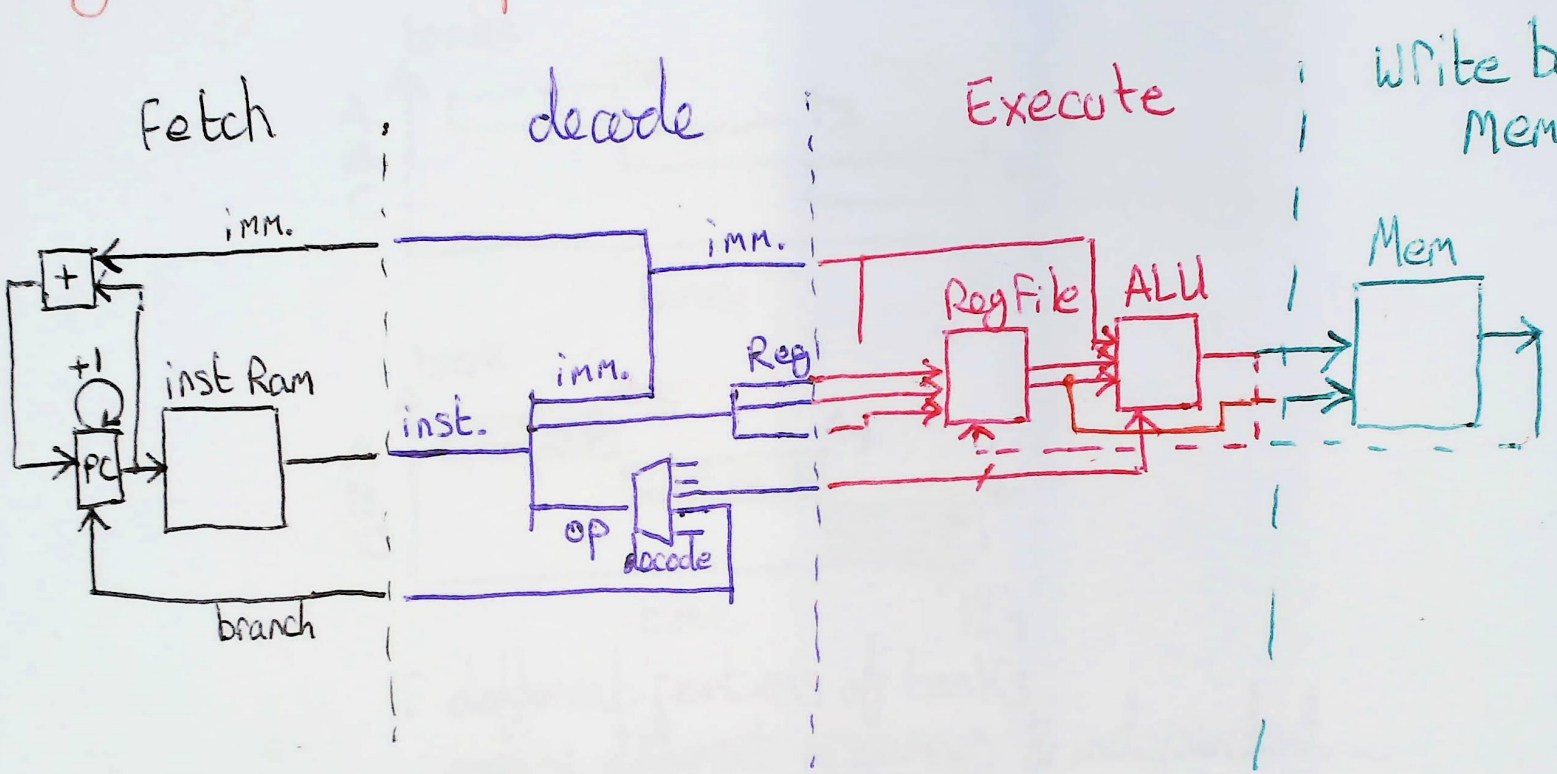
chapter 4: The processor

cpu perfomance factors processor

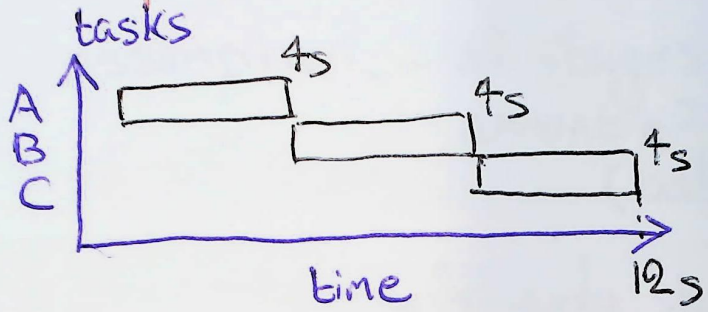
- Instruction Count:
  determined by ISA, compiler, algorithm.
- clocks per Instruction (CPI)
  or Instructions per clock (IPC)
  determined design of architecture and Technology (nm)

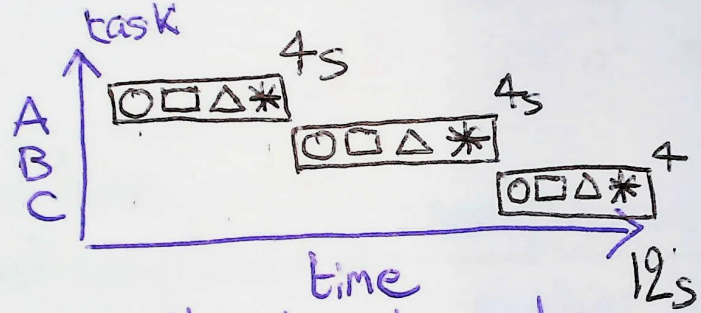
Logic Critical path



critical path: longest possible path through the gates of a circuit.

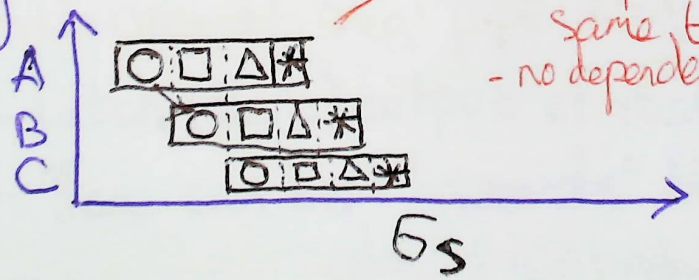
Pipeling in concept





if different portions of tasks require <u>different resources</u> & independent...

Pipelining



→ each portion takes about same time.
- no dependencies between <u>portions</u>

How to calculate speedup of pipelining

assumption: - all stages take same amount of time. (balanced pipeline)

- "steady state" speedup is of interest.

$$\text{Pipelined time of task completion} = \frac{\text{non-pipelined time of task}}{\text{number of pipeline stages}}$$

$$\text{ideal speedup} = \frac{\text{non-pipelined time}}{\text{pipelined time}} = \#\text{ pipeline stages}$$

Things that prevent ideal pipelining

- Latch (store) output of each stage has overhead
- There can be "hazards" that prevent ideal pipeling.

in processor easy to avoid hazard with "noop".

Pipeline Hazards

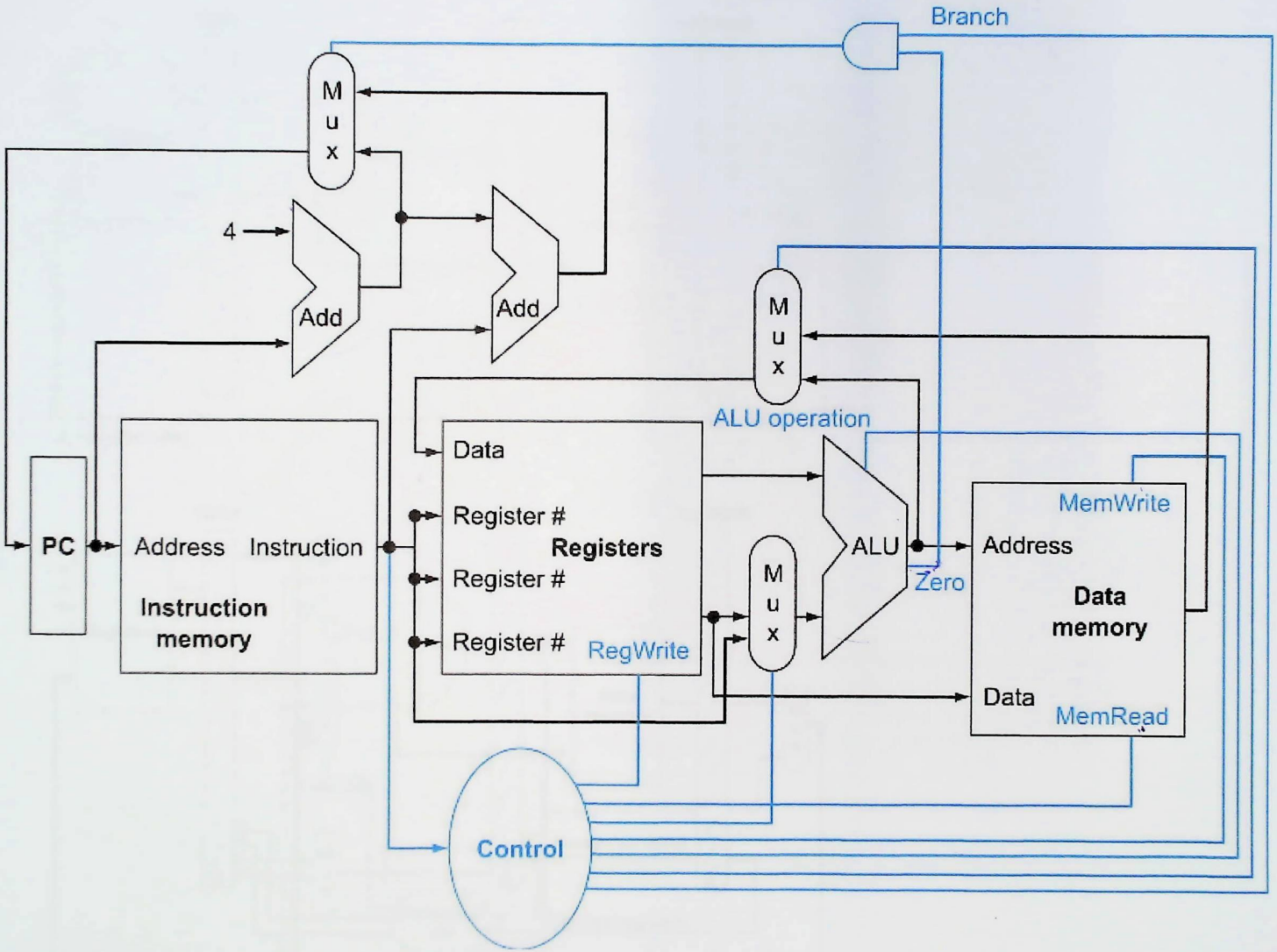
(dependencies that prevent ideal pipelining)

3 main type of hazards:

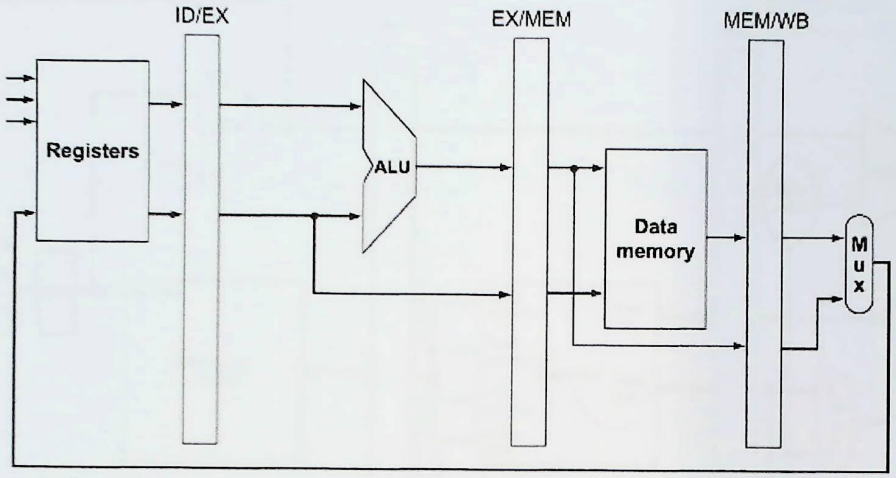
- structral: when different stages of pipeline access same hardware resource

- data: when result of a prior instruction is need to execute next instruction.

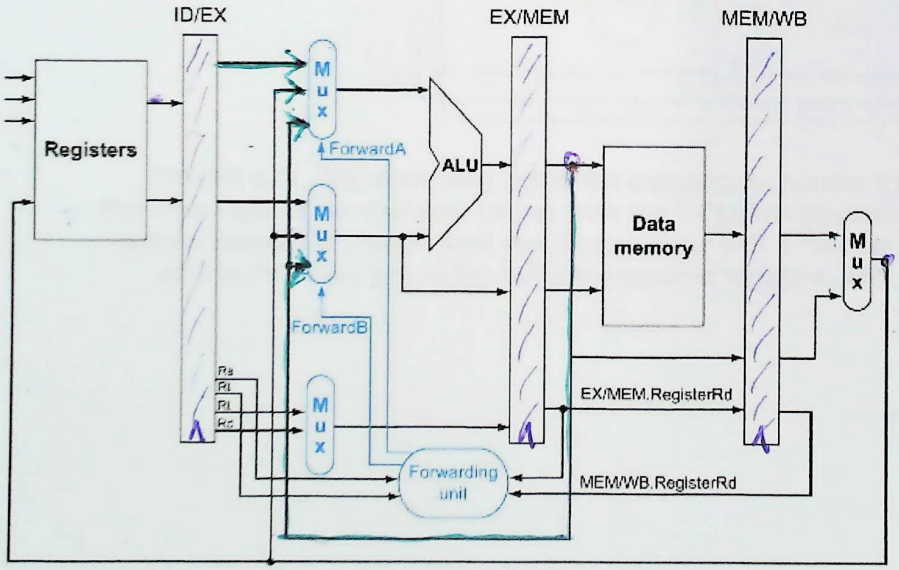
- control: when the next instruction to fetch depends on a prior instruction.


Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
PC
Address
Instruction
Instruction
memory
Data
Register #
Registers
Register #
Register #
RegWrite
M
u
x
ALU
Zero
MemWrite
Address
Data
memory
Data
MemRead
Control

latch



a. No forwarding



b. With forwarding

bypassing

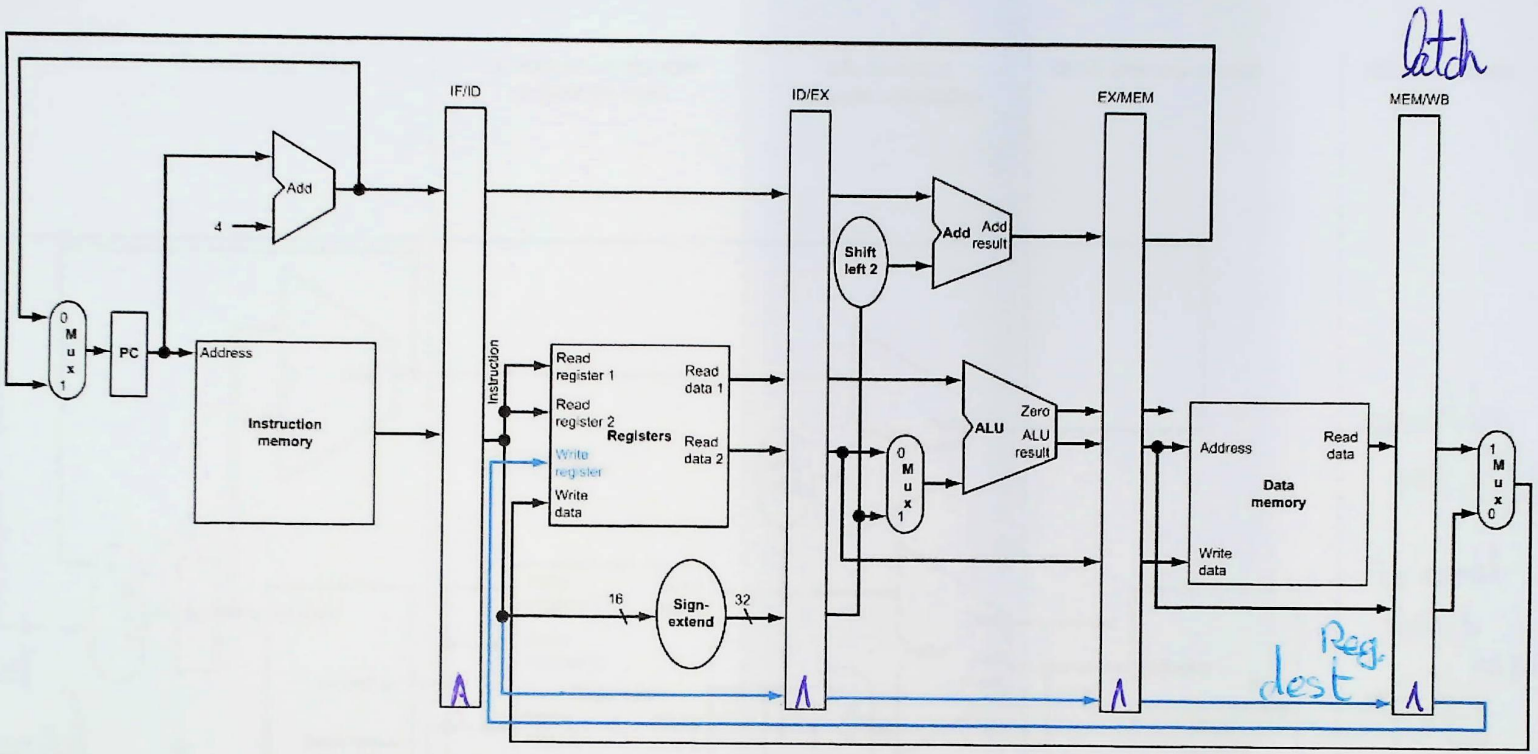
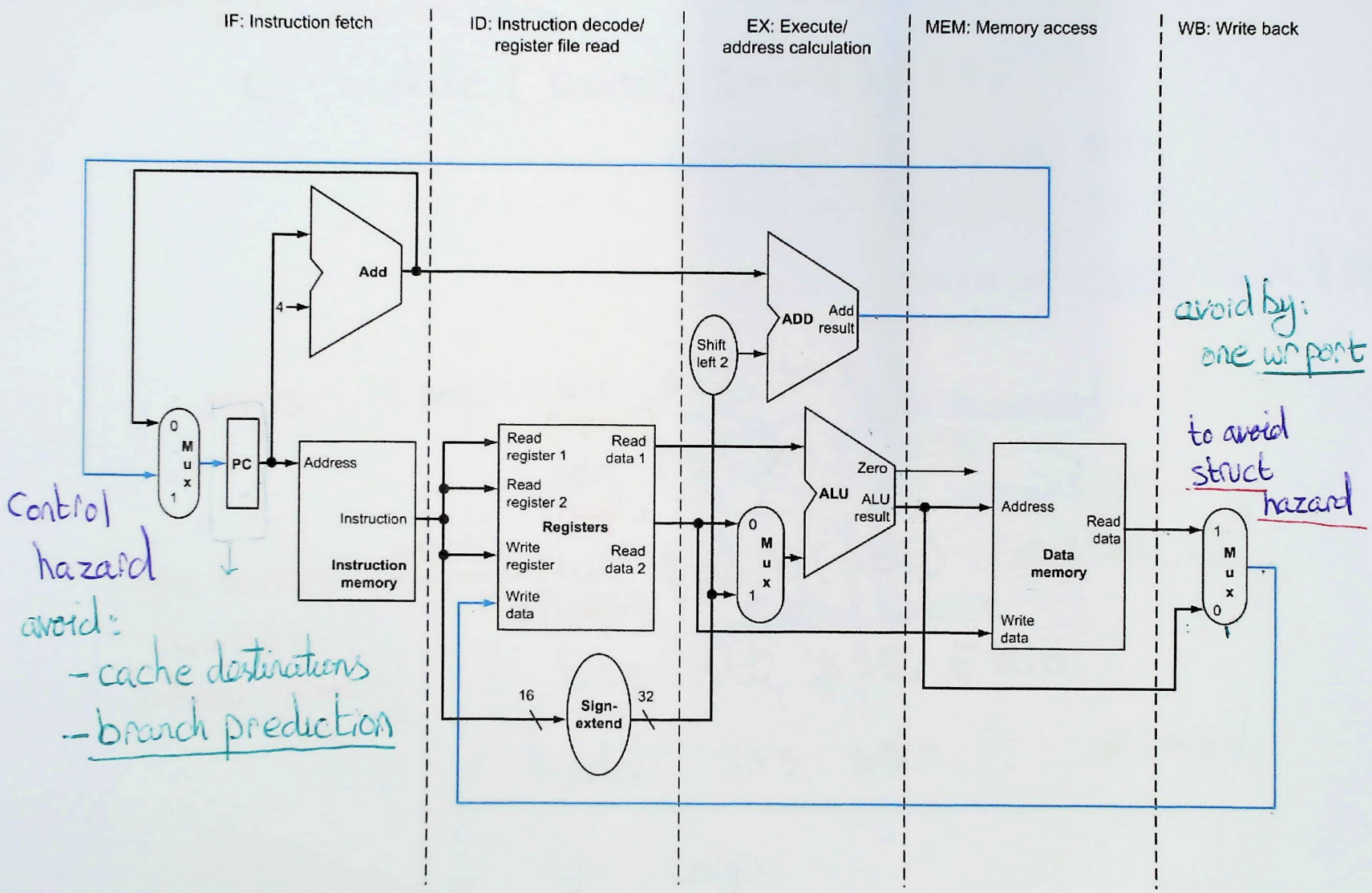


FIGURE 4.41 The corrected pipelined datapath to handle the load instruction properly. The write register number now comes from the MEM/WB pipeline register along with the data. The register number is passed from the ID pipe stage until it reaches the MEM/WB pipeline register, adding five more bits to the last three pipeline registers. This new path is shown in color.



Control hazard

avoid:

- cache destinations
- branch prediction

avoid by:

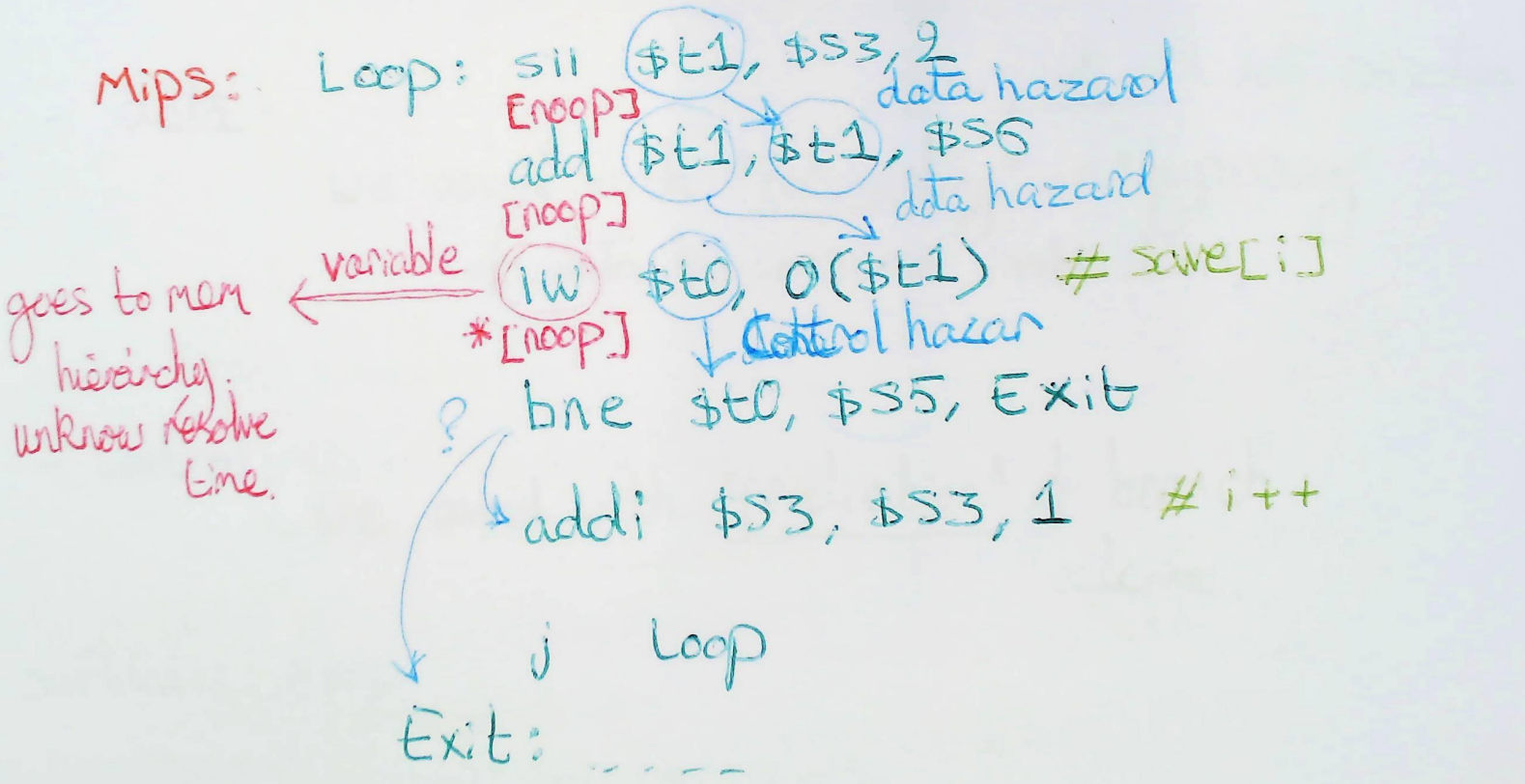
one wr port

to avoid struct hazard

Example of pipeline hazards code

C: `while ( save[i] == k ) i++;`

assump.: i is in $s3
k is in $s5
base of save is in $s6



How we avoid different pipeline hazards: [in Hardware]

- structural:

"writeback" → wb phase

We avoid by having single point of access to resource.

- data:

→ a quick path data connection

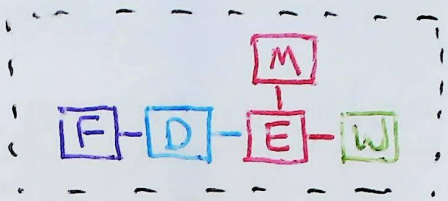
we avoid with "Forwarding" or "bypassing" of data to consumer instruction.

- Control:

we avoid with "prediction" of branch outcome.

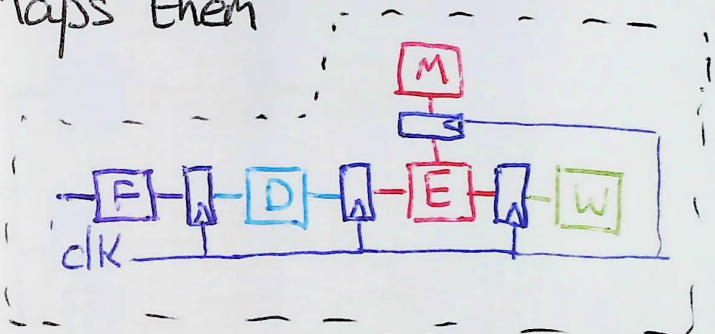
software: noop

Diffent Types of parallelism



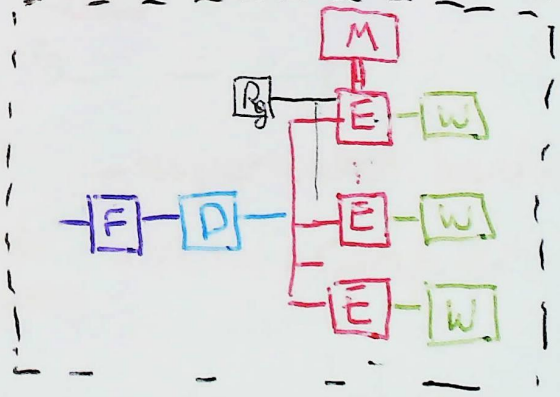
1) sub-instruction parallelism

- pipelining
- break instructions into portions and overlaps them



2) data-level parallelism

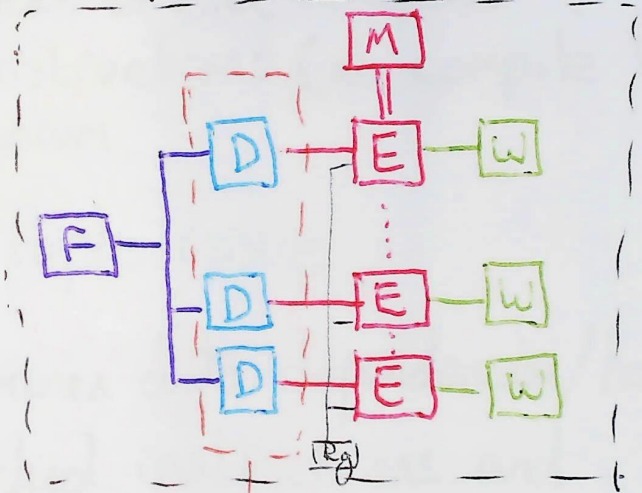
- one instruction works on multiple data.
- SIMD (single instruction multiple data)
- common in media/GPU processors



3) Instruction level parallelism

- single fetch. single progm counter.
- But possibly can fetch multiple inst. per clk.
- Can execute multiple inst. per clock.

Forms
1. superscalar*
2. Very Long Inst. Width. (VLIW)



pull out independent instructions

- superscalar: hardware.
- VLIW: Compiler.

3.a) ILP : static multiple issue

- pack multiple indepent instructions together, and fetchead togethe
- Very Long Instruction width architecture (VLIW)
- Compiler need to Find indepent instructions.(at compile time).

e.x. Intel Itanium

3.b) ILP : dynamic multiple issue

- hardware exarines data dependencies/hazards between Fetched instructions. and clears them for execute.
- superscalar

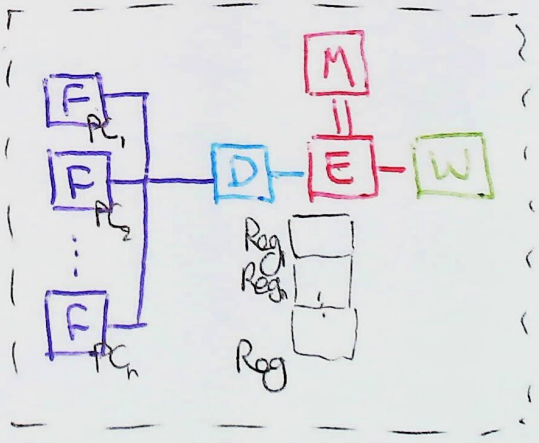
4) Thread Level parallelism

Multithreading:

- managed by operating system (OS).
- OS stores and retreives the program counter and reg file of a program, to switch it in/out of execution.

Symultanious Multithreading

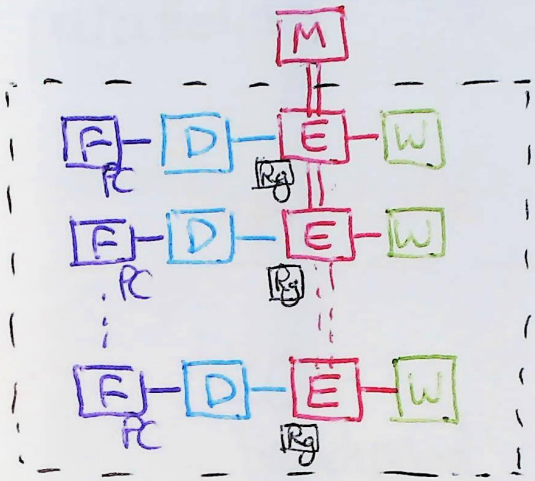
- Keep multiple program counter and regfile.
- dynamically pick instructions from threads to keep processor bussy.
- Intel: "hyper-threading"



5) multicore - Full thread-level parallelism

- stamp out copies of your "core" processor

- important: how to hook up all the cores to a unified memory.



Contemporary processors and forms of parallelism

	AMD Ryzen 7000	Intel Raptor Lake	NVida 4090 GPU
pipelining (sub-instr parallelism)	20 branch pred.	20+ branchpred	deep
data level	special SIMD for Media	special SIMD for Media	SIMD 32 default
Instruction level	superscalar 5x	superscalar 5x	None
thread level	2 thread per core	1 thread per core 2 thread per core	32 threads per core
Core-level	16 core	16 "E" (2 thread) + 8 "P" (1 thread)	very long. +