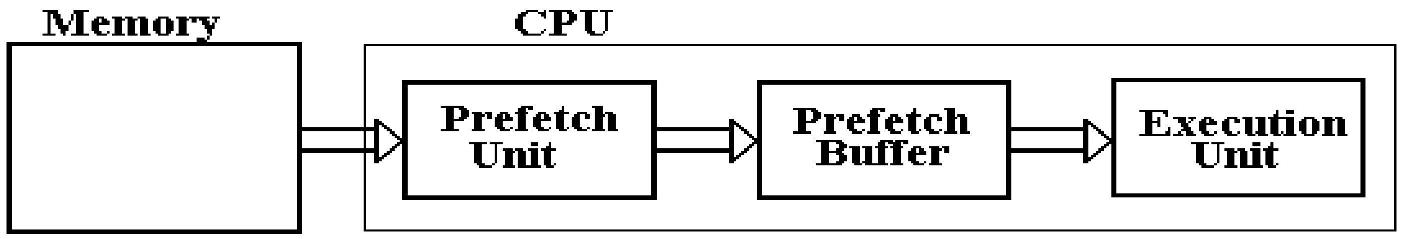
Instruction–Level Parallelism: Instruction Prefetch

Break up the fetch–execute cycle and do the two in parallel.

This dates to the IBM Stretch (1959)



The prefetch buffer is implemented in the CPU with on–chip registers.

The prefetch buffer is implemented as a single register or a queue.
The CDC–6600 buffer had a queue of length 8 (I think).

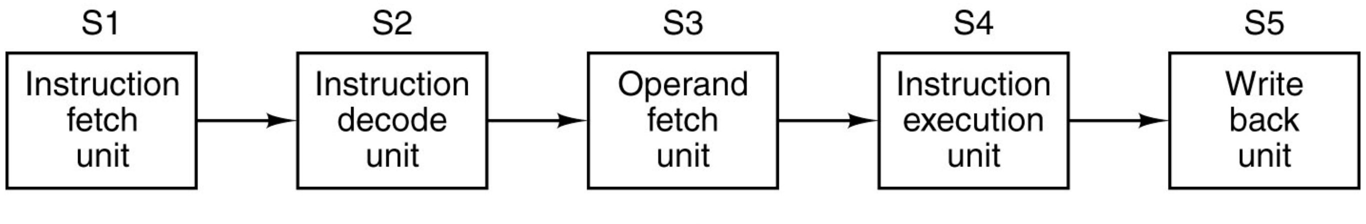
Think of the prefetch buffer as containing the IR (Instruction Register)

When the execution of one instruction completes, the next one is already in the buffer and does not need to be fetched.

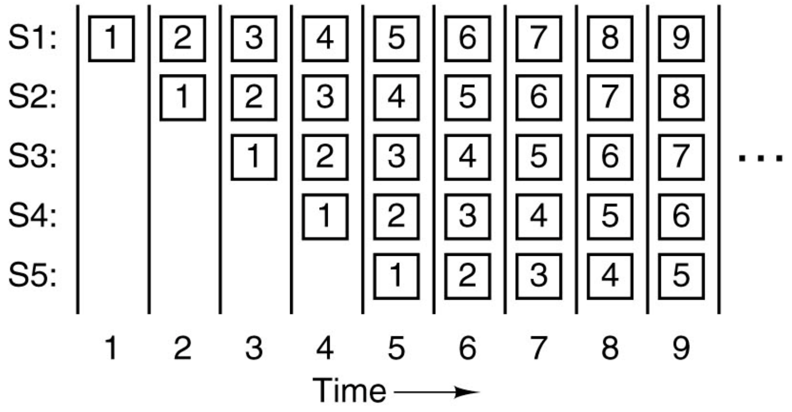
Naturally, a program branch (loop structure, conditional branch, etc.) invalidates the contents of the prefetch buffer, which must be reloaded.

Instruction–Level Parallelism: Pipelining

Better considered as an “assembly line”



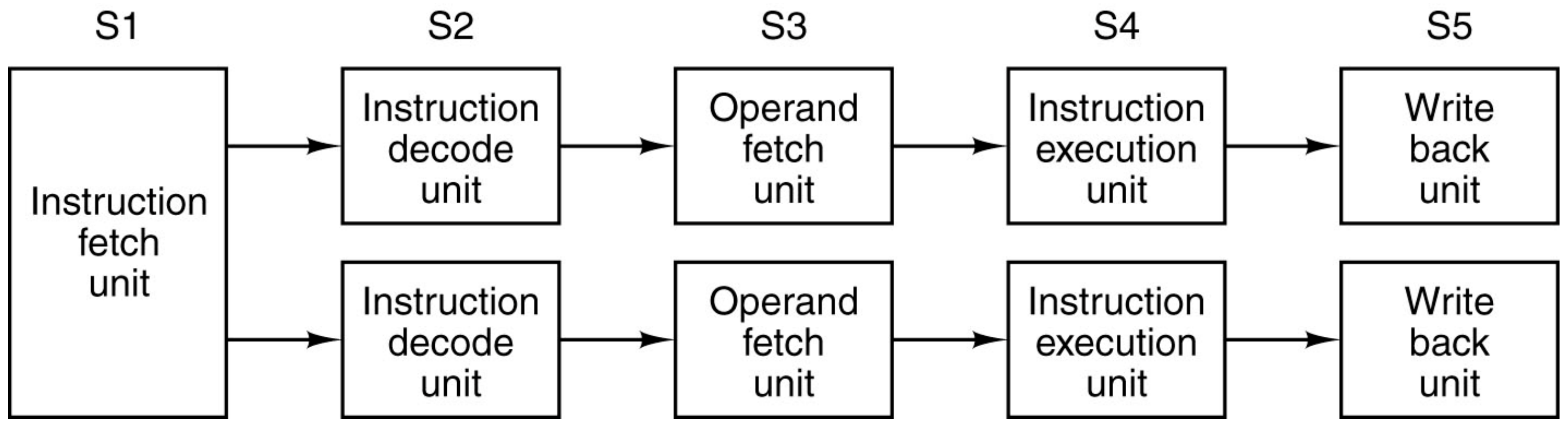
(a)



(b)

Note that the throughput is distinct from the time required for the execution of a single instruction. Here the throughput is five times the single instruction rate.

What About Two Pipelines?



Code emitted by a compiler tailored for this architecture has the possibility to run twice as fast as code emitted by a generic compiler.

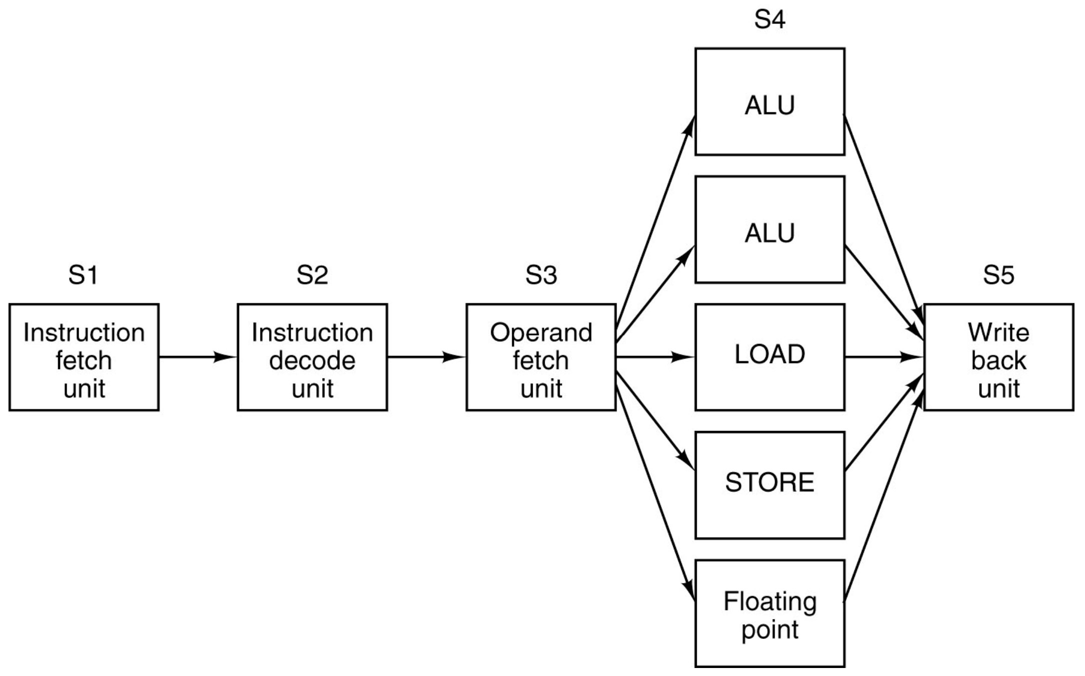
Some pairs of instructions are not candidates for dual pipelining.

C = A + B	C = A + B
D = A + C	C = C / D

Superscalar Architectures

Having 2, 4, or 8 completely independent pipelines on a CPU is very resource–intensive and not directly in response to careful analysis.

Often, the execution units are the slowest units by a large margin. It is usually a better use of resources to replicate the execution units.


S1
Instruction fetch unit
S2
Instruction decode unit
S3
Operand fetch unit
S4
ALU
ALU
LOAD
STORE
Floating point
S5
Write back unit