Architectures and instruction sets

- Computer architecture taxonomy.
- Assembly language.

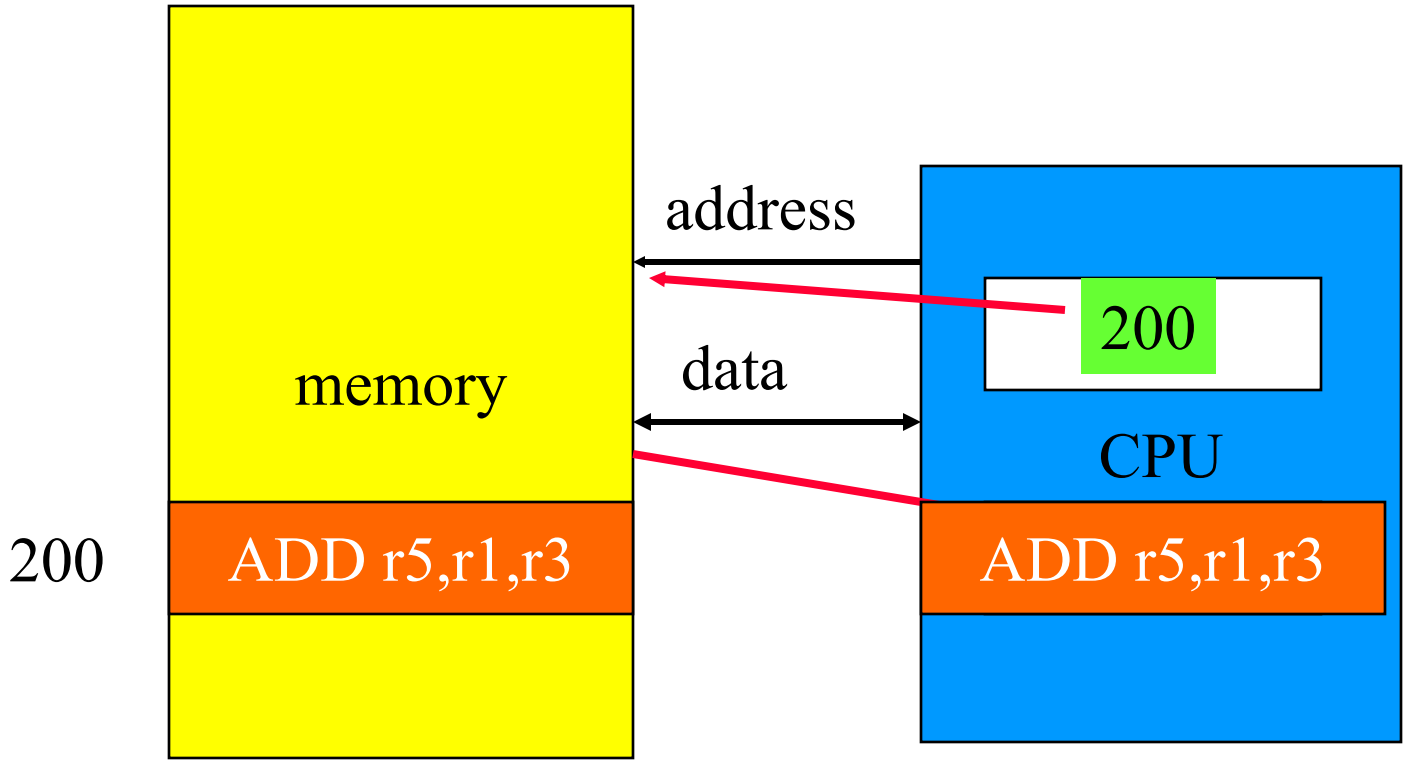
© 2000 Morgan Kaufman

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.

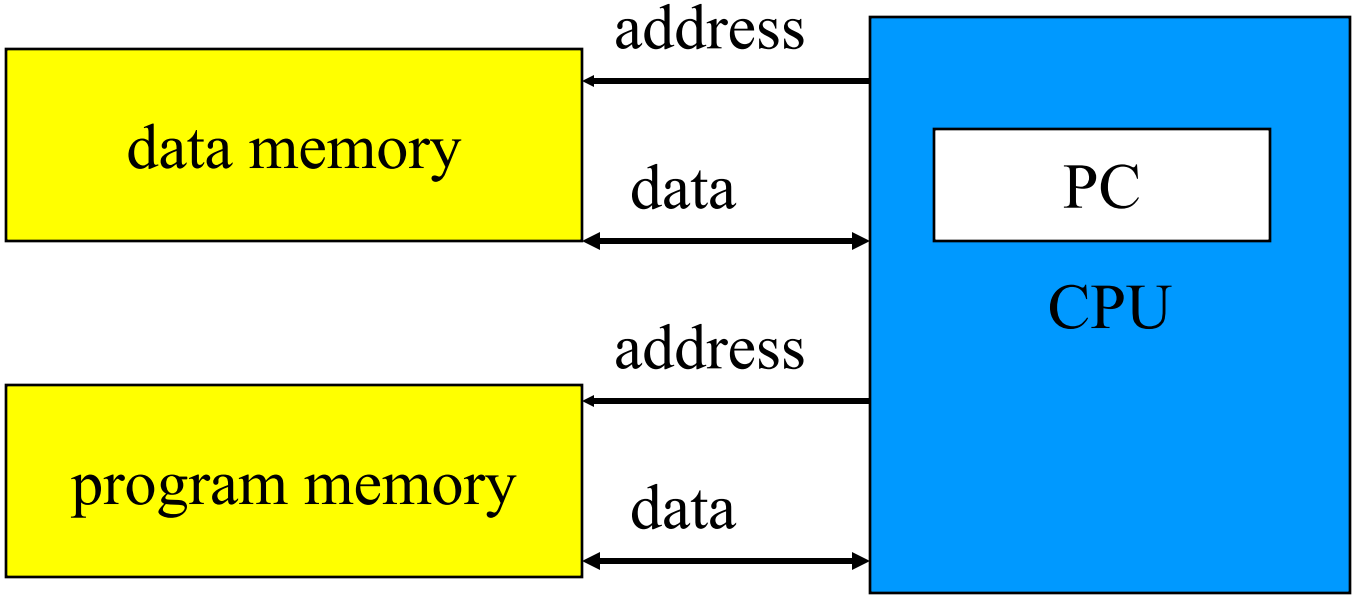
© 2000 Morgan Kaufman

CPU + memory



© 2000 Morgan Kaufman

Harvard architecture



© 2000 Morgan Kaufman

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.

© 2000 Morgan Kaufman

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.

© 2000 Morgan Kaufman

Programming model

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

© 2000 Morgan Kaufman

Multiple implementations

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

© 2000 Morgan Kaufman

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.

© 2000 Morgan Kaufman

ARM assembly language example

```
label1      ADR r4,c
            LDR r0,[r4] ; a comment
            ADR r4,d
            LDR r1,[r4]
            SUB r0,r0,r1 ; comment
```

destination

© 2000 Morgan Kaufman

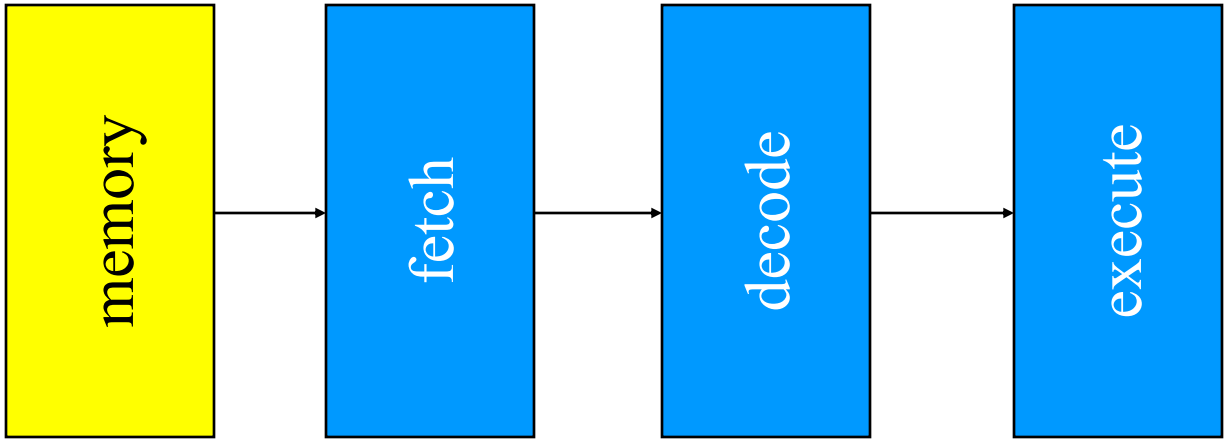
Pseudo-ops

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

© 2000 Morgan Kaufman

Pipelining

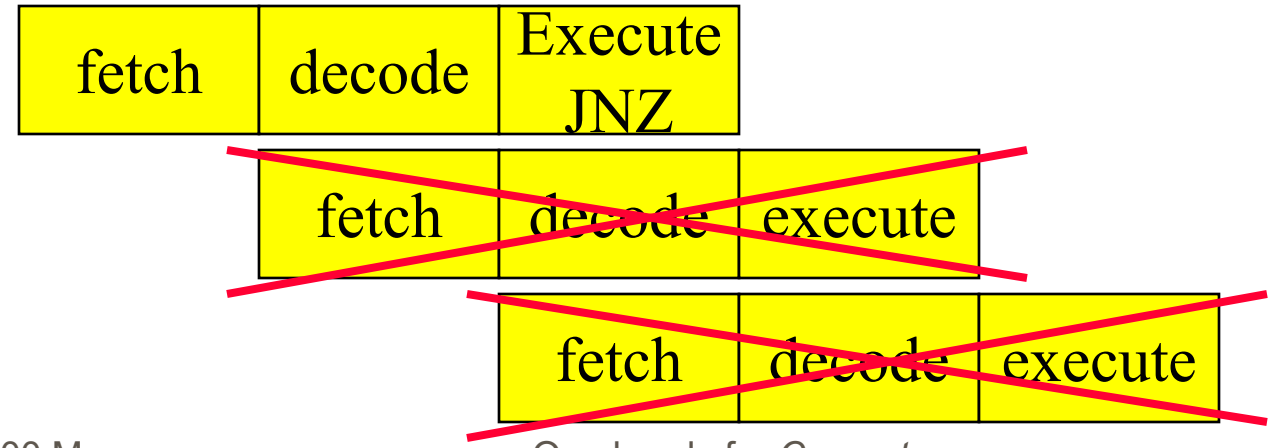
- Execute several instructions simultaneously but at different stages.
- Simple three-stage pipe:



© 2000 Morgan Kaufman

Pipeline complications

- May not always be able to predict the next instruction:
  - Conditional branch.
- Causes bubble in the pipeline:



© 2000 Morgan Kaufman

Superscalar

- RISC pipeline executes one instruction per clock cycle (usually).
- Superscalar machines execute multiple instructions per clock cycle.
  - Faster execution.
  - More variability in execution times.
  - More expensive CPU.

© 2000 Morgan Kaufman

Simple superscalar

- Execute floating point and integer instruction at the same time.
  - Use different registers.
  - Floating point operations use their own hardware unit.
- Must wait for completion when floating point, integer units communicate.

© 2000 Morgan Kaufman

Costs

- Good news---can find parallelism at run time.
  - Bad news---causes variations in execution time.
- Requires a lot of hardware.
  - n^2 instruction unit hardware for n-instruction parallelism.

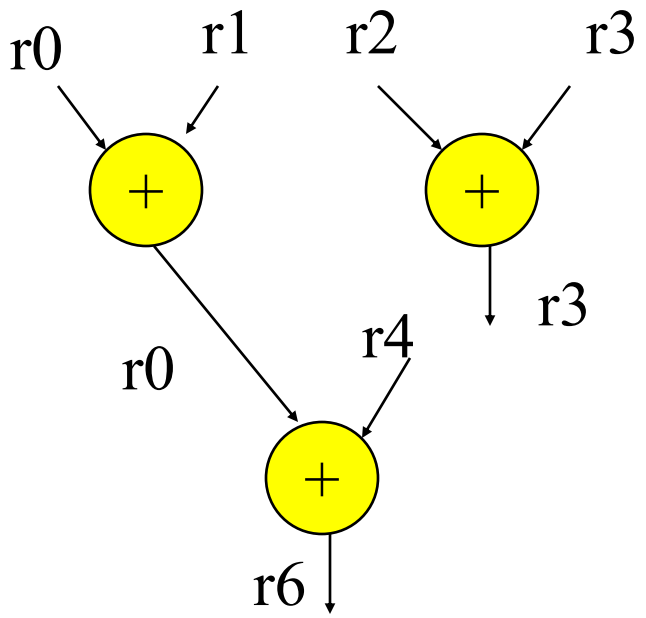
Finding parallelism

- Independent operations can be performed in parallel:

  ADD r0, r0, r1

  ADD r2, r2, r3

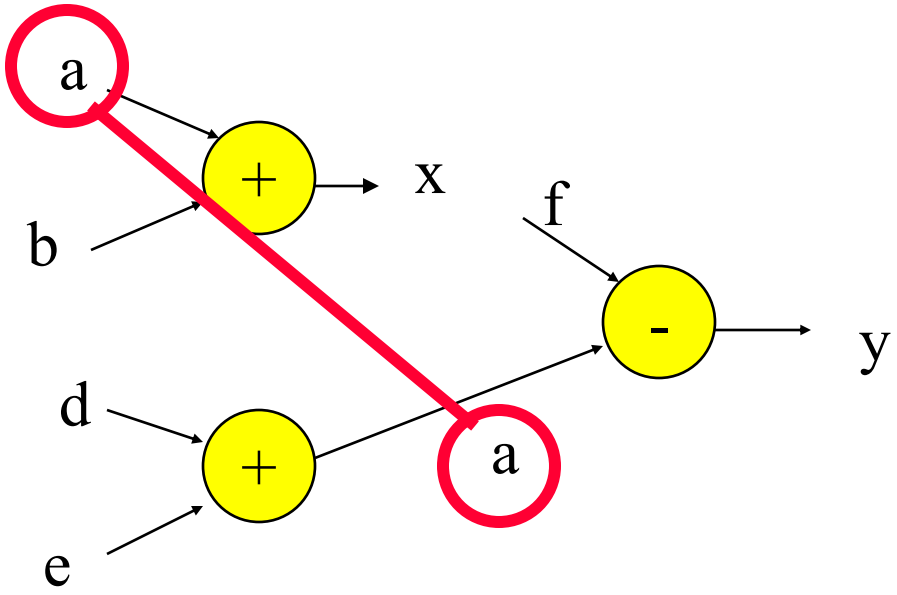
  ADD r6, r4, r0



Pipeline hazards

- Two operations that require the same resource cannot be executed in parallel:

```
x = a + b;
a = d + e;
y = a - f;
```



© 2000 Morgan Kaufman

Scoreboarding

- Scoreboard keeps track of what instructions use what resources:

	Reg file	ALU	FP
instr1	X	X	
instr2			X

Order of execution

- In-order:
  - Machine stops issuing instructions when the next instruction can't be dispatched.
- Out-of-order:
  - Machine will change order of instructions to keep dispatching.
  - Substantially faster but also more complex.

© 2000 Morgan Kaufman

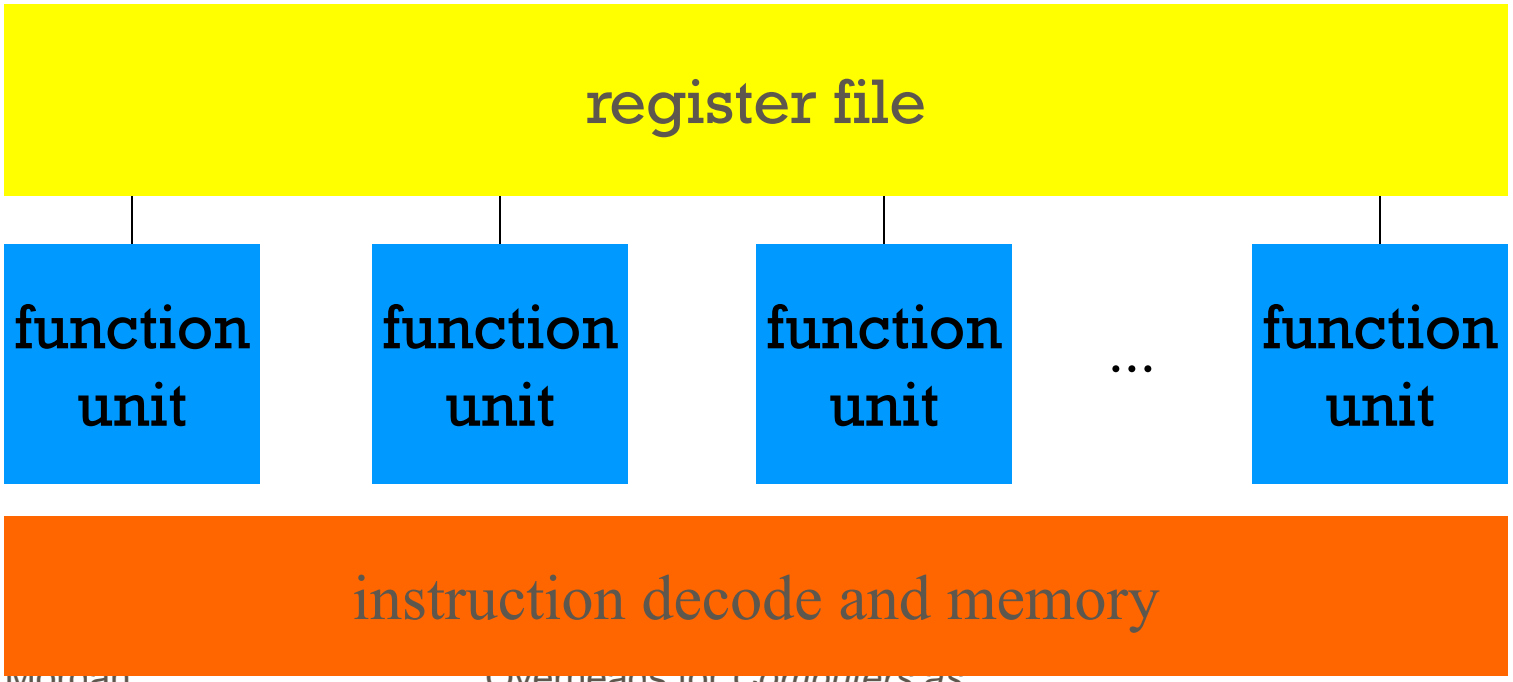
VLIW architectures

- Very long instruction word (VLIW) processing provides significant parallelism.
- Rely on compilers to identify parallelism.

© 2000 Morgan Kaufman

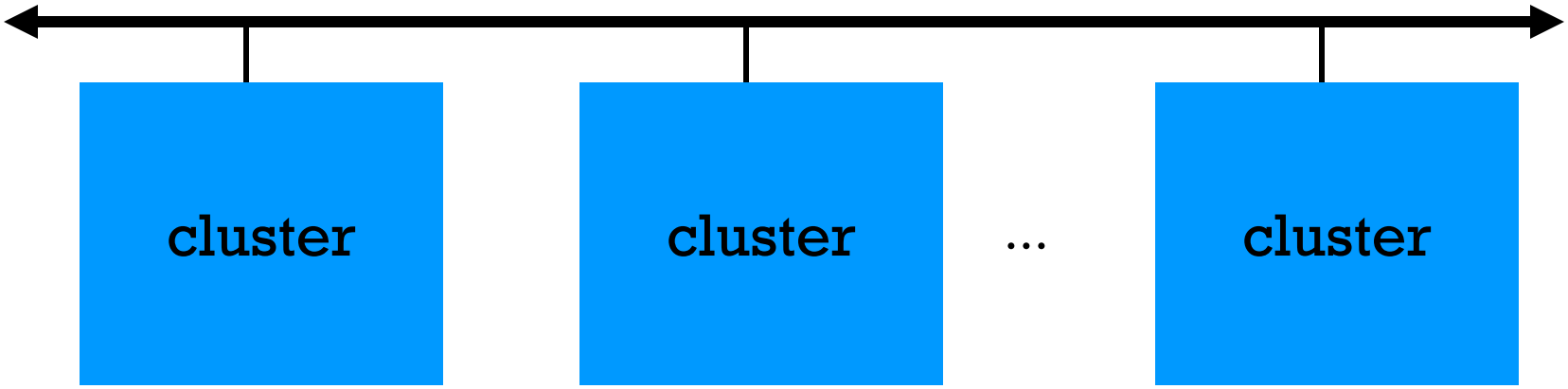
What is VLIW?

- Parallel function units with shared register file:



VLIW cluster

- Organized into clusters to accommodate available register bandwidth:



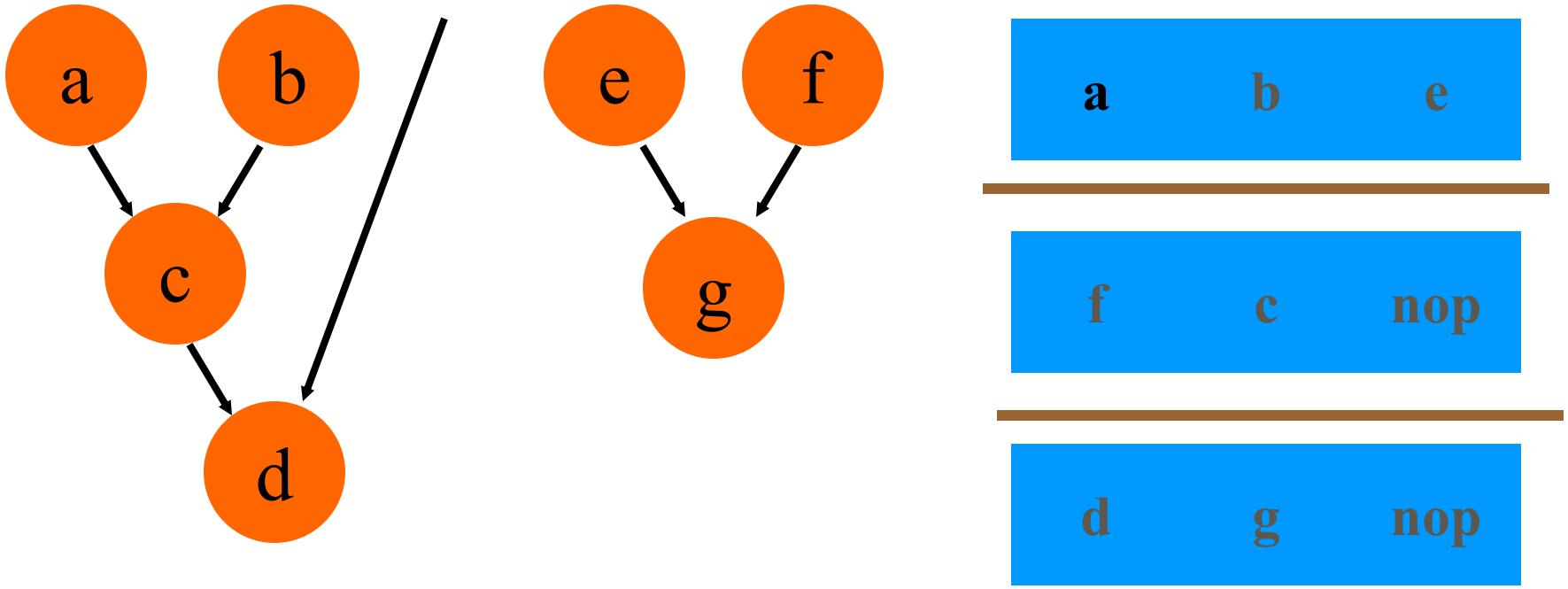
© 2000 Morgan Kaufman

VLIW and compilers

- VLIW requires considerably more sophisticated compiler technology than traditional architectures---must be able to extract parallelism to keep the instructions full.
- Many VLIWs have good compiler support.

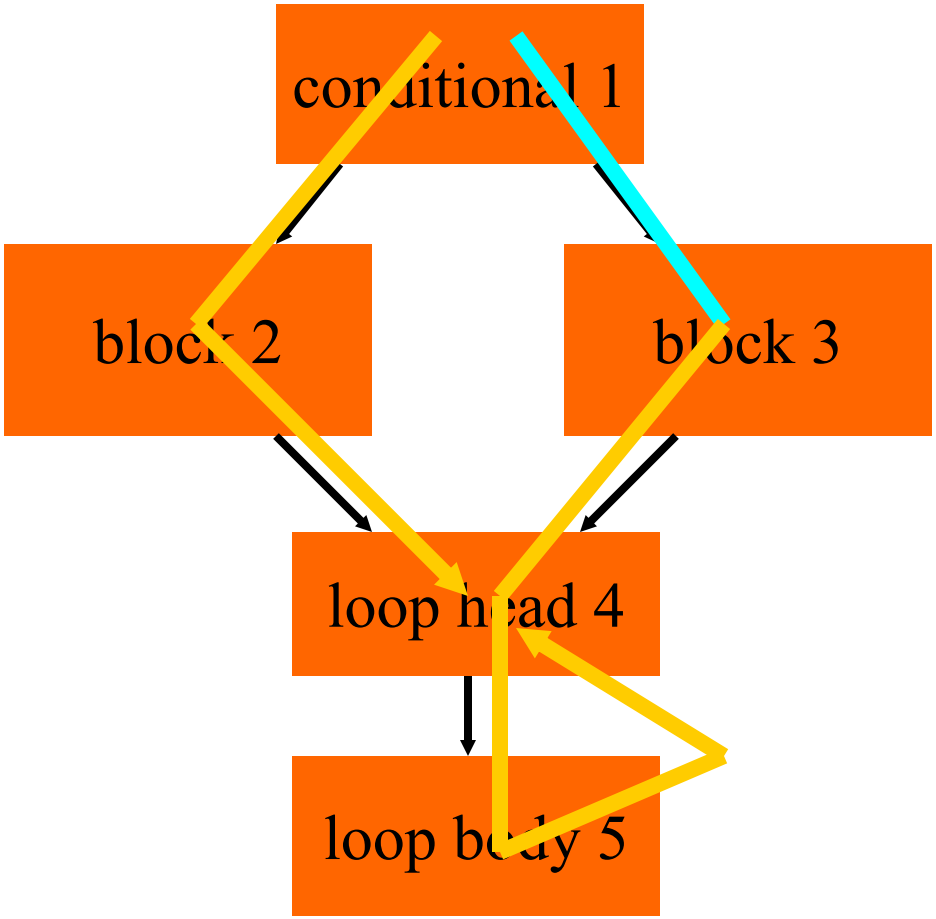
© 2000 Morgan Kaufman

Static scheduling



© 2000 Morgan Kaufman

Trace scheduling



Rank paths in order of frequency.

Schedule paths in order of frequency.

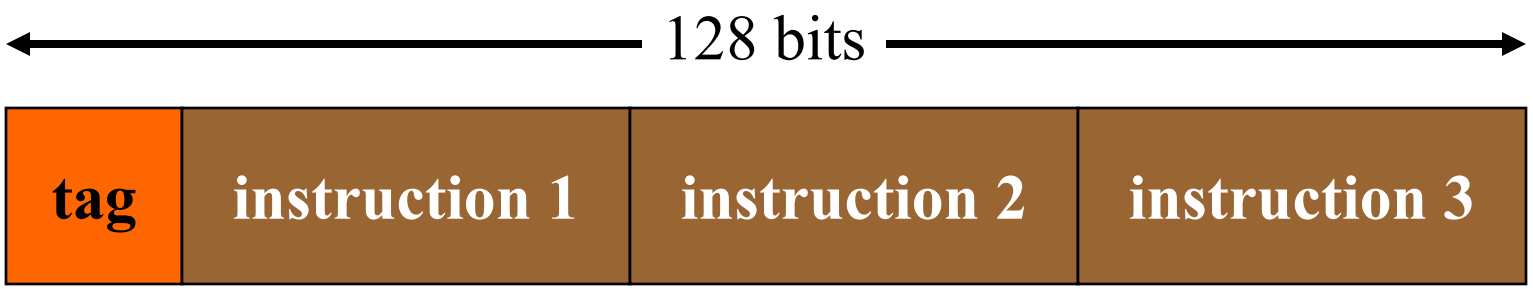
© 2000 Morgan Kaufman

EPIC

- EPIC = Explicitly parallel instruction computing.
- Used in Intel/HP Merced (IA-64) machine.
- Incorporates several features to allow machine to find, exploit increased parallelism.

IA-64 instruction format

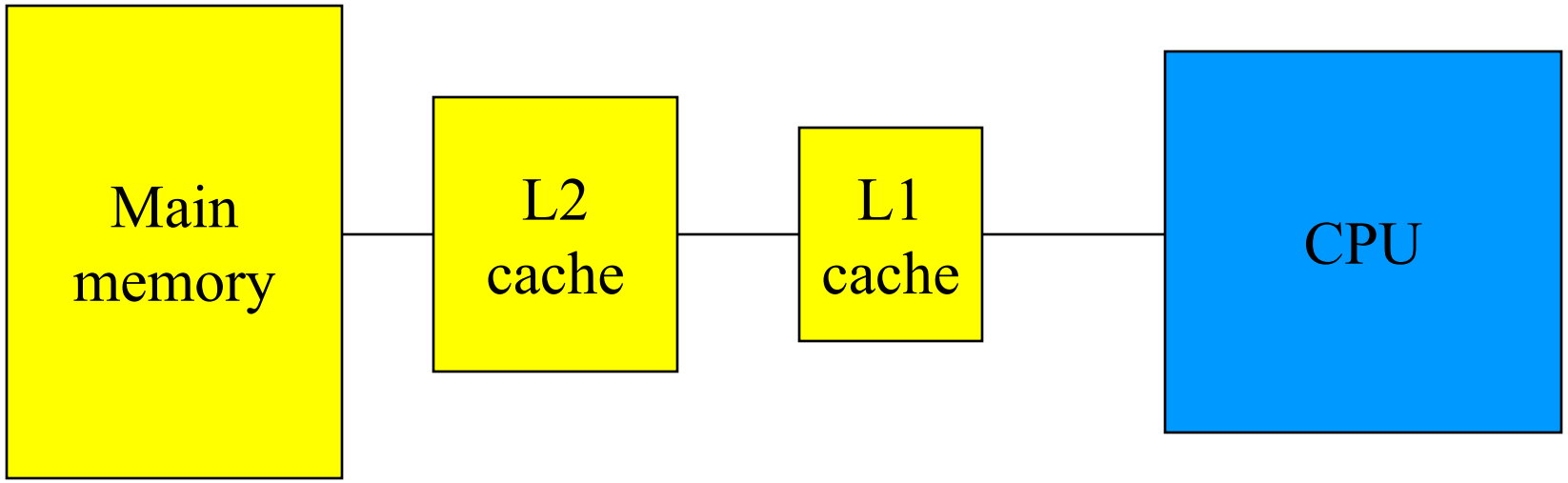
- Instructions are bundled with tag to indicate which instructions can be executed in parallel:



© 2000 Morgan Kaufman

Memory system

- CPU fetches data, instructions from a memory hierarchy:



© 2000 Morgan Kaufman

Memory hierarchy complications

- Program behavior is much more state-dependent.
  - Depends on how earlier execution left the cache.
- Execution time is less predictable.
  - Memory access times can vary by 100X.

© 2000 Morgan Kaufman