

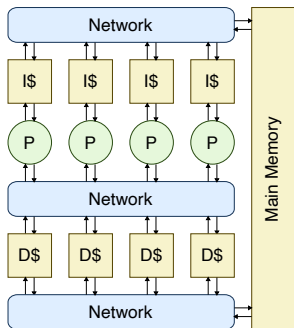
ECE 4750 Computer Architecture, Fall 2022

Topic 5: Integrating Processors and Memories

School of Electrical and Computer Engineering
Cornell University

revision: 2022-10-24-10-00

1	Processor and L1 Cache Interface	2
2	Analyzing Processor + Cache Performance	7
3	Case Study: MIPS R4000	9



- **Processors** for computation
- **Memories** for storage
- **Networks** for communication

Copyright © 2022 Christopher Batten. All rights reserved. This handout was prepared by Prof. Christopher Batten at Cornell University for ECE 4750 Computer Architecture. Download and use of this handout is permitted for individual educational non-commercial purposes only. Redistribution either in part or in whole via both commercial or non-commercial means requires written permission.

1. Processor and L1 Cache Interface

Approaches to integrate L1 caches into a processor pipeline vary based on how the L1 memory system is encapsulated and implemented.

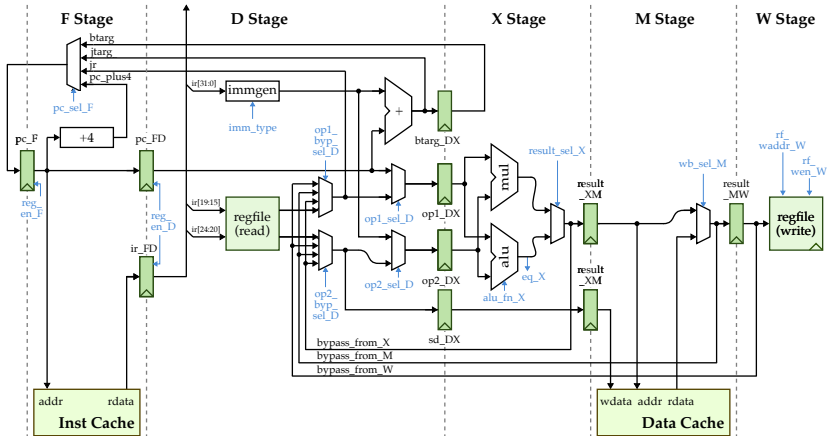
Tightly Coupled Interface

Loosely Coupled Interface

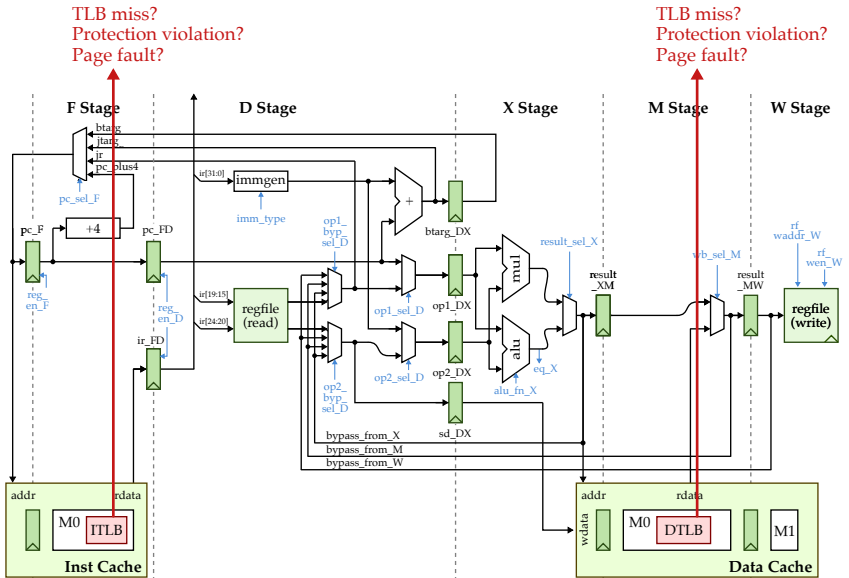
Processor contains L1 memory system and has fine grain control over the L1 microarchitecture through control/status signals.

Processor communicates with L1 memory system over (potentially latency insensitive) communication channels.

Zero-Cycle Hit Latency with Tightly Coupled Interface



Integrating Instruction and Data TLBs



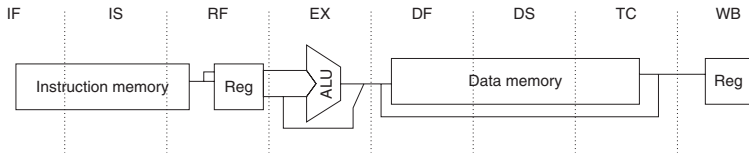
- TLB miss needs a hardware or software mechanism to refill TLB
- Software handlers need restartable exceptions on page fault
- Need mechanism to cope with the additional latency of a TLB
 - Increase the cycle time
 - Pipeline the TLB and cache access
 - Use virtually addressed caches
 - Access TLB and cache in parallel

2. Analyzing Processor + Cache Performance

How long in cycles will it take to execute the `vvadd` example assuming `n` is 64? Assume cache is initially empty, parallel-read/pipelined-write, four-way set-associative, write-back/write-allocate, and miss penalty is two cycles.

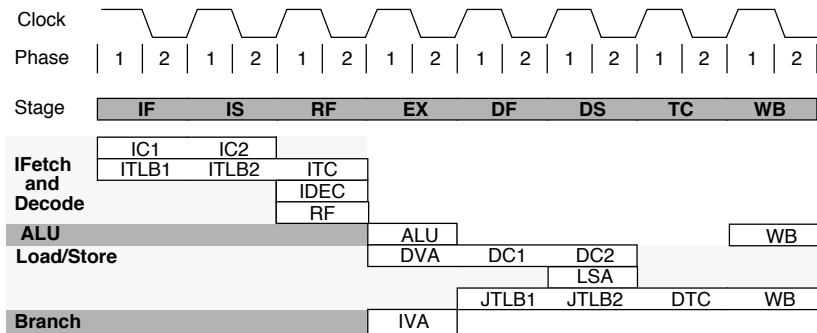
```
loop:
  lw   x5, 0(x13)
  lw   x6, 0(x14)
  add  x7, x5, x6
  sw   x7, 0(x12)
  addi x13, x12, 4
  addi x14, x14, 4
  addi x12, x12, 4
  addi x15, x15, -1
  bne  x15, x0, loop
  jr   x1
```


3. Case Study: MIPS R4000



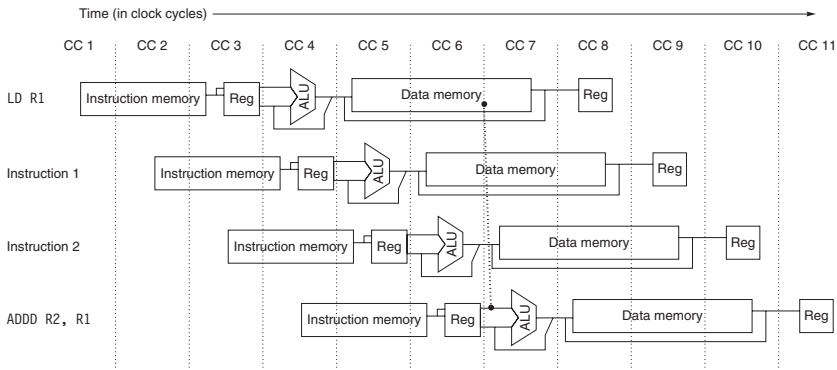
- 8-stage pipeline with extra stages for instruction/data mem access
 - IF: First-half of inst fetch
 - IS: Second half of inst fetch
 - RF: Instruction decode, register read, stall logic
 - EX: Execution (including effective address calculation)
 - DF: First-half of data fetch
 - DS: Second half of data fetch
 - TC: Tag check
 - WB: Write-back for loads and reg-reg operations
- Longer pipeline results in
 - Decreased cycle time
 - Increased load-use delay latency and branch resolution latency
 - More bypass paths

3. Case Study: MIPS R4000



- IC1 Instruction cache access stage 1
- IC2 Instruction cache access stage 2
- ITLB1 Instruction address translation stage 1
- ITLB2 Instruction address translation stage 2
- ITC Instruction tag check
- IDEC Instruction decode
- RF Register operand fetch
- ALU Operation
- DVA Data virtual address calculation
- DC1 Data cache access stage 1
- DC2 Data cache access stage 2
- LSA Data load or store align
- JTLB1 Data/Instruction address translation stage 1
- JTLB2 Data/Instruction address translation stage 2
- DTC Data tag check
- IVA Instruction virtual address calculation
- WB Write back to register file

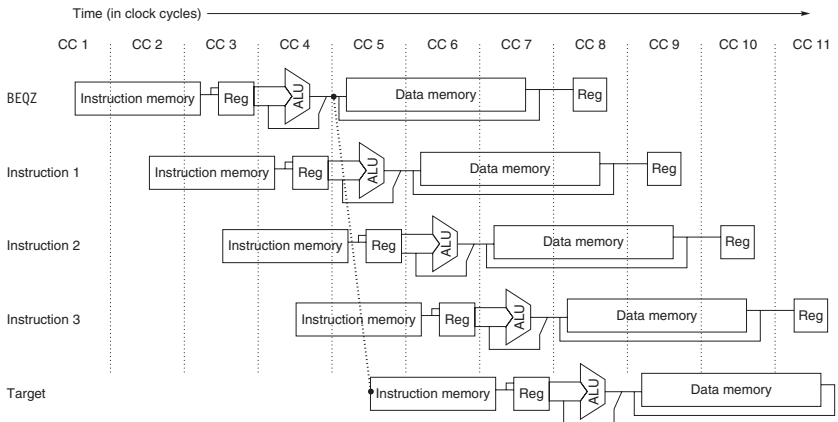
Load-Use Delay Latency



Cycle	Run	Run	Run	Run	Run	Run	Run	Run	Stl	Stl	Stl	Stl	Stl	Stl	Run	Run	Run	Run	Run		
Restart														Rst2	Rst1						
Load	IF	IS	RF	EX	DF	DS	TC					DF	DS	TC	WB						
		IF	IS	RF	EX	DF	DS							DF	DS	TC	WB				
ALU			IF	IS	RF	EX	DF								DF	DS	TC	WB			
				IF	IS	RF	EX-								RF	EX+	DF	DS	TC	WB	
					IF	IS	RF									EX	DF	DS	TC	WB	

- Load-use delay latency increased by one cycle
- Data is forwarded from end of DS stage to end of RF stage
- Tag check does not happen until TC!
- On miss, instruction behind load may have bypassed incorrect data
- EX stage of dependent instruction needs to be *re-executed*

Branch Resolution Latency



- Branches are resolved in EX stage
- Instruction 1 is in the branch delay slot
- Use predicted not-taken for instruction 2 and 3

cycle	Iteration 1												Iteration 2											
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4
lw x5, 0(x13)	F	D	X	M	M	M	W																	
lw x6, 0(x14)	F	D	X	X	M	M	M	W																
add x7, x5, x6	F	D	D	D	D	D	D	X	M	W														
sw x7, 0(x12)	F	F	F	F	F	F	F	X	M	M	M	W												
addi x13, x12, 4					F	D	X	X	M	W														
addi x14, x14, 4					F	D	D	D	X	M	W													
addi x12, x12, 4					F	F	F	D	X	M	W													
addi x15, x15, -1					F	D	X	M	W															
bne x15, x0, loop					F	D	X	M	W															
opA					F	-	-	-	-															
opB					F	-	-	-	-															
lw x5, 0(x13)					F	D	X	M	W															
lw x6, 0(x14)					F	D	X	M	W															
add x7, x5, x6					F	D	X	M	W															
sw x7, 0(x12)					F	F	D	X	M	W														
addi x13, x12, 4					F	D	X	M	W															
addi x14, x14, 4					F	D	X	M	W															
addi x12, x12, 4					F	D	X	M	W															
addi x15, x15, -1					F	D	X	M	W															
bne x15, x0, loop					F	D	X	M	W															
opA					F	-	-	-	-															
opB					F	-	-	-	-															
lw x5, 0(x13)					F	D	X	M	W															

u = cycle of useful work
 m = cycle lost due to memory stall
 r = cycle lost due to RAW stall
 c = cycle lost due to control squashes

* First Iteration CPI

+ num of insts = 9
 + num of cycles = 18
 + CPI = 2.00

* First Iteration CPI Breakdown

+ u = 9 cycles, 9/9 = 1.00 CPI
 + m = 6 cycles, 6/9 = 0.67 CPI
 + r = 1 cycle, 1/9 = 0.11 CPI
 + c = 2 cycles, 2/9 = 0.22 CPI
 + total = 2.00 CPI

* Second Iteration CPI

+ num of insts = 9
 + num of cycles = 12
 + CPI = 1.33

* Second Iteration CPI Breakdown

+ u = 9 cycles, 9/9 = 1.00 CPI
 + m = 0 cycles, 0/9 = 0.00 CPI
 + r = 1 cycle, 1/9 = 0.11 CPI
 + c = 2 cycles, 2/9 = 0.22 CPI
 + total = 1.33 CPI

* Overall CPI

+ num of iterations like first iteration = 16
 + num of iterations like second iteration = 64-16 = 48
 + total num insts = 9*64 = 576
 + total num cycles = 16*18 + 48*12 = 864
 + total CPI = 1.50

* Overall CPI Breakdown

+ u = 16*9 + 48*9 = 576 cycles, 576/576 = 1.00 CPI
 + m = 16*6 + 48*0 = 96 cycles, 96/576 = 0.17 CPI
 + r = 16*1 + 48*1 = 64 cycles, 64/576 = 0.11 CPI
 + c = 16*2 + 48*2 = 128 cycles, 128/576 = 0.22 CPI
 + total = 1.50 CPI