Memory Design

Memory can be built in different ways...

- Dynamic (DRAM): "leaky," has to be refreshed periodically
- Static (SRAM)

Memory access latency:

- DRAM: 50-70ns, larger memory size (can fit more in a given area)
- SRAM: 3-10ns, smaller memory size

CPU clock cycle time: 0.5-2ns

Processor-Memory Gap

Current trends:

Component	Capacity	Speed
Transistors/logic	1.4x each year	2x in 3 years
DRAM	4x in 3 years	1.4x in 10 years
Disk	4x in 3 years	1.4x in 10 years

The gap between processor speed and memory speed is growing.

CPI Equation

Suppose: 60ns DRAM, 500MHz CPU.

Instruction mix:

ALU: 50%, load/store: 30%, branch: 20%

Assume branch delay slots are all filled with `nop`s.

MIPS CPI:

$0.5 \times 1 + 0.2 \times 2 + 0.3 \times 30$

$\Rightarrow 9.9$!

Problem: memory is too slow.

CSL

Speeding Up Memory Access

Basic idea:

- Build a small, fast memory (*cache*)
- Use to store frequently accessed *blocks* of memory
- When it fills up, discard some blocks and replace them with others
- Works well if we are reusing data blocks

Examples: incrementing a variable, loops, function calls, etc.

Locality Principles

Temporal Locality

- the location of a memory reference is likely to be the same as another recent reference
- loops, function calls, etc (code)
- variables are reused in a program

Spatial Locality

- the location of a memory reference is likely to be near another recent reference
- matrices, arrays
- stack accesses

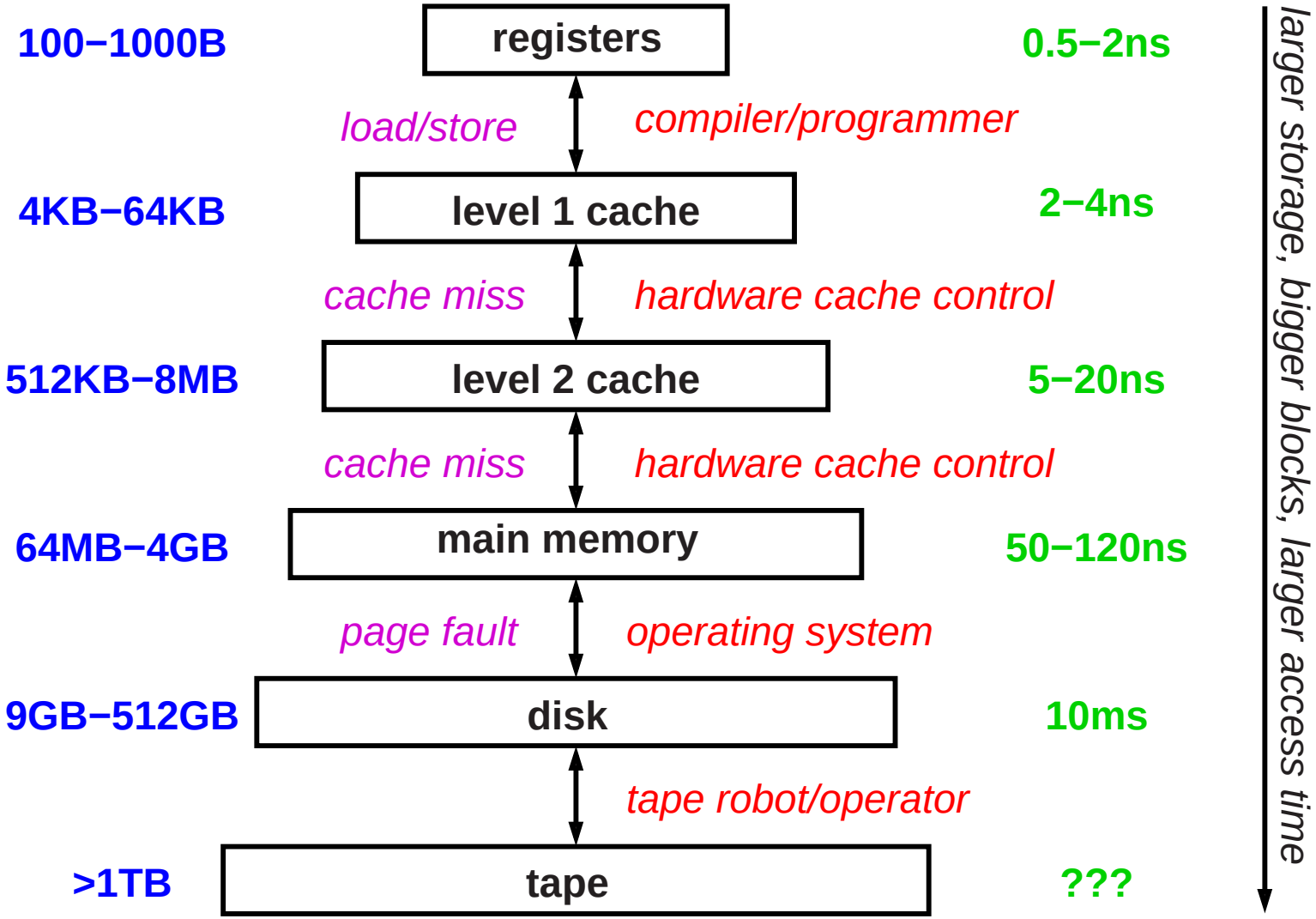
Cache Access

Reading memory in the presence of a cache:

- Check if the address is present in the cache
- If present, return data from the cache (a cache *hit*)
- If absent (a cache *miss*), read a *block* of data from the main memory, store it in the cache, and return data.

Why not repeat this since memory is slow?

Memory Hierarchy



Access Time

Hit rate

- fraction of accesses found at a given level
- normally talk about the *miss rate* (1 – hit rate)

Hit time

- Time to access cache + check for hit/miss

Miss penalty

- Time taken to replace block from next level

Average Memory Access Time

hit time + miss rate × miss penalty

Access Time

Example: 500MHz CPU

ALU: 50%, load/store: 30%, branch: 20%

Assume branch delay slots are all filled with nops.

Hit rate: 95%, miss penalty: 60ns, hit time: 2ns

MIPS CPI w/o cache for load/store:

$$0.5 \times 1 + 0.2 \times 2 + 0.3 \times 30 = 9.9$$

MIPS CPI w/ cache for load/store:

$$0.5 \times 1 + 0.2 \times 2 + 0.3 \times \underbrace{(1 + 0.05 \times 30)}_{memory} = 1.65$$

Cache Organization

The cache is simply a smaller memory; suppose it holds N blocks of M words each.

- need mapping function
  $f: addr \rightarrow [0, N-1] \times [0, M-1]$
- what happens when $f(a) = f(b)$ for $a \neq b$?
  $\Rightarrow$ store information to reconstruct the *address*

Simple strategy: direct-mapped

- Let a be a word address. Then:
  $$f(a) = ((a/M) \bmod N, a \bmod M)$$
- Save a/MN to reconstruct address

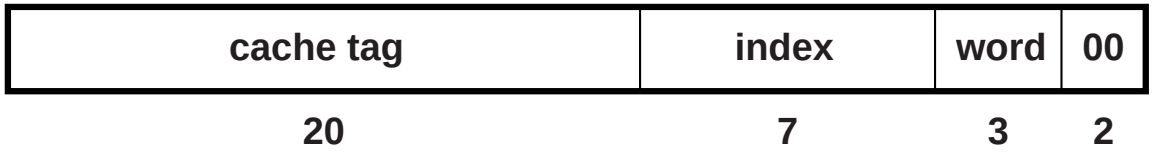
Direct-Mapped Cache

Example: 4KB cache, 32 byte blocks.

- M = 32/4 = 8 words
- N = 4096/32 = 128 blocks

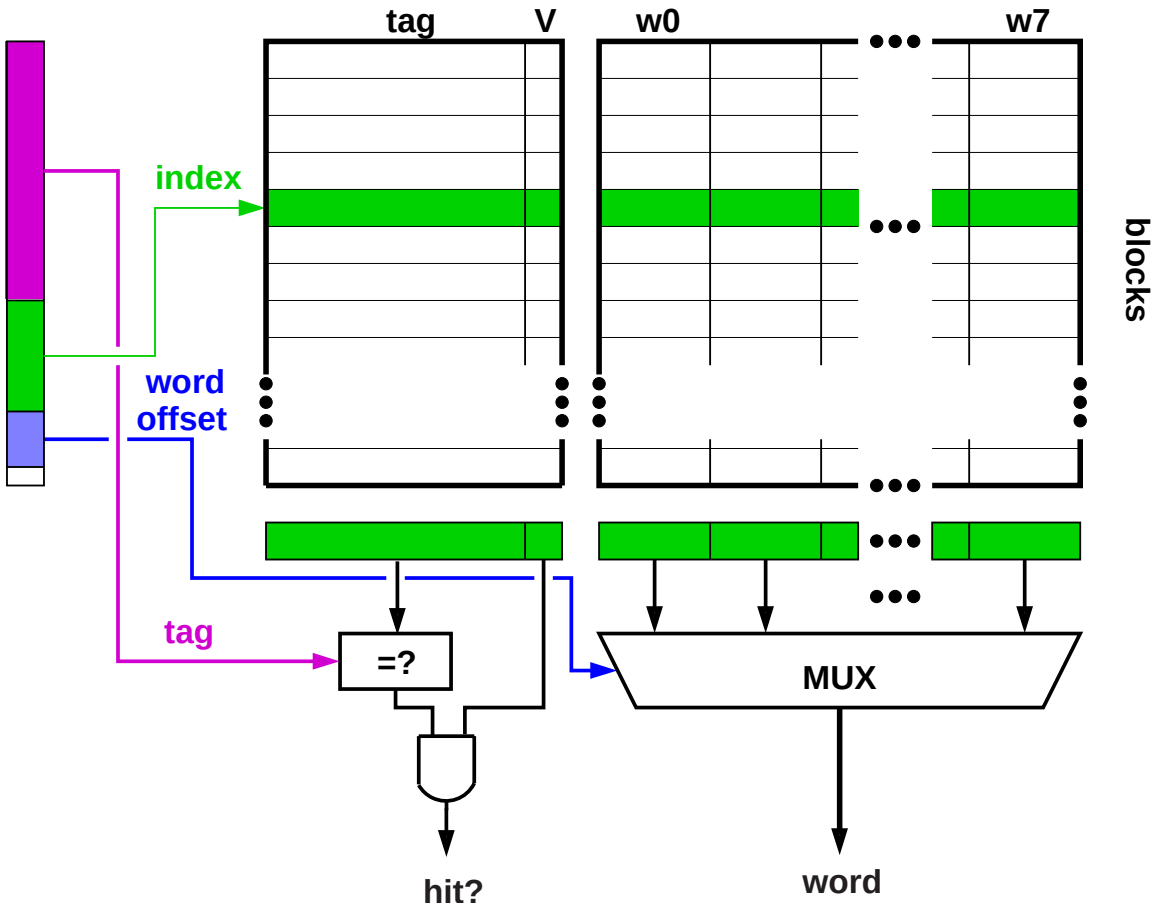
Calculating $f(a)$:

- Block index: (a/M) mod N
- Word offset: a mod M



The top bits of the address (tag) are stored in the cache so we can reconstruct the address.

Direct-Mapped Cache Organization



Data + tag for a block is called a cache line.

Cache Misses

Handling a cache miss:

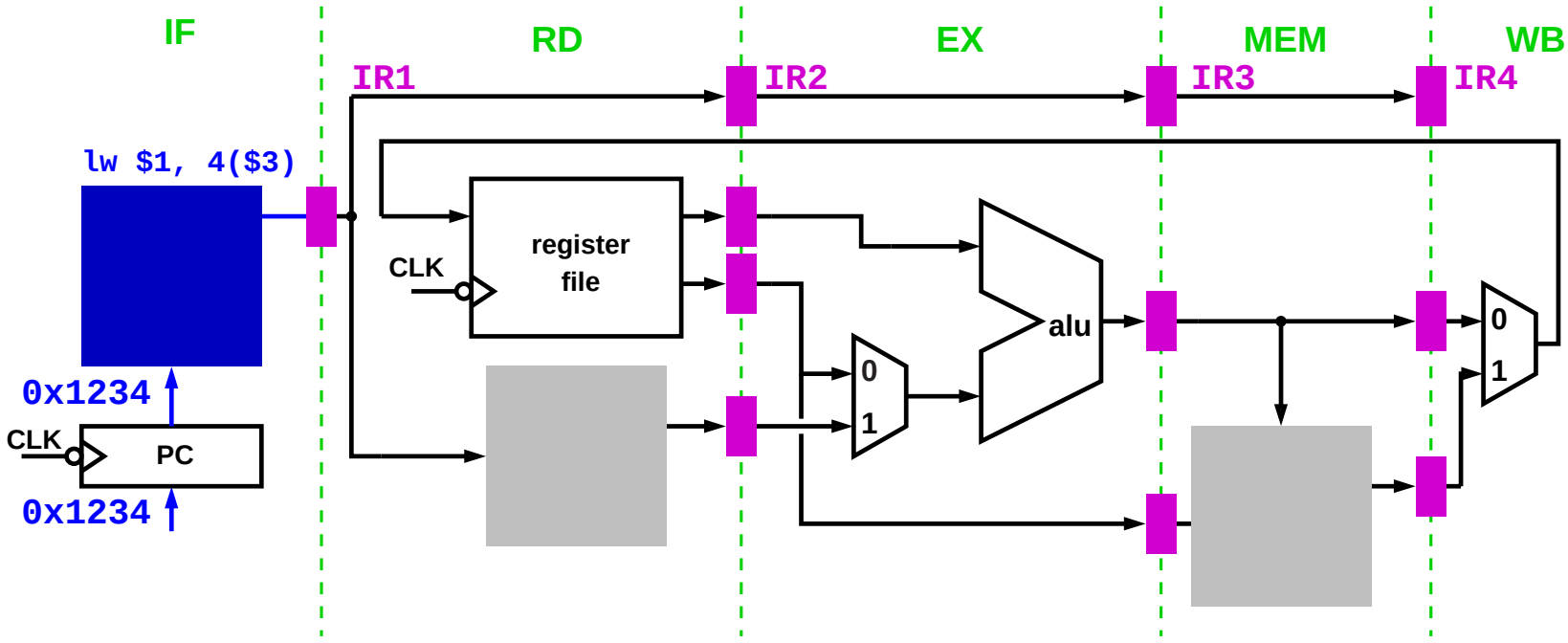
- Miss detected
- Address sent to the next level of the memory hierarchy
- Once data returned, *refill* the cache
- Processor gets data

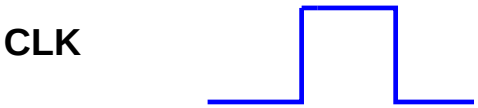
Done by a hardware finite state machine:

the cache control

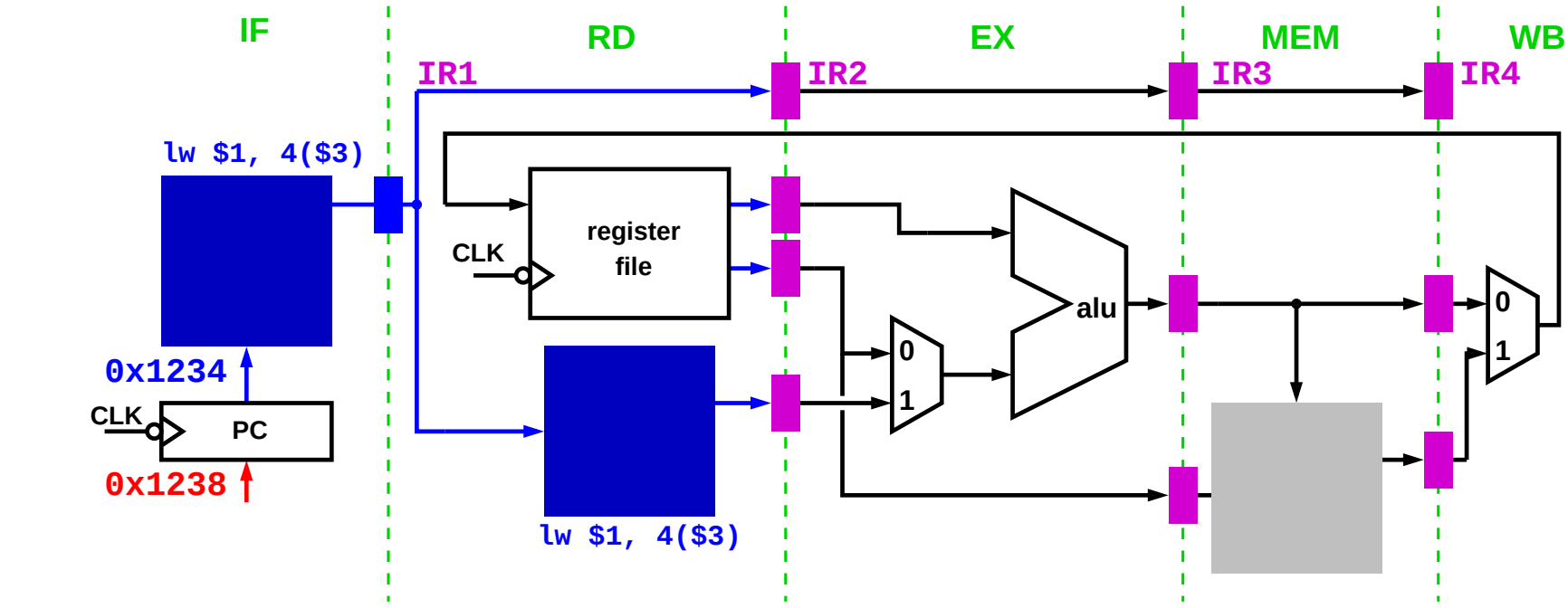
What happens to the processor pipeline?

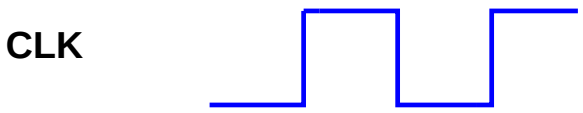
Pipelined Execution



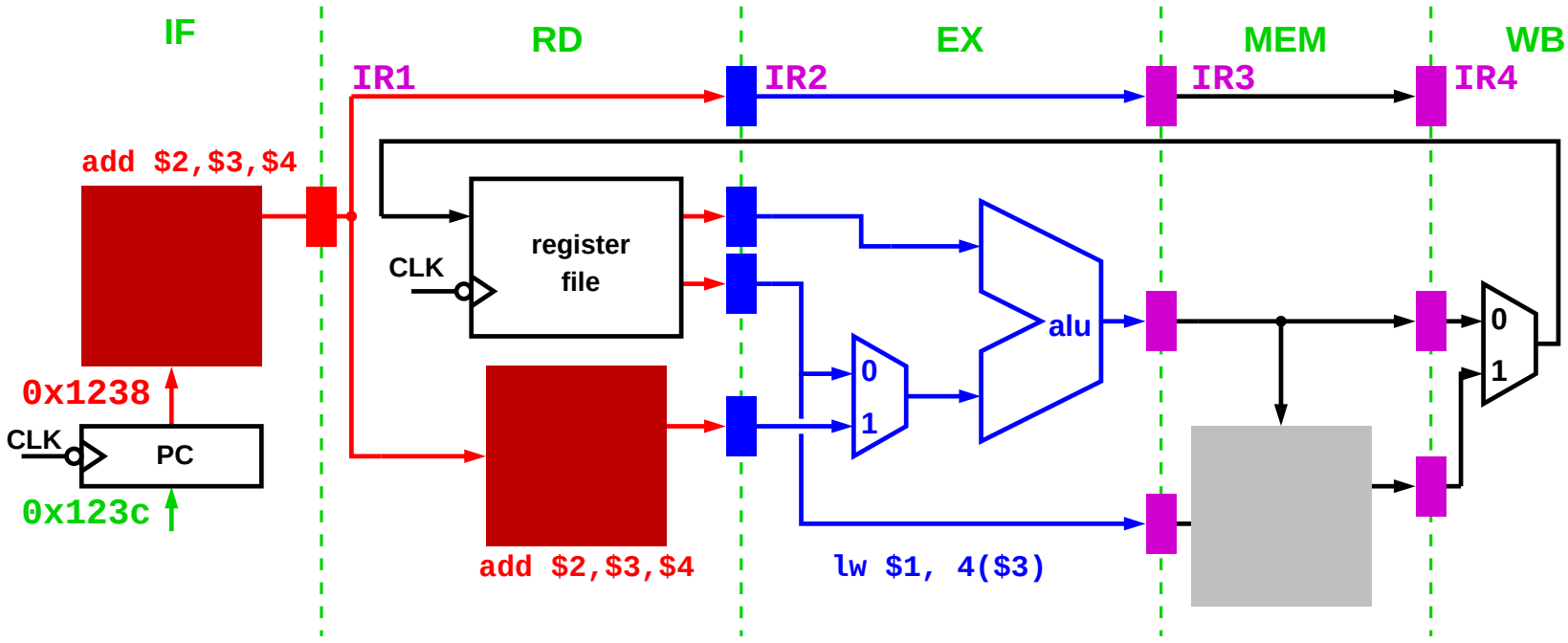


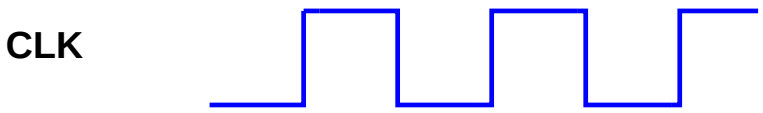
Pipelined Execution



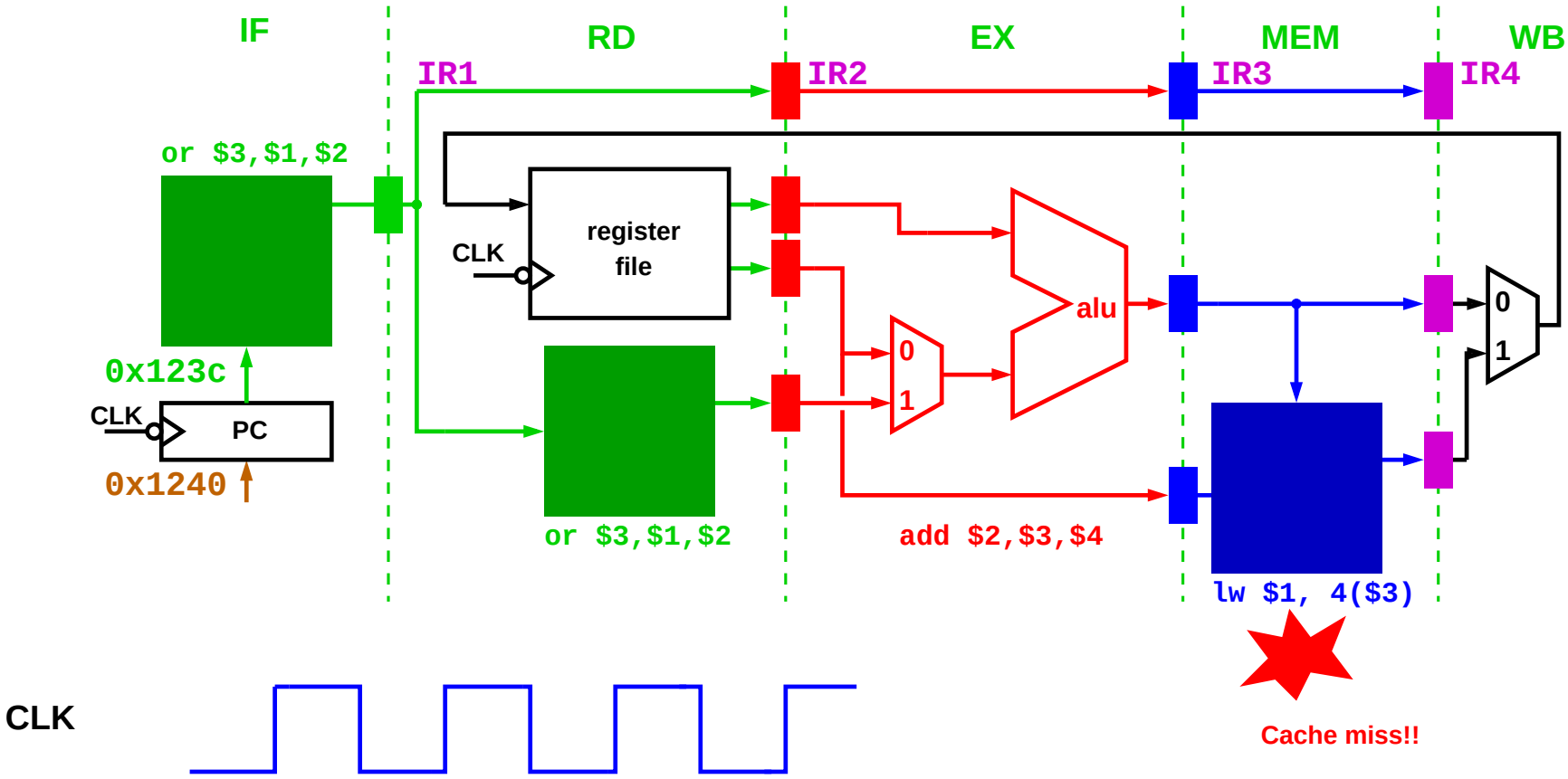


Pipelined Execution

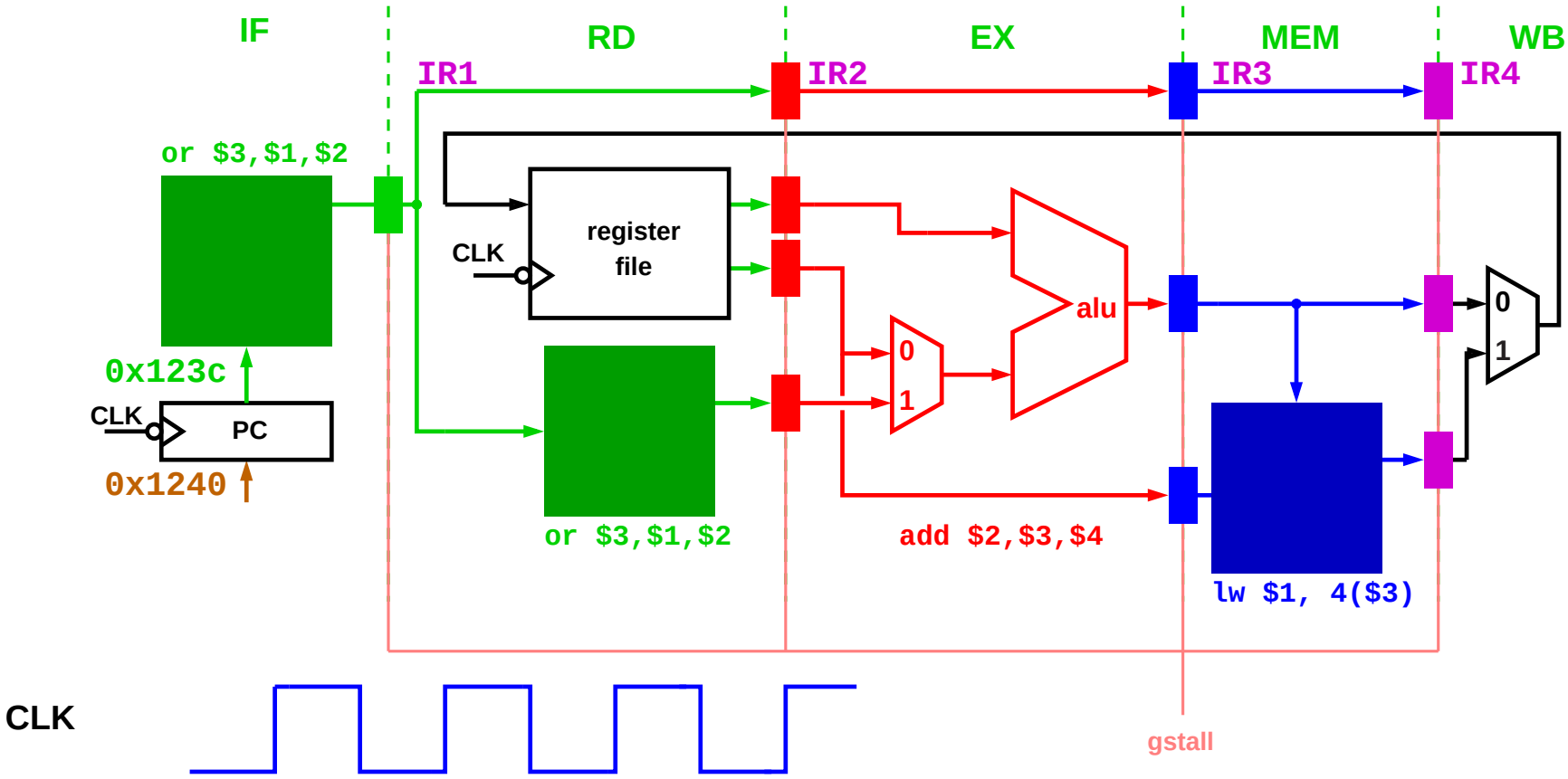




Pipelined Execution



Pipelined Execution



Handling Writes

Write through

- Write data to the cache + next level of hierarchy

Write back

- Write data to cache only
- Next level no longer has latest copy of data!
- Keep track of whether line is *clean* or *dirty*

Pros and Cons?