Lecture 4:

Buses and I/O devices

www.cl.cam.ac.uk/Teaching/2001/OSFounds/

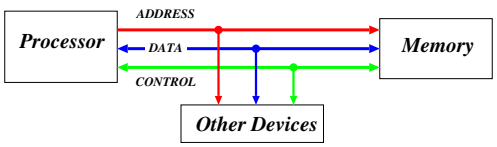
Lecture 4: Friday 12th October 2001

Today's Lecture

Today we'll cover: The rest of the machine!

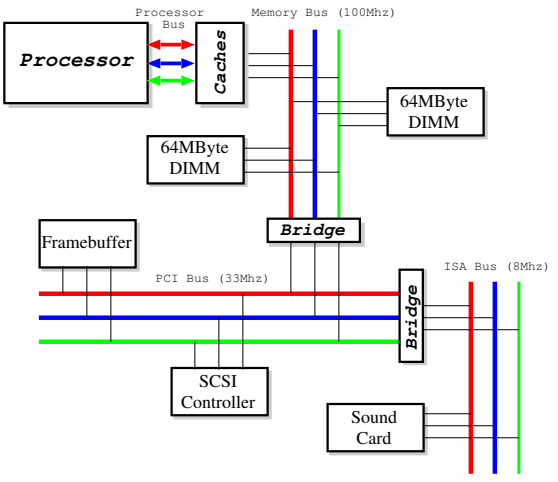
1. Buses
2. I/O devices

Buses



- Bus = collection of *shared* communication wires:
  - ✔ low cost.
  - ✔ versatile / extensible.
  - ✘ potential bottle-neck.
- Typically comprises address lines, data lines and control lines (+ power/ground).
- Operates in a **master-slave** manner, e.g.
  1. master decides to e.g. read some data.
  2. master puts addr onto bus and asserts 'read'
  3. slave reads addr from bus and retrieves data.
  4. slave puts data onto bus.
  5. master reads data from bus.

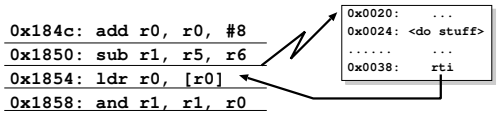
Bus Hierarchy



- In practice, have lots of different buses with different characteristics e.g. data width, max #devices, max length.
- Most buses are **synchronous** (share clock signal).

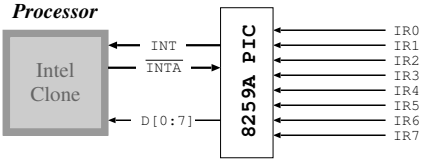
Interrupts

- Bus reads and writes are *transaction* based: CPU requests something and waits until it happens.
- But e.g. reading a block of data from a hard-disk takes $\sim 2ms$, which is $\sim 1,000,000$ clock cycles!
- **Interrupts** provide a way to decouple CPU requests from device responses.
  1. CPU uses bus to make a request (e.g. *writes* some special values to a device).
  2. Device goes off to get info.
  3. Meanwhile CPU continues doing other stuff.
  4. When device finally has information, raises an **interrupt**.
  5. CPU uses bus to read info from device.
- When interrupt occurs, CPU **vectors** to handler, then **resumes** using special instruction, e.g.



Interrupts cont.

- Interrupt lines ($\sim 4-8$) are part of the bus.
- Often only 1 or 2 pins on chip $\Rightarrow$ need to encode.
- e.g. ISA & x86:



1. Device asserts `IRx`.
2. PIC asserts `INT`.
3. When CPU can interrupt, strobes `INTA`.
4. PIC sends interrupt number on `D[0:7]`.
5. CPU uses number to index into a table in memory which holds the addresses of handlers for each interrupt.
6. CPU saves registers and jumps to handler.

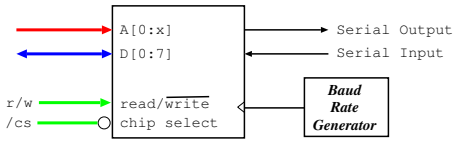
Direct Memory Access (DMA)

- Interrupts good, but even better is a device which can read and write processor memory **directly**.
- A generic DMA "command" might include
  - source address
  - source increment / decrement / do nothing
  - sink address
  - sink increment / decrement / do nothing
  - transfer size
- Get one interrupt at end of data transfer
- DMA channels may be provided by devices themselves:
  - e.g. a disk controller
  - pass disk address, memory address and size
  - give instruction to read or write
- Also get "stand-alone" programmable DMA controllers.

Input/Output Devices

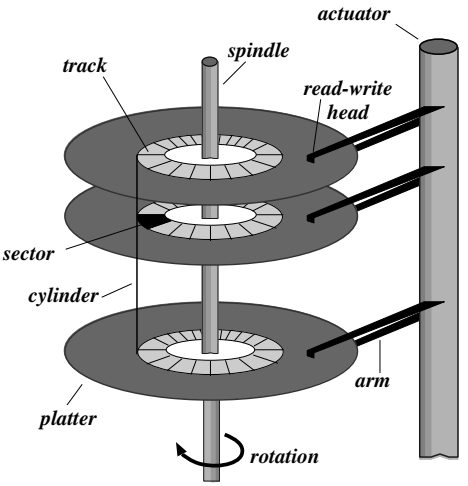
- Devices connected to processor via a bus (e.g. ISA, PCI, AGP).
- Includes a wide range:
  - Mouse,
  - Keyboard,
  - Graphics Card,
  - Sound card,
  - Floppy drive,
  - Hard-Disk,
  - CD-Rom,
  - Network card,
  - Printer,
  - Modem
  - etc.
- Often two or more stages involved (e.g. IDE, SCSI, RS-232, Centronics, etc.)

UARTs



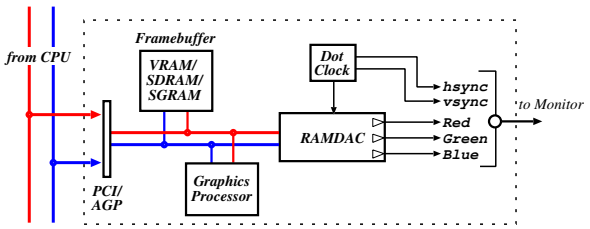
- **U**niversal **A**synchronous **R**eceiver/**T**ransmitter:
  - stores 1 or more bytes internally.
  - converts parallel to serial.
  - outputs according to RS-232.
- Various baud rates (e.g. 1,200 – 115,200)
- Slow and simple. . . and very useful.
- Make up "serial ports" on PC.
- Max throughput $\sim$ 14.4KBytes; variants up to 56K (for modems).

Hard Disks



- Whirling bits of (magnetized) metal. . .
- Rotate 3,600 – 7,200 times a minute.
- Capacity $\sim$ 40 GBytes ($\approx 40 \times 2^{30} bytes$).

Graphics Cards



- Essentially some RAM (framebuffer) and some digital-to-analogue circuitry (RAMDAC).
- RAM holds array of **pixels**: picture elements.
- Resolutions e.g. 640x480, 800x600, 1024x768, 1280x1024, 1600x1200.
- Depths: 8-bit (LUT), 16-bit (RGB 555), 24-bit (RGB 888), 32-bit (RGBA 888).
- Memory requirement $= x \times y \times$ depth, e.g. 1024x768 @ 16bpp needs 1536KB.

$\Rightarrow$ full-screen 50Hz video requires 7.5MBytes/s (or 60Mbits/s).

Summary

You should now understand:

- Buses:
  - Bus hierarchy,
  - **Interrupts & Interrupt vectors**,
  - **Direct Memory Access**.
- I/O devices:
  - Different devices

Next lecture: Operating Systems: The Basics

Background Reading:

- Hennessy/Patterson:
  - Section 8.4—Buses
  - Section 8.3—I/O devices
- Silberschatz et al.:
  - Section 13.2.2—Interrupts
  - Section 13.2.3—DMA

Summary of Part I

- Computers made up of four main parts:
  1. Processor (including register file, control unit and execution unit),
  2. Memory (caches, RAM, ROM),
  3. Devices (disks, graphics cards, etc.), and
  4. Buses (interrupts, DMA).
- Information represented in all sorts of formats:
  - signed & unsigned integers,
  - strings,
  - floating point,
  - data structures,
  - instructions.
- Can (hopefully) understand all of these at some level, but gets pretty complex.

$\Rightarrow$ to be able to actually *use* a computer, need an **operating system**. (Part II!)