11. System-Level Computer Architecture

- Information Representation
- Exploration of Computing System Hardware
- Some Examples
- Special-Purpose Processors and Devices
11. System-Level Computer Architecture

- Information Representation
- Exploration of Computing System Hardware
- Some Examples
- Special-Purpose Processors and Devices
Integers

Binary Representation of Non-Negative Integers
- Arithmetic modulo $2^b$ ($3 \leq b \leq 64$ in general)
- Example: $0000\ 0111\ 1011\ 1000_2 = 0x07d8 = 2008$

Binary Representation of Relative Integers
- Question: how to represent negative numbers?
Integers

Binary Representation of Non-Negative Integers
- Arithmetic modulo $2^b$ ($3 \leq b \leq 64$ in general)
- Example: $0000\ 0111\ 1011\ 1000_2 = 0x07d8 = 2008$

Binary Representation of Relative Integers
- Question: how to represent negative numbers?
- First idea: *one's complement*
  - $0 \leq i \leq 2^{b-1} - 1$: binary representation of $i$
    (most significant bit unset)
  - $-2^{b-1} - 1 \leq i \leq 0$: binary representation of $2^{b-1} - i$
    (most significant bit set)
- Caveats: there are two zeroes, case-based algorithm for addition
Integers

Binary Representation of Non-Negative Integers
- Arithmetic modulo \(2^b\) (\(3 \leq b \leq 64\) in general)
- Example: \(0000\ 0111\ 1011\ 1000_2 = 0x07d8 = 2008\)

Binary Representation of Relative Integers
- Question: how to represent negative numbers?
- Much better: *two’s complement*
  - \(0 \leq i \leq 2^{b-1} - 1\): binary representation of \(i\) (most significant bit unset)
  - \(-2^{b-1} \leq i < 0\): binary representation of \(2^b + i\) (most significant bit set)
- Important property:
  \[\forall i \in \{-2^{b-1}, \ldots, 2^{b-1} - 1\},\]
  \[-i \equiv (2^b - 1) + 1 - i \equiv \bar{i}_2 + 1 \mod 2^b\]
Rational Numbers

Fix Point Numbers

- Bounded mantissa subset of $\mathbb{Q}$ with fix exponent
- Mantissa represented in two’s complement
- The exponent (position of the .) is set implicitly w.r.t. variation intervals and analysis of roundoff errors acceptable for a given algorithm and application domain
## Rational Numbers

### Floating Point Numbers: IEEE754

- Bounded *mantissa* subset of $\mathbb{Q}$ with *variable, bounded exponent*
- Separate sign and mantissa (one’s complement)
- Normalized numbers of the form $\pm 0.mantissa \times 10^{\text{exponent-bias}}$
  - 32-bit *float*: 1 sign bit, 8 exponent bits, 23 mantissa bits
  - 64-bit *double*: 1 sign bit, 11 exponent bits, 52 mantissa bits
  - 80-bit *extended*: 1 sign bit, 15 exponent bits, 64 mantissa bits
- Support for denormalized numbers and custom rounding modes
- Special cases: $\pm \infty$ and NaN (not-a-number)
- Reminder: exact representation for rational numbers whose denominator is a power of two only
**Endianness: Representation of Integers in Memory**

**Example:** $0x07d8 = 2008$

<table>
<thead>
<tr>
<th>Address base</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big endian</td>
<td>$0x07$</td>
<td>$0xd8$</td>
</tr>
<tr>
<td>Little endian</td>
<td>$0xd8$</td>
<td>$0x07$</td>
</tr>
</tbody>
</table>

**Example:** $0x0123456789abcdef = 81985529216486895$

<table>
<thead>
<tr>
<th>Address</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big endian</td>
<td>$0x01$</td>
<td>$0x23$</td>
<td>$0x45$</td>
<td>$0x67$</td>
<td>$0x89$</td>
<td>$0xab$</td>
<td>$0xcd$</td>
<td>$0xef$</td>
</tr>
<tr>
<td>Little endian</td>
<td>$0xef$</td>
<td>$0xcd$</td>
<td>$0xab$</td>
<td>$0x89$</td>
<td>$0x67$</td>
<td>$0x45$</td>
<td>$0x23$</td>
<td>$0x01$</td>
</tr>
</tbody>
</table>
Android Example: Frame Buffer Encoding

Image Frame Alternatives

- Color *palette* vs. *true color* RGB components per pixel
- Bitmap: one 2D-array per bit, same bit from multiple arrays form a pixel (color # or RGB components)
- Pixel array: one word per pixel, word size depends on number of bits per pixel (grayscale or RGB)

Android’s Frame Buffer

- **320 × 480** 2D-array of 16-bit words
- Each word is the little-endian encoding of R, G and B components on 5 bits, plus an unused 16th bit (often reserved for transparency, a.k.a. α-channel)

<table>
<thead>
<tr>
<th></th>
<th>f</th>
<th>e</th>
<th>d</th>
<th>c</th>
<th>b</th>
<th>a</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>In memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>α</td>
<td>00 f8</td>
<td></td>
</tr>
<tr>
<td>Green</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>α</td>
<td>c0 07</td>
<td></td>
</tr>
<tr>
<td>Blue</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>α</td>
<td>3e 00</td>
</tr>
</tbody>
</table>
Portable Data Representation

**eXternal Data Representation (XDR Standard)**

- Standard representation and programmer interface to communicate data structures across systems/devices
- Scalars
- Structured data
  - Pointers: neither portable across processes (inconsistent virtual memory addressing) nor across systems/devices
11. System-Level Computer Architecture

- Information Representation
- Exploration of Computing System Hardware
- Some Examples
- Special-Purpose Processors and Devices
A Bird’s Eye View of a Processor Core

Original Von Neumann Architecture

Drivers for the Evolution of Computer Architecture

- Main driver: performance, through frequency, parallelism and specialization
- Other drivers:
  - Energy efficiency (autonomy, energy cost)
  - Power efficiency (thermal envelope, cooling cost)
  - Predictability and reliability (real time, redundancy)
  - Cost and area
Von Neumann Architecture Pushed to the Extreme

Other examples: Intel Pentium 4; IBM PowerPC 970 (G5), IBM Power 6
What About Embedded Systems?

Processors and Machine Languages for Embedded Devices

- **Application specific (ASIC)**
  - Controller
  - Datapath
  - Finite State Machine (FSM) with datapath
  - Hardware implementation (partial) of a Java Virtual Machine

- **General Purpose**
  - Complex Instruction Set Computing (CISC)
    - Intel x86, Motorola/Freescale 680x0
    - Compact code, specialized instructions, extra complexity/latency
  - Reduced Instruction Set Computing (RISC)
    - MIPS, Sun Sparc, IBM Power/PowerPC, ARM
    - Better pipelining, higher frequency, instruction-level parallelism
  - Instruction Level Parallelism (ILP)
    - Superscalar (in-order or out-of-order execution)
    - Very Long Instruction Word (VLIW)
  - Single Instruction Multiple Data (SIMD)

- Information Representation
- Exploration of Computing System Hardware
- Some Examples
- Special-Purpose Processors and Devices
PC Motherboard: General-Purpose System

Nano ITX format, “fan-less” processor (600MHz Luke CoreFusion from VIA)
System and Memory Busses

Main Motherboard Components

Bus Configuration and Communication

- Bus arbiter
- Bus clock vs. processor and device clocks
- Interrupts: “jumpers”, or dynamic configuration
- Internal vs. external busses
Cache-Coherent Multiprocessor Architectures

Non-Uniform Memory Architecture (NUMA)

Shared-memory parallel processing

But what are the basic principles?

- Bus: clocked data communication circuit, *synchronous* with its connected units/cores/devices, with dedicated *arbitration* controller
  Which core or device is allowed to write on the bus at a given cycle? see INF559

- Networking layer: network-on-chip
  *Routing packets (routing tables), see INF431, INF566*

- Cache coherence: *snoop* or *directory* protocols?
  *How to maintain coherence of the data loaded from caches and stored to memory? see INF559*
Graphics Card: Special-Purpose System

Tradeoffs

- **Computational density** vs. optimization for general-purpose applications
  - No operating system support (no interrupts, privileges, address translation, etc.) or dynamically linked data structures

- **Memory bandwidth** for streaming (input-compute-output) applications (more important than a large cache)

- **Specialization** of data paths and control logic
  - Extreme case: one dedicated circuit per task of the **graphics pipeline**: 3D vertex computation, geometrical shading, texture rendering; or 2D block-transfers, filtering, line-drawing and area filling

Quantitative Impact of Specialization

<table>
<thead>
<tr>
<th></th>
<th>General-purpose core</th>
<th>Graphical processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory bandwidth</td>
<td>13 GB/s</td>
<td>80 GB/s</td>
</tr>
<tr>
<td>Peak performance</td>
<td>48 GFLOPS</td>
<td>330 GFLOPS</td>
</tr>
</tbody>
</table>
Graphics Processing Unit (GPU)

NVidia GeForce 7800 GTX

- SLI Connector
- Single slot cooling
- sVideo TV Out
- DVI x 2
- 16x PCI-Express
- 256MB/256-bit DDR3
  - 600 MHz
  - 8 pieces of 8Mx32
Graphics Processing Unit (GPU)

NVidia GeForce 7800 GTX
Is it Enough?

The Downside of Specialization: Programmability

- “The biggest challenge facing game companies right now is the problem of writing multithreaded code that fully supports the multiple-core architectures of the latest PCs and the next generation game consoles.” — CEO Valve Software

- “If a programming genius like John Carmack (the designer of Doom) can be so befuddled by mysterious issues coming from multithreaded programming, what chance do mere mortals have?” — Jeremy Reimer, game industry expert

General Purpose computations on GPU (GPGPU)

- Towards more and more programmability of GPUs
- Converge with massively parallel, conventional SIMD processors
- New programming models: NVidia CUDA, AMD CTM and Brook, etc.
11. System-Level Computer Architecture

- Information Representation
- Exploration of Computing System Hardware
- Some Examples
- Special-Purpose Processors and Devices
## What About External Devices?

### Low-Level Device Functions

- **Hardware Startup**, initialization of the hardware upon power-on or reset
- **Hardware Shutdown**, configuring hardware into its power-off state
- **Hardware Disable**, allowing software to disable hardware on-the-fly
- **Hardware Enable**, allowing software to enable hardware on-the-fly
- **Hardware Acquire**, allowing software to gain (lock) access to hardware
- **Hardware Release**, allowing software to free (unlock) hardware
- **Hardware Read**, allowing software to read data from hardware
- **Hardware Write**, allowing software to write data to hardware
- **Hardware Install**, allowing software to install new hardware on-the-fly
- **Hardware Uninstall**, allowing software to remove installed hardware on-the-fly
Networking: Driver Layer

Application Software Layer

LAN

- Wireless
  - IEEE802.2 LLC/SNAP
  - IEEE802.11 MAC
  - Bluetooth LMP, L2CAP, Baseband ...

- Wired
  - IEEE 802.2 LLC/SNAP
  - IEEE 802.3 Ethernet
  - ARCnet, FDDI, IEEE 802.5 Token Ring ...

System Software Layer

WAN

- Wireless
  - NS
  - BSSGP
  - PPP
  - RFCOMM

- Wired
  - X.25 PSTN LAPB
  - HDLC
  - SLIP
  - ATM ...

Hardware Layer

Physical Layer
Networking: Hardware Layer
Networking: Physical Layer

Device 1

Layer 1: Physical Layer
Layer 2: Data Link Layer

Transmission Medium

Device 2

Layer 1: Physical Layer
Layer 2: Data Link Layer
Physical Link Example: Serial Link

**Synchronous Link**
- Clock signal: acceptable for short-distance links

**Asynchronous Link**
- Frame communication protocol additional START/STOP bits
Physical Link Example: Serial Link

Universal Asynchronous Receiver-Transmitter (UART)

- Full-duplex transmission, class of micro-controllers compatible with the ancient 8251 PC UART
- Modern UARTs
  - Timers (real-time clocks and counters) and interrupt controller
  - Baud rate generator
  - Micro-controller (very simple processor core)
  - Internal buffer
  - Micro-program (ROM)
  - Direct Memory Access (DMA): finite-state controller for autonomous data transfer
  - Links with lower-level physical devices
- One UART at each side of the asynchronous link

Extensions

- Universal Serial Bus (USB)
- Serial connection over analog lines: modulator-demodulator (Modem)
### Serial vs. Parallel Links

#### Serial
- One bit at a time
- Single data wire if *half-duplex*
- Two data wires if *full-duplex*
- Optional clock wire for *synchronous* serial connections

#### Parallel
- Multiple bits at a time
- One wire per bit
- Optional clock for synchronous parallel connections

### Why Serial is Generally Faster than Parallel?
- Much easier to reduce *clock skew, cross-talk* (proper isolation of wires)
- Allows for much higher frequencies, especially on long-distance links
Physical Link Example: Ethernet

**Ethernet**

- Robert Metcalfe, 1973
  - XEROX PARC, then 3Com founder
  - IEEE 802.3

- *Carrier Sense Multiple Access w/ Collision Detection* (CSMA/CD)
  - Intuition: shared communication medium ("ether")
Physical Link Example: Ethernet

Main Procedure

1. Frame ready for transmission
2. Is medium idle? If not, wait until it becomes ready and wait the interframe gap period (9.6 \mu s in 1 Mbit/s Ethernet)
3. Start transmitting
4. Did a collision occur?
5. If so, go to collision detected procedure
6. Reset retransmission counters and end frame transmission

Collision Detected Procedure

1. Continue transmission until minimum packet time is reached (jam signal) to ensure that all receivers detect the collision
2. Increment retransmission counter
3. Was the maximum number of transmission attempts reached? If so, abort transmission
4. Calculate and wait random backoff period based on number of collisions (exponential backoff if repeated)
5. Re-enter main procedure at stage 1
Tradeoffs

- Simplicity and low cost vs. expandability
- Bandwidth vs. power envelope, cost and predictability

Examples

- \( \text{I}^2\text{C} \) (or SMBUS): designed by Philips for consumer electronics
  - Non-expandable and master-slave bus, half-duplex, synchronous, 8-bit serial, for connections to low-performance on-board devices at less than 3.4 Mbit/s
Tradeoffs

- Simplicity and low cost vs. expandability
- Bandwidth vs. power envelope, cost and predictability

Examples

- PCI: designed by a consortium of HW vendors

  ▶ Expandable and symmetric bus, full-duplex, synchronous, for connection to high-performance on-board or closeby devices (e.g., hard disks, Ethernet and graphics cards) at up to 3.2 GB/s (16× PCI Express)