Cache Optimization Summary

Technique	MR	MP	HT	Complexity
Larger Block Size	+	-		0
Higher Associativity	+		-	1
Victim Caches	+			2
Pseudo-Associative Caches	+			2
HW Prefetching of Instr/Data	+	+		2
Compiler Controlled Prefetching	+	+		3
Compiler Reduce Misses				0
Priority to Read Misses		+		1
Early Restart & Critical Word 1st		+		2
Non-Blocking Caches		+		3
Second Level Caches		+		2
Better memory system		+		3
Small & Simple Caches	-		+	0
Avoiding Address Translation			+	2
Pipelining Caches			+	2

Comparing the 2 levels of hierarchy

Parameter	L1 Cache	Virtual Memory
Block/Page	16B – 128B	4KB – 64KB
Hit time	1 – 3 cc	100 – 200 cc
Miss Penalty (Access time) (Transfer time)	8 – 200 cc 6 – 160 cc 2 – 40 cc	1M – 10M cc (Page Fault) 800K – 8M cc 200K – 2M cc
Miss Rate	0.1 – 10%	0.00001 – 0.001%
Placement:	DM or N-way SA	Fully associative (OS allows pages to be placed anywhere in main memory)
Address Mapping	25-45 bit physical address to 14-20 bit cache address	32-64 bit virtual address to 25-45 bit physical address
Replacement:	LRU or Random (HW cntr.)	LRU (SW controlled)
Write Policy	WB or WT	WB

Paging vs. Segmentation: Pros and Cons

	Page	Segment
Words per address	One	Two (segment + offset)
Programmer visible?	Invisible to AP	May be visible to AP
Replacing a block	Trivial (all blocks are the same size)	Hard (must find contiguous, variable-size unused portion
Memory use inefficiency	Internal fragmentation (unused portion of page)	External fragmentation (unused pieces of main memory)
Efficient disk traffic	Yes (adjust page size to balance access time and transfer time)	Not always (small segments transfer few bytes)

Typical TLB Format

- Tag: Portion of virtual address
- Data: Physical Page number
- Dirty: since use write back, need to know whether or not to write page to disk when replaced
- Ref: Used to help calculate LRU on replacement
- Valid: Entry is valid
- Access rights: R (read permission), W (write perm.)

Virtual Addr.	Physical Addr.	Dirty	Ref	Valid	Access Rights

Techniques to exploit parallelism

Technique (Section in the textbook)	Reduces
Forwarding and bypassing	Data hazard (DH) stalls
Delayed branches	Control hazard stalls
Basic dynamic scheduling	DH stalls (RAW)
Dynamic scheduling with register renaming	WAR and WAW stalls
Dynamic branch prediction	CH stalls
Issuing multiple instruction per cycle	Ideal CPI
Speculation	Data and control stalls
Dynamic memory disambiguation	RAW stalls w. memory
Loop Unrolling	CH stalls
Basic compiler pipeline scheduling	DH stalls
Compiler dependence analysis	Ideal CPI, DH stalls
Software pipelining and trace scheduling	Ideal CPI and DH stalls
Compiler speculation	Ideal CPI, and D/CH stalls

Execution in Dual-issue Tomasulo Pipeline

Iter.	Inst.	Issue	Exe. (begins)	Mem. Access	Write at CDB	Com.
1	LD.D F0,0(R1)	1	2	3	4	first issue
1	ADD.D F4,F0,F2	1	5		8	Wait for LD.D
1	S.D 0(R1), F4	2	3	9		Wait for ADD.D
1	DADDIU R1,R1,-#8	2	4		5	Wait for ALU
1	BNE R1,R2,Loop	3	6			Wait for DAIDU
2	LD.D F0,0(R1)	4	7	8	9	Wait for BNE
2	ADD.D F4,F0,F2	4	10		13	Wait for LD.D
2	S.D 0(R1), F4	5	8	14		Wait for ADD.D
2	DADDIU R1,R1,-#8	5	9		10	Wait for ALU
2	BNE R1,R2,Loop	6	11			Wait for DAIDU
3	LD.D F0,0(R1)	7	12	13	14	Wait for BNE
3	ADD.D F4,F0,F2	7	15		18	Wait for LD.D
3	S.D 0(R1), F4	8	13	19		Wait for ADD.D
3	DADDIU R1,R1,-#8	8	14		15	Wait for ALU
3	BNE R1,R2,Loop	9	16			Wait for DAIDU

Four Steps of Speculative Tomasulo Algorithm

- 1. Issue—get instruction from FP Op Queue
  - If reservation station and reorder buffer slot free, issue instr & send operands & reorder buffer no. for destination (this stage sometimes called “dispatch”)
- 2. Execution—operate on operands (EX)
  - When both operands ready then execute; if not ready, watch CDB for result; when both in reservation station, execute; checks RAW (sometimes called “issue”)
- 3. Write result—finish execution (WB)
  - Write on Common Data Bus to all awaiting FUs & reorder buffer; mark reservation station available.
- 4. Commit—update register with reorder result
  - When instr. at head of reorder buffer & result present, update register with result (or store to memory) and remove instr from reorder buffer. Mispredicted branch flushes reorder buffer (sometimes called “graduation”)