The CPU:
real machines

The CPU: real machines - Outline

- **computer architecture**
  - CPU instructions
  - interaction with memory
- **caching: making things seem faster than they are**
- **how chips are made**
- **Moore's law**
- **equivalence of all computers**
  - von Neumann, Turing

Real processors

- **multiple accumulators (called "registers")**
- **more instructions, though basically the same kinds**
  - typical CPU has dozens to few hundreds of instructions in repertoire
- **instructions and data usually occupy multiple memory locations**
  - typically 2 - 8 bytes
- **modern processors have several "cores" that are all CPUs on the same chip**

Typical instructions

- **move data of various kinds and sizes**
  - load a register from value stored in memory
  - store register value into memory
- **arithmetic of various kinds and sizes:**
  - add, subtract, etc., usually operating on registers
- **comparison, branching**
  - select next instruction based on results of computation
    - change the normal sequential flow of instructions
    - normally the CPU just steps through instructions in successive memory locations
- **control rest of computer**

Computer architecture

- **what instructions does the CPU provide?**
  - CPU design involves complicated tradeoffs among functionality, speed, complexity, programmability, power consumption, …
  - Intel and PowerPC are unrelated, totally incompatible
    - Intel: lot more instructions, many of which do complex operations e.g., add two memory locations and store result in a third
    - PowerPC: fewer instructions that do simpler things, but faster e.g., load, add, store to achieve same result
- **how is the CPU connected to the RAM and rest of machine?**
  - memory is the real bottleneck; RAM is slow (60-70 nsec to fetch)
    - modern computers use a hierarchy of memories so that frequently used information is accessible to CPU without going to memory
    - Called **caches**

Computer architecture, continued

- **what tricks do designers play to make it go faster?**
  - overlap fetch, decode, and execute so several instructions are in various stages of completion (pipeline)
  - do several instructions in parallel
  - do instructions out of order to avoid waiting
  - multiple "cores" (CPUs) in one package to compute in parallel
- **speed comparisons are hard, not very meaningful**

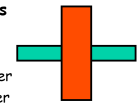
Physical implementation (microprocessors)

Integrated circuits ("chips")

- **Active elements and wires all made at same time out of same materials**
  - Match in size and speed
- **Active elements**
  - Transistors – act as controlled switches
    Logically much like 1940s relays but not physically
  - 1-bit memory elements (volatile)
- **Chips packaged and connected to "pins" that plug in to printed circuit board**

Fabrication: making chips

- **grow layers of conducting and insulating materials on a thin wafer of very pure silicon**
- **each layer has intricate pattern of connections**
  - created by complex sequence of chemical and photographic processes
- **dice wafer into individual chips, put into packages**
  - yield is less than 100%, especially in early stages
- **how does this make a computer?**
  - when conductor on one layer crosses one on lower layer
    voltage on upper layer controls current on lower layer
  - this creates a transistor that acts as off-on switch
    that can control what happens at another transistor
- **wire widths keep getting smaller: more components in given area**
  - today ~0.032 micron = 32 nanometers
    1 micron == 1/1000 of a millimeter (human hair is about 100 microns)
  - eventually this will stop, but has been "10 years from now" for a long time



Moore's Law (1965, Gordon Moore, founder & former CEO of Intel)

- **computing power (roughly, number of transistors on a chip)**
  - doubles about every 18 months
  - and has done so since ~1961
- **consequences**
  - cheaper, faster, smaller, less power consumption per unit
  - ubiquitous computers and computing
- **limits to growth**
  - fabrication plants now cost $2-4B; most are elsewhere
  - line widths are nearing fundamental limits (10 more years?)
  - complexity is increasing
- **maybe some other technology will come along**
  - atomic level; quantum computing
  - optical
  - biological: DNA computing

The bigger picture: universal computing machines

Turing machines

- **Alan Turing *38**
- **showed that a simple model of a computer was universal**
  - now called a Turing machine
  - looks nothing like our microprocessor
- **all computers have the same computational power**
  - i.e., they can compute the same things
  - though they may vary enormously in speed, memory used, etc.
- **equivalence proven / demonstrated by simulation**
  - any machine can simulate any other
  - a "universal Turing machine" can simulate
    any other Turing machine
- **see also**
  - Turing test
  - Turing award

Computing machines wrap-up: Fundamental ideas

- **a computer is a general-purpose machine**
  - executes very simple instructions very quickly
  - controls its own operation according to computed results
- **"von Neumann architecture"**
  - change what it does by putting new instructions in memory
  - instructions and data stored in the same memory
  - indistinguishable except by context
    attributed to von Neumann (1946)
    (and Charles Babbage, in the Analytical Engine (1830's))
  - logical structure largely unchanged for 60+ years
  - physical structures changing very rapidly
- **Turing machines**
  - all computers have exactly the same computational power:
    they can compute exactly the same things; differ only in performance
  - one computer can simulate another computer
    a program can simulate a computer

Additional Important Hardware Ideas

- **Microprocessors have simple instructions that do arithmetic, compare items, select next instruction based on results**
- **bits at the bottom**
  - everything ultimately reduced to representation in bits (binary numbers)
  - groups of bits represent larger entities: numbers of various sizes, letters in various character sets, instructions, memory addresses
  - interpretation of bits depends on context
    one person's instructions are another's data
- **there are many things that we do not know how to represent as bits, nor how to process by computer**