)

- Given input values (address), get output value (contents)
- Like multiplexer, but with fixed data

4-bit deep data

4-bit output

Select lines "address"
Adders

- 1-bit adder
  \[ A=1 \quad B=0 \quad \text{Carry}=0 \quad \text{Sum}=1 \]

- 1-bit adder with carry
  \[ \text{Carry}=1 \quad A=1 \quad B=0 \quad \text{Carry}=1 \quad \text{Sum}=0 \]

Implementation

- Build a 4-bit adder from four 1-bit adders

Simple processor

http://www.gamezero.com/team-0/articles/math_magic/microstage4.html