

EN164: Design of Computing Systems

Lecture 20: Processor / ILP 1

Professor Sherief Reda

<http://scale.engin.brown.edu>

Electrical Sciences and Computer Engineering
School of Engineering
Brown University
Spring 2011



[material from Patterson & Hennessy, 4th ed and Harris 1st ed]

Instruction Level Parallelism

- Pipelining: executing multiple instructions in parallel
- To increase ILP
 - Deeper pipeline: more multiple instructions
 - Less work per stage \Rightarrow shorter clock cycle
 - Multiple issue
 - Replicate pipeline stages \Rightarrow multiple pipelines
 - Start multiple instructions per clock cycle
 - $CPI < 1$, or Instructions Per Cycle (IPC) > 1
 - E.g., 4GHz 4-way multiple-issue
 - 16 BIPS, peak $CPI = 0.25$, peak $IPC = 4$
 - But dependencies reduce this in practice

Where can performance be improved?

$$\begin{aligned} \text{CPU Time} &= \text{CPU Clock Cycles} \times \text{Clock Cycle Time} \\ &= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} \end{aligned}$$

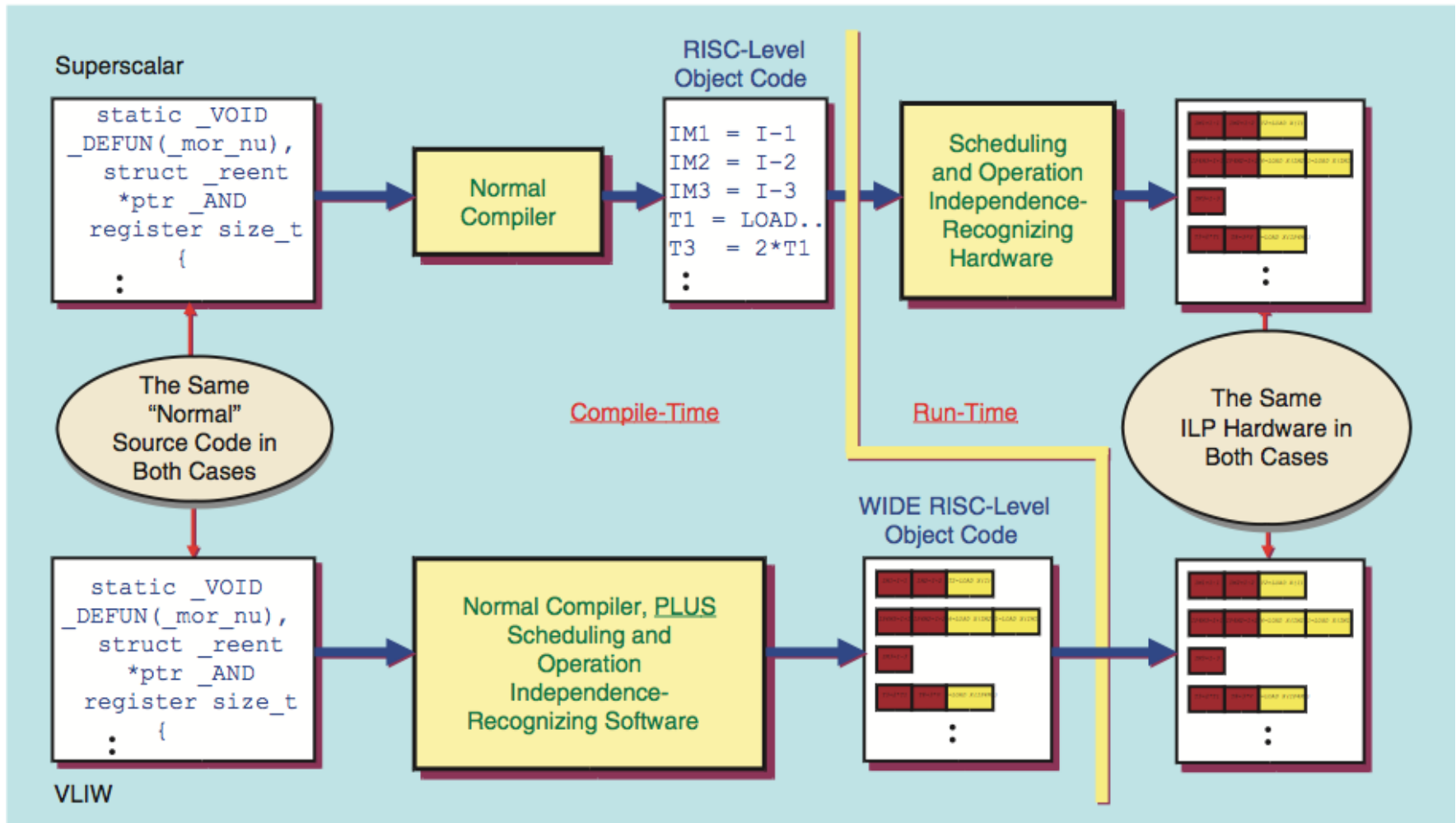
optimization goal for multiple-issue processors

- Pipeline CPI = Ideal pipeline CPI + Structural Stalls + Data Hazard Stalls + Control Stalls

Processor design for multiple issue

- Static multiple issue (VLIW)
 - Compiler groups instructions to be issued together
Very Large Instruction Words (VLIW)
 - Packages them into “issue slots”
 - Compiler detects and avoids hazards
- Dynamic multiple issue (superscalar)
 - CPU examines instruction stream and chooses instructions to issue each cycle
 - Compiler can help by reordering instructions
 - CPU resolves hazards using advanced techniques at runtime
 - *There are also static multiple issue superscalars*

Difference between superscalar and VLIW



[from Fisher *et al.*]

Static multiple issue

- Compiler groups instructions into “issue packets”
 - Group of instructions that can be issued on a single cycle
 - Determined by pipeline resources required
- Think of an issue packet as a very long instruction
 - Specifies multiple concurrent operations
 - ⇒ Very Long Instruction Word (VLIW)

Scheduling static multiple issue

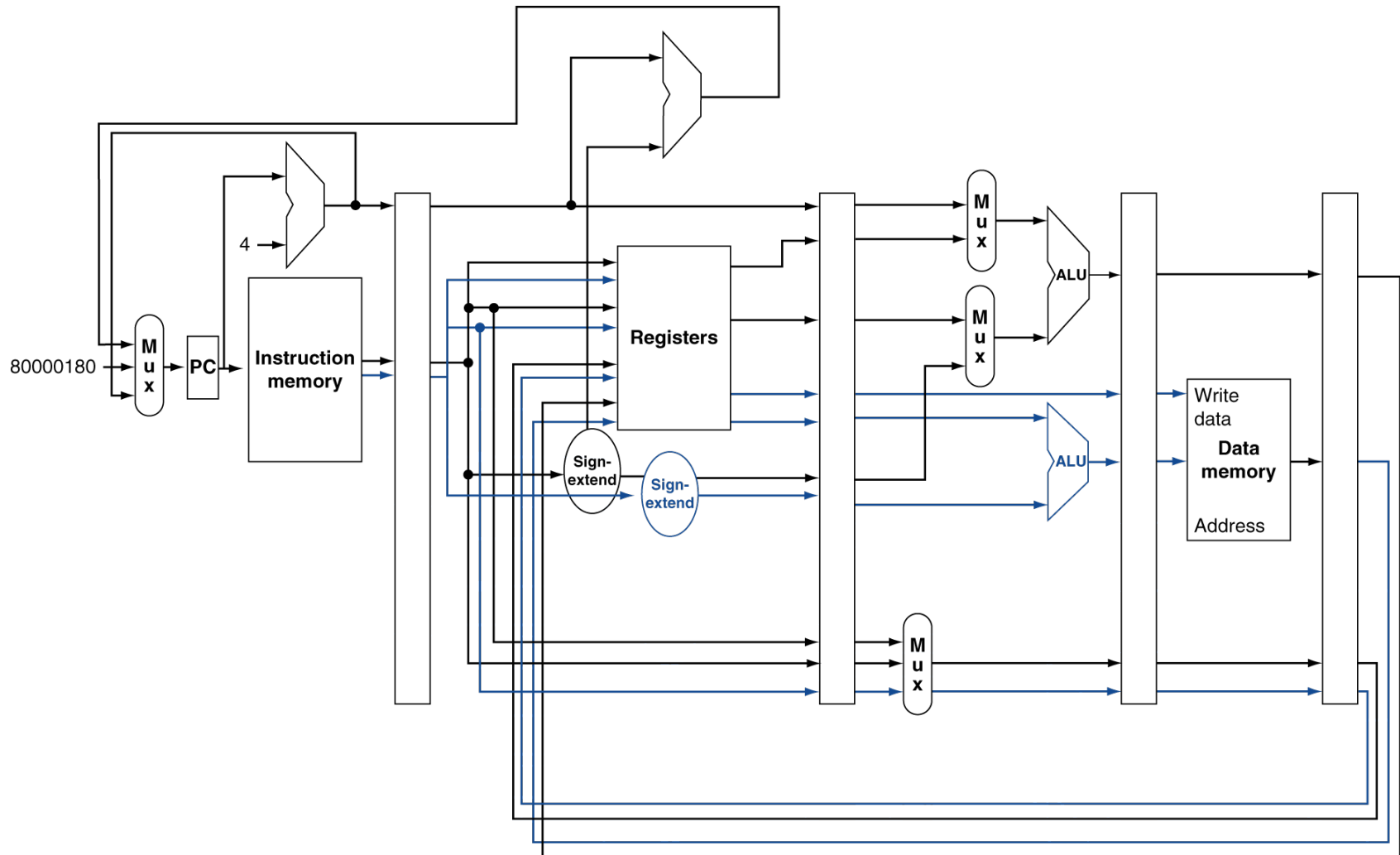
- Compiler must remove some/all hazards
 - Reorder instructions into issue packets
 - No dependencies with a packet
 - Possibly some dependencies between packets
 - Varies between ISAs; compiler must know!
 - Pad with nop if necessary

MIPS with static dual issue

- Two-issue packets
 - One ALU/branch instruction
 - One load/store instruction
 - 64-bit aligned
 - ALU/branch, then load/store
 - Pad an unused instruction with nop

Address	Instruction type	Pipeline Stages						
n	ALU/branch	IF	ID	EX	MEM	WB		
n + 4	Load/store	IF	ID	EX	MEM	WB		
n + 8	ALU/branch		IF	ID	EX	MEM	WB	
n + 12	Load/store		IF	ID	EX	MEM	WB	
n + 16	ALU/branch			IF	ID	EX	MEM	WB
n + 20	Load/store			IF	ID	EX	MEM	WB

Pipeline design for multiple issue



Hazards in dual-issue MIPS

- More instructions executing in parallel
- EX data hazard
 - Forwarding avoided stalls with single-issue
 - Now can't use ALU result in load/store in same packet
 - add `$t0, $s0, $s1`
Load `$s2, 0($t0)`
 - Split into two packets, effectively a stall
- Load-use hazard
 - Still one cycle use latency, but now two instructions
- More aggressive scheduling required