Instruction sets

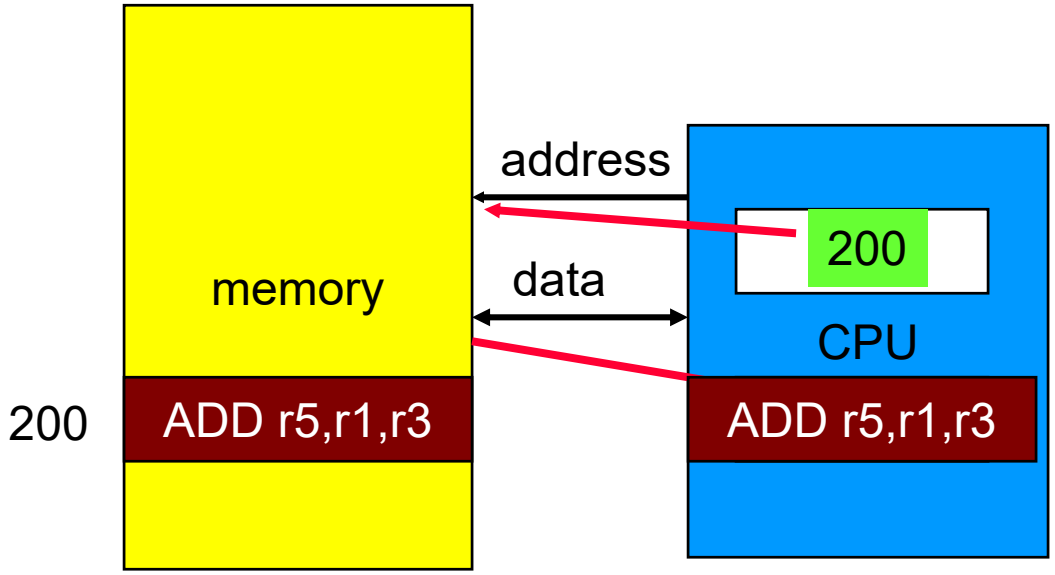
- Computer architecture taxonomy.
- Assembly language.

To command a computer's hardware, you must speak its language. The words of a computer's language are called *instructions*, and its vocabulary is called an **instruction set**.

von Neumann architecture

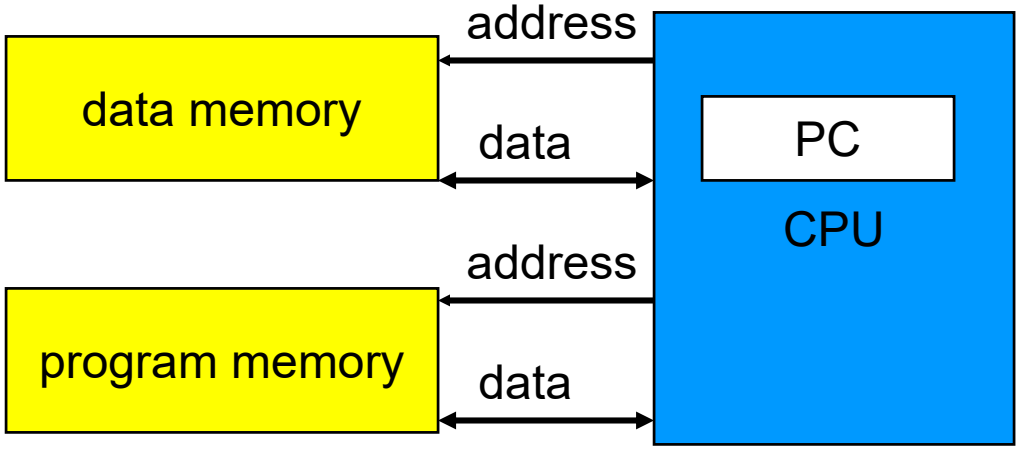
- Memory holds data, instructions.
- Central processing unit (CPU) fetches instructions from memory.
  - Separate CPU and memory distinguishes programmable computer.
- CPU registers help out: program counter (PC), instruction register (IR), general-purpose registers, etc.

CPU + memory



A computer whose memory holds both data and instructions is known as a ***von Neumann*** machine. The CPU fetches the instruction from memory, decodes the instruction, and executes it. The program counter does not directly determine what the machine does next, but only indirectly by pointing to an instruction in memory

Harvard architecture



Harvard machine has separate memories for data and program. The program counter points to program memory, not data memory. As a result, it is harder to write self-modifying programs (programs that write data values, then use those values as instructions) on Harvard machines.

von Neumann vs. Harvard

- Harvard can't use self-modifying code.
- Harvard allows two simultaneous memory fetches.
- Most DSPs use Harvard architecture for streaming data:
  - greater memory bandwidth;
  - more predictable bandwidth.

RISC vs. CISC

- Complex instruction set computer (CISC):
  - many addressing modes;
  - many operations.
- Reduced instruction set computer (RISC):
  - load/store;
  - pipelinable instructions.

Instruction set characteristics

- Fixed vs. variable length.
- Addressing modes.
- Number of operands.
- Types of operands.

Programming model

- Programming model: registers visible to the programmer.
- Some registers are not visible (IR).

Multiple implementations

- Successful architectures have several implementations:
  - varying clock speeds;
  - different bus widths;
  - different cache sizes;
  - etc.

Assembly language

- One-to-one with instructions (more or less).
- Basic features:
  - One instruction per line.
  - Labels provide names for addresses (usually in first column).
  - Instructions often start in later columns.
  - Columns run to end of line.

ARM assembly language example

```
label1      ADR r4,c ; ; get address for c
            LDR r0,[r4] ; get value of c
            ADR r4,d ; get address for d, reusing r4
            LDR r1,[r4] ; get value of d
            SUB r0,r0,r1 ; ; compute c-d
```

Pseudo-ops

- Some assembler directives don't correspond directly to instructions:
  - Define current address.
  - Reserve storage.
  - Constants.

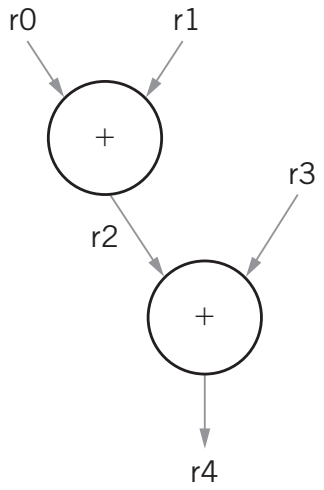
VLIW

- VLIW: very long instruction word.
- Performs several instructions simultaneously.
  - Architecture usually restricts the combination of instructions that can be performed at once.
- Superscalar vs. VLIW:
  - Superscalar runs standard code, determines parallel operations at run time.
  - VLIW determines parallelism at compile time.
- Packet: a set of instructions to be executed in parallel.

Data and control dependencies

```
add r2, r0, r1
add r4, r2, r3
```

Instructions



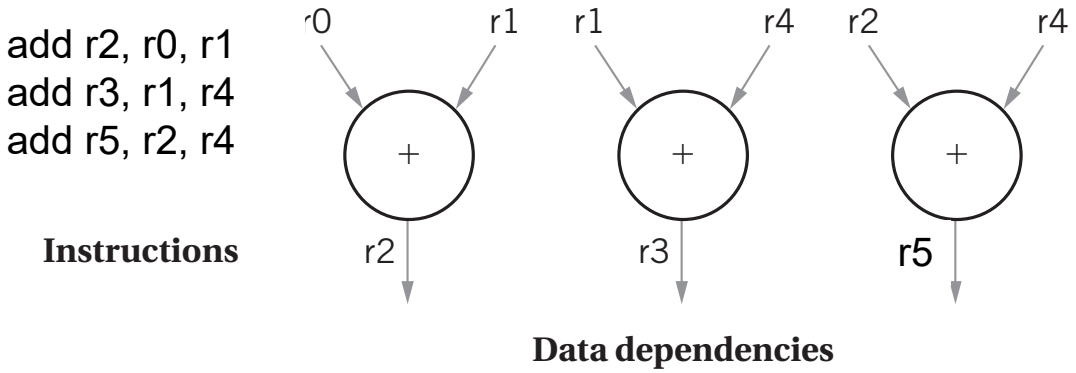
Data dependencies

- r0 + r1 must be performed before r2 + r3.

```
bnz r3, foo
add r0, r1, r2
Foo: …
```

- Add will be executed only if bnz fails.

Data dependencies and parallelism



- All three additions can be performed at the same time.

VLIW and embedded computing

- VLIW is more energy-efficient than superscalar.
- VLIW is widely used in signal processing.
  - Multimedia processing.
  - Processing multiple channels of signals.