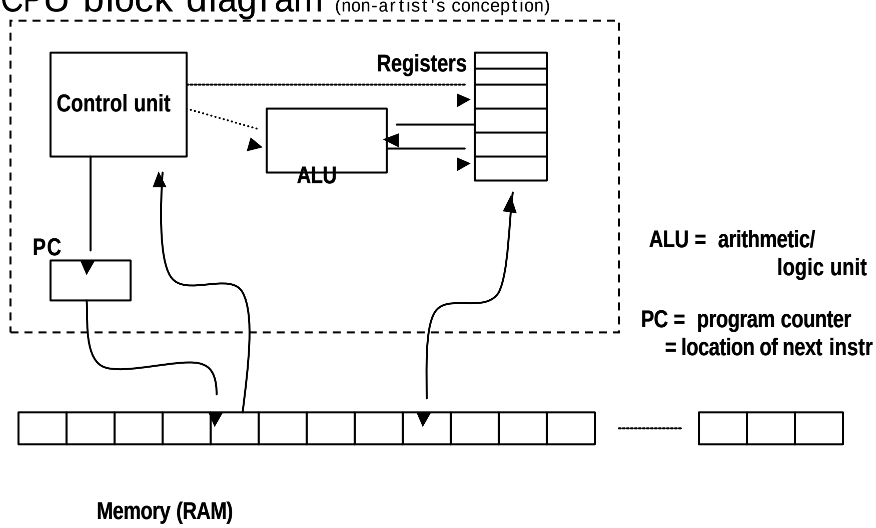
Real machines

- **multiple scratchpads (called "registers")**
- **more instructions, though basically the same kinds**
  - move data
    - load a register from value stored in memory
    - store register value into memory
  - arithmetic:
    - 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 it just steps through instructions in successive memory locations
  - control rest of computer
  - typical CPU has dozens to few hundreds of different instructions
- **instructions and data usually occupy multiple memory locations**
  - typically 2 - 8 bytes
- **real programs are enormous!**

CPU block diagram (non-artist's conception)



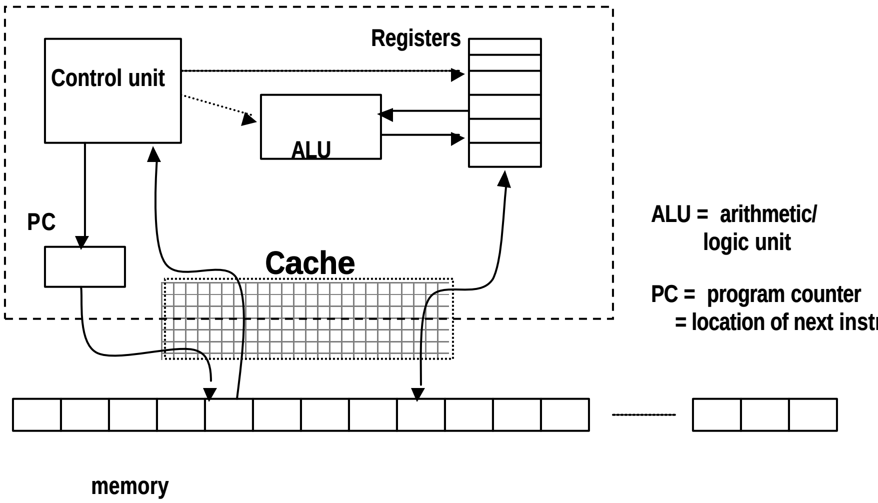
Computer architecture

- **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 memory and rest of machine?**
  - memory is the real bottleneck; memory is slow (70 nsec)
    - modern computers use a hierarchy of memories (caches) so that frequently used information is accessible to CPU without going to memory
- **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
- **speed comparisons are very hard, not terribly meaningful**

Caching: making things seem faster than they are

- **cache: small very fast memory for recently-used information**
  - loads a block of info around the requested info
- **CPU looks in the cache first, before looking in main memory**
- **CPU chip usually includes some cache ("L1" = level 1, ~ 16MB)**
- **CPU chip often includes L2 cache as well**
  - somewhat slower, usually much bigger (e.g., 512KB)
  - may be a separate chip
- **caching works because recently-used info is more likely to be used again soon**
  - therefore more likely to be in the cache already
- **cache usually loads nearby information at the same time**
  - nearby information is more likely to be used soon
  - therefore more likely to be in the cache when needed
- **this kind of caching is invisible to users**
  - except that machine runs faster than it would without

CPU block diagram (non-artist's conception)



Caching is a much more general idea

- **things work more efficiently if what we need is close**
- **if we use something now**
  - will use it again soon (time locality)
  - or will use something nearby soon (space locality)
- **other caches in computers:**
  - CPU registers
  - L1 cache in CPU
  - L2 cache in CPU
  - RAM as a cache for disk or network or …
  - disk as a cache for network
  - network caches as a cache for faraway networks
- **some are automatic, some are controlled by software, some you have some control**

Fundamental ideas

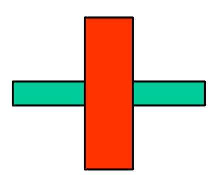
- **von Neumann model**
  - general-purpose machine
    change what it does by putting new instructions in memory
  - instructions and data are in the same memory
  - they are indistinguishable except by context

    attributed to John von Neumann;
    actually Eckert & Mauchly, in ENIAC

- **Turing machines (Alan Turing, *38)**
  - all computers have exactly the same computational power
  - though their performance will vary

Fabrication

- **http://www.intel.com/education/teachtech/learning/chips**
- **grow layers of conducting material on a wafer of very pure silicon**
- **each layer has intricate pattern of connections**
  - defined by chemical processes, mostly etching of unwanted material
- **dice wafer into individual chips, put into packages**
  - yield is less than 100%, especially in early stages
- **how does this make a computer?**
  - voltage on upper layer controls current on lower layer
  - this is a transistor that acts as off-on switch



- **how big? wire thickness today less than 1/10 micron**
  - 1 micron == 1/1000 of a millimeter
  - human hair is about 100 microns

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

- **computing power (roughly, number of transistors on a chip)**
  - doubles every 18 months
  - and has done so since ~1961
  - an aside on the Rule of 72
    - something that compounds at r percent doubles in 72/r periods
    - e.g., if you invest at 10% per year, your money will double in 7.2 years
- **limits to growth**
  - fabrication plants now cost $2-4B
  - line widths are nearing fundamental limits (10 more years?)
  - complexity is increasing
    - Pentium bug
- **maybe some other technology will come along**
  - atomic level; quantum computing
  - optical
  - biological

Wrapup on hardware

- **CPU executes very simple instructions very quickly**
  - can change what it does next according to computed results
- **instructions and data are stored in the same memory**
  - interpretation depends only on context
- **same basic logical structure**
  - all have the same computing capabilities, differ only in performance
  - many different physical structures
  - one machine can simulate another machine
    - a program can simulate a machine
- **Moore's Law: exponential increase in capabilities for 40+ years**
  - cheaper, faster, smaller, less power consumption per unit
  - ubiquitous computers and computing