Computer Science 146
Computer Architecture

Fall 2019

Harvard University

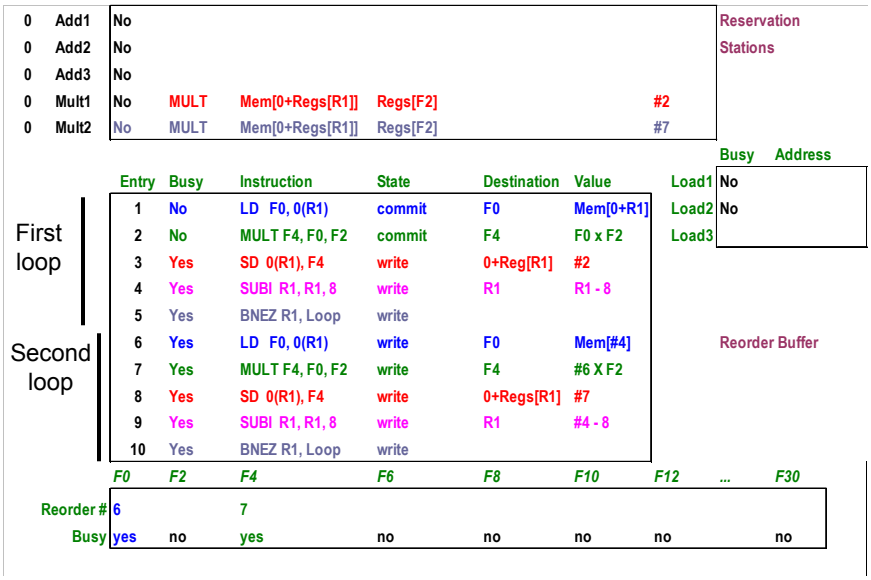
Instructor: Prof. David Brooks
dbrooks@eecs.harvard.edu

Lecture 9: Limits of ILP, Case Studies

Lecture Outline

- Speculative Execution
  - "Implementing Precise Interrupts in Pipelined Processors" – J.E. Smith and A. Pleszkun (Trans. Computers '88)
  - "Instruction Issue Logic for High-Performance Interruptable Pipelined Processors" – G. Sohi and S. Vajapeyam (ISCA '87)
  - Tomasulo with ROB example
  - Pointer Based Methods
- SuperScalar/ILP Limits
  - "Limits of ILP," – D. Wall (ASPLOS'91)
  - "Complexity Effective Superscalar Design," – S. Palacharla, N. Jouppi, J.E. Smith (ISCA'97)
- Case Studies
  - Pentium III
  - Pentium 4
    - "Trace Cache: A Low Latency Approach to High Bandwidth Instruction Fetching," E. Rotenberg, S. Bennett, J.E. Smith (MICRO '96)

Example of Speculative State of Reorder Buffer



0	Add1	No					Reservation Stations
0	Add2	No					
0	Add3	No					
0	Mult1	No	MULT	Mem[0+Regs[R1]]	Regs[F2]	#2	
0	Mult2	No	MULT	Mem[0+Regs[R1]]	Regs[F2]	#7	

	Busy	Address
Load1	No	
Load2	No	
Load3		

	Entry	Busy	Instruction	State	Destination	Value
First loop	1	No	LD F0, 0(R1)	commit	F0	Mem[0+R1]
	2	No	MULT F4, F0, F2	commit	F4	F0 x F2
	3	Yes	SD 0(R1), F4	write	0+Reg[R1]	#2
	4	Yes	SUBI R1, R1, 8	write	R1	R1 - 8
	5	Yes	BNEZ R1, Loop	write		
Second loop	6	Yes	LD F0, 0(R1)	write	F0	Mem[#4]
	7	Yes	MULT F4, F0, F2	write	F4	#6 X F2
	8	Yes	SD 0(R1), F4	write	0+Regs[R1]	#7
	9	Yes	SUBI R1, R1, 8	write	R1	#4 - 8
	10	Yes	BNEZ R1, Loop	write		

Reorder Buffer

	F0	F2	F4	F6	F8	F10	F12	...	F30
Reorder #	6		7						
Busy	yes	no	yes	no	no	no	no		no

Multiply has just reached commit, so other instructions can start committing

Tomasulo + ROB Summary

- Many implementations are very similar
  - Pentium III, PowerPC, etc
- Some limitations
  - Too many value copy operations
    - Register file => RS => ROB => Register File
  - Too many muxes/busses (CDB)
    - Values are coming from everywhere to everywhere else!
  - Reservation Stations mix values(data) and tags(control)
    - Slows down the max clock frequency

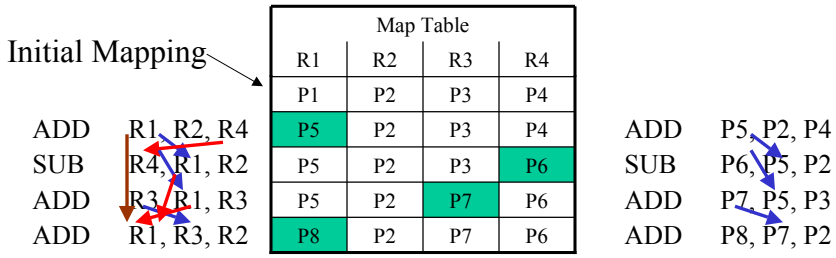
Alternative to ROB

- Separate control (ROB/RS) from data (RegFile)
- Store all data in physical register file

MIPS R10K Register Renaming

- Architectural Register file is removed
- Physical Register file holds all values
  - #Physical Register = #Architectural Registers + #ROB entries
  - Map architectural registers to physical registers
  - Removes WAW, WAR hazards
    - Physical registers replace RS
- Register Status Table replaced by Register Map Table
  - But all registers must be mapped somewhere!
- Free List tracks unallocated registers
  - ROB returns physical registers to free list

MIPS R10K: Register Map Table



MIPS R10K: How to free registers?

- Old Method (Tomasulo + Reorder Buffer)
  - Don't free speculative storage explicitly
  - At Retire:
    - Copy value from ROB to register file, free ROB entry
- MIPS R10K
  - Can't free physical register when instructions retire
    - There is no architectural register to copy to
  - Free physical register previously mapped to same logical register
  - All instructions that will read it have retired already

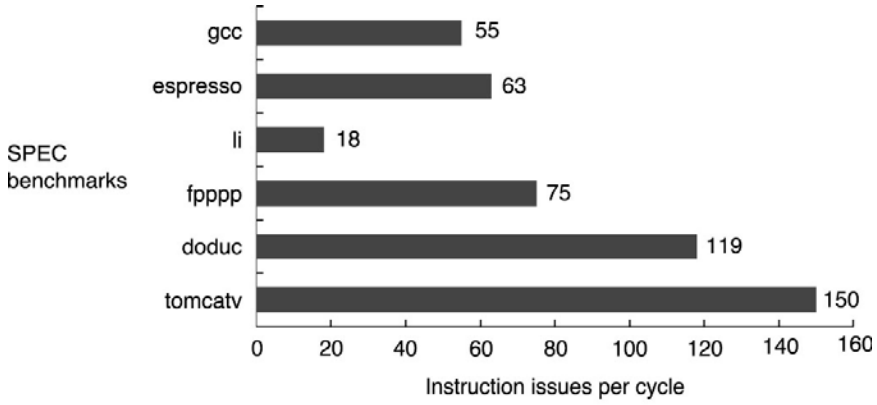
MIPS R10K Precise State

- Physical registers are written at commit
  - No architectural register file to update
  - "Free" written registers and "restore" old ones
- Restore register map table to the way it was
- Choose to:
  - Roll back ROB serially
  - Restore from checkpoints
    - Checkpoint Cache
    - MIPS R10K can only speculate past 4 branches, 4 checkpoints

MIPS R10K vs. Pentium III

Feature	Pentium III	MIPS R10K
Value Storage	Architectural Register File, ROB, Reservation Station	Physical Register File
Reg. Read	On Issue, Write into RS	On Execute, to FU
Reg. Write	On Commit, from ROB	On Writeback from FU
Reg. Free	Instruction Commits	Overwriting instruction retires
Precise State	Simple → Reset Structures	Complex Checkpoints

Limits on ILP/SuperScalars: Perfect Processor

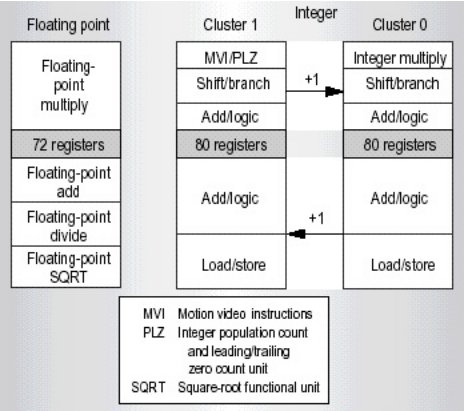


- What limits?

Implementation Issues: SuperScalar Width Implications

- Wide Fetch
  - Must predict multiple branches (what if you have two taken branches!)
- Wide Decode
  - Ok for fixed width ISAs -- What about variable length?
- Wide Rename
  - Complexity grows with N^2
- Wide issue and bypass
  - Many tag checks needed, muxes, datapaths

Implementation Issues: Alpha 21264

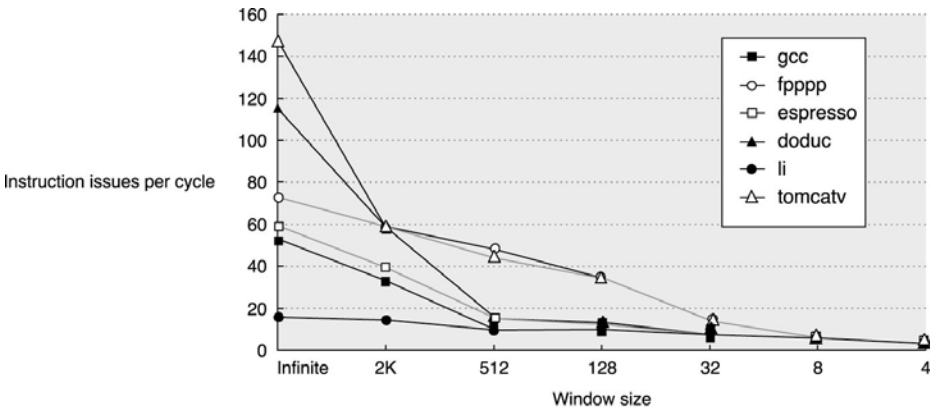


- 6-way superscalar processor (4-way integer)
- Intercluster communication:
  - 1 Cycle Latency

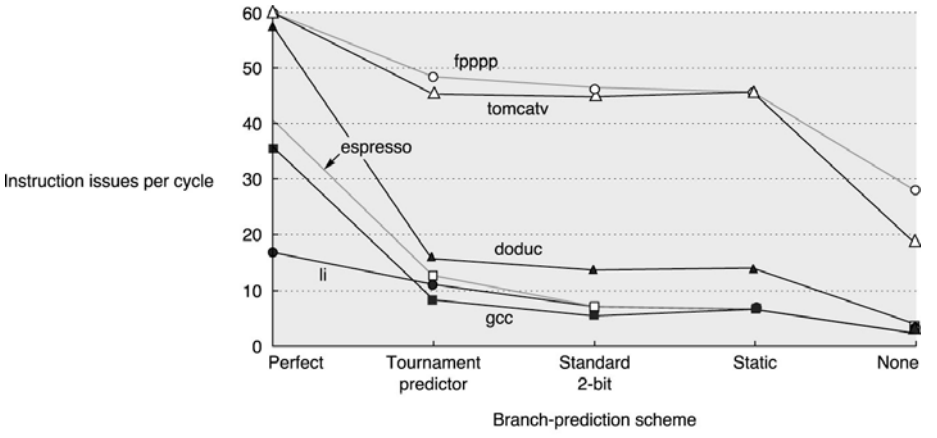
Implementation Issues: Reservation Station Design

- Logical Structure
  - Unified
    - Better Utilization, More Complex Logic (multiported)
    - Pentium III – Unified RS Queue
  - Distributed
    - Worse utilization, Simpler logic
    - MIPS R10K – Three RS Queues (Int, Mem, FP)
- Physical Implementation
  - FIFO vs. RAM

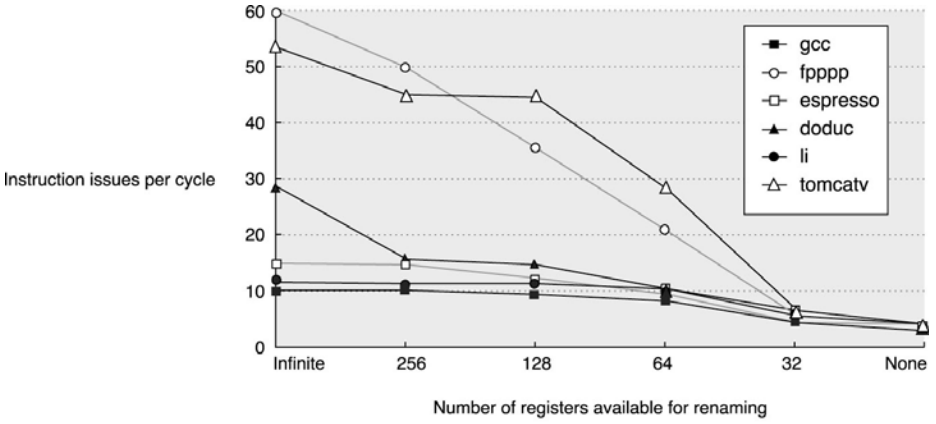
Limits on ILP: Instruction Window Size



Limits on ILP: Realistic Branch Prediction



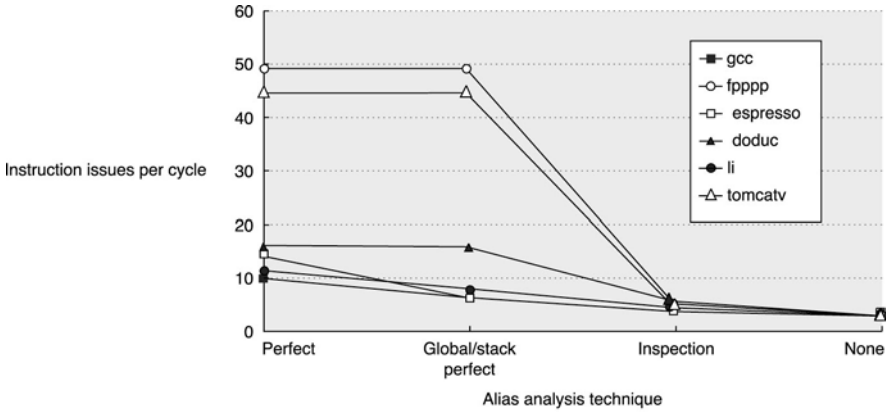
Limits on ILP: Rename Registers



Limits on ILP: Load/Store Disambiguation

- "Alias analysis" problem
  - How do we analyze dependencies through memory?
- Compiler Solutions
  - Examine Registers + base offsets to check for conflicts
- Hardware Solutions
  - In-order load/stores (slow!)
  - Loads in-order with other stores, but not loads
  - Loads issue out of order, cleanup mis-speculations (complex)
  - Predictors to choose from above policies

Limits on ILP: Load/Store Disambiguation



Dynamic Scheduling in P6 (Pentium Pro, II, III)

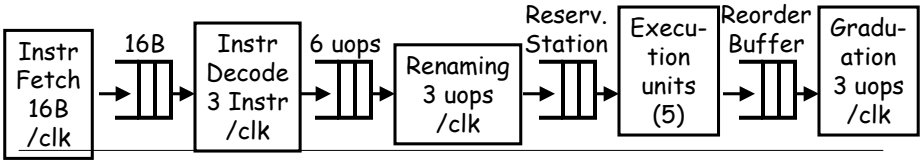
- Q: How pipeline 1 to 17 byte 80x86 instructions?
- P6 doesn't pipeline 80x86 instructions
- P6 decode unit translates the Intel instructions into 72-bit micro-operations (~ MIPS)
- Sends micro-operations to reorder buffer & reservation stations
- Many instructions translate to 1 to 4 micro-operations
- Complex 80x86 instructions are executed by a conventional microprogram (8K x 72 bits) that issues long sequences of micro-operations
- 14 clocks in total pipeline (~ 3 state machines)

Dynamic Scheduling in P6

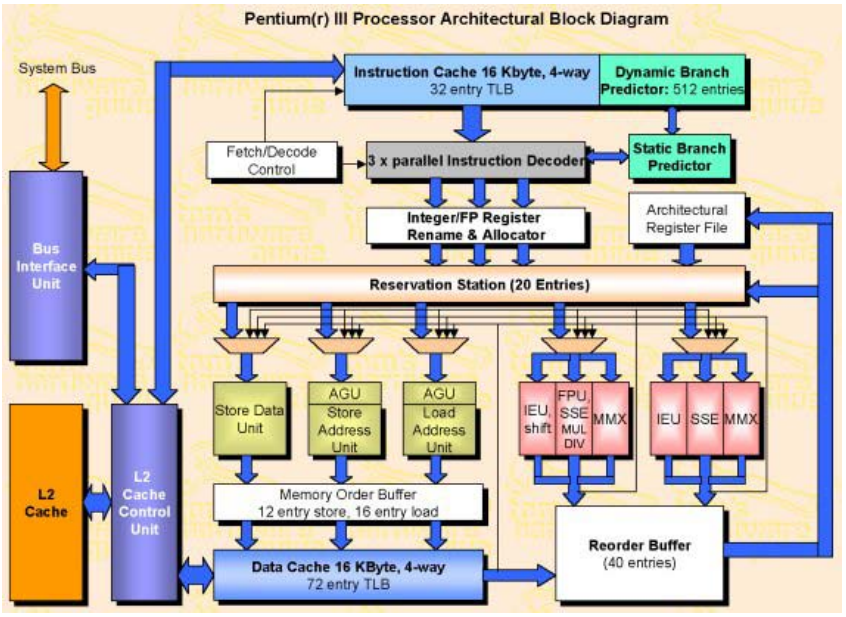
Parameter	80x86	microops
Max. instructions issued/clock	3	6
Max. instr. complete exec./clock		5
Max. instr. committed/clock		3
Window (Instrs in reorder buffer)		40
Number of reservations stations	20	
Number of rename registers	40	
No. integer functional units (FUs)	2	
No. floating point FUs	1	
No. SIMD Fl. Pt. FUs1		
No. memory Fus	1 load + 1 store	

P6 Pipeline

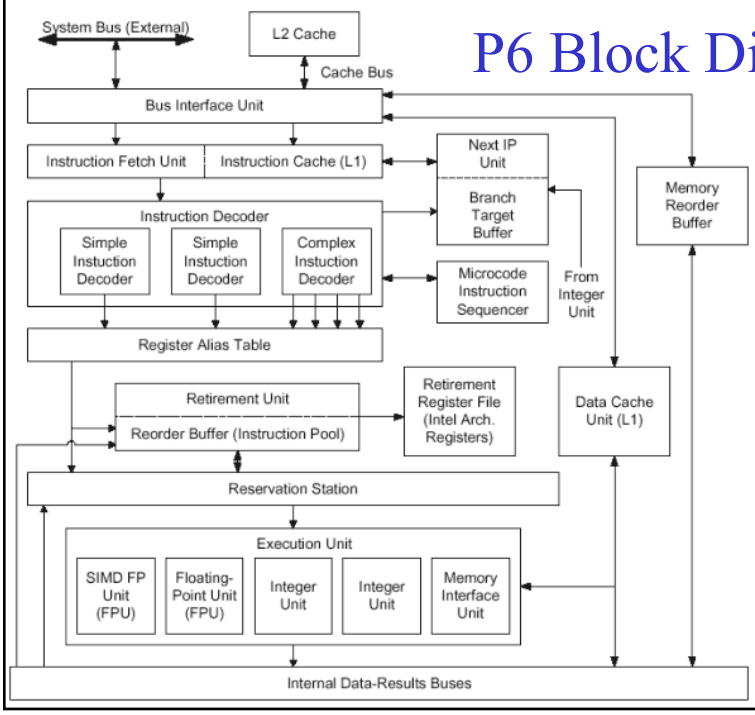
- 8 stages are used for in-order instruction fetch, decode, and issue
  - Takes 1 clock cycle to determine length of 80x86 instructions + 2 more to create the micro-operations (uops)
- 3 stages are used for out-of-order execution in one of 5 separate functional units
- 3 stages are used for instruction commit



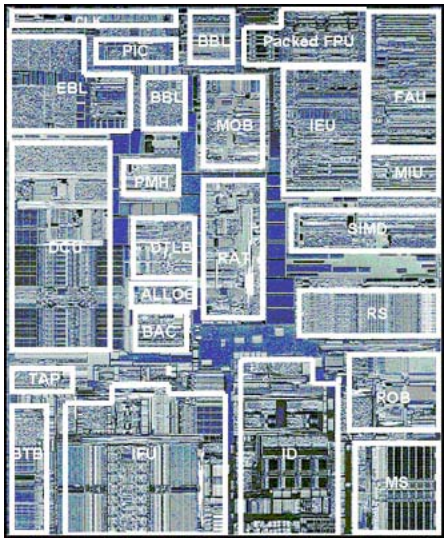
Pentium III Overview



P6 Block Diagram



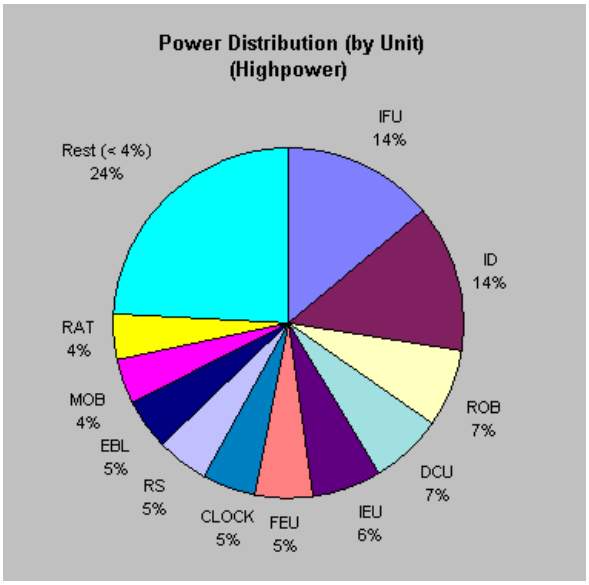
Pentium III Die Photo



- EBL/BBL - Bus logic, Front, Back
- MOB - Memory Order Buffer
- Packed FPU - MMX Fl. Pt. (SSE)
- IEU - Integer Execution Unit
- FAU - Fl. Pt. Arithmetic Unit
- MIU - Memory Interface Unit
- DCU - Data Cache Unit
- PMH - Page Miss Handler
- DTLB - Data TLB
- BAC - Branch Address Calculator
- RAT - Register Alias Table
- SIMD - Packed Fl. Pt.
- RS - Reservation Station
- BTB - Branch Target Buffer
- IFU - Instruction Fetch Unit (+I$)
- ID - Instruction Decode
- ROB - Reorder Buffer
- MS - Micro-instruction Sequencer

1st Pentium III, Katmai: 9.5 M transistors, 12.3 x 10.4 mm, 250 nm CMOS with 5 layers of Al

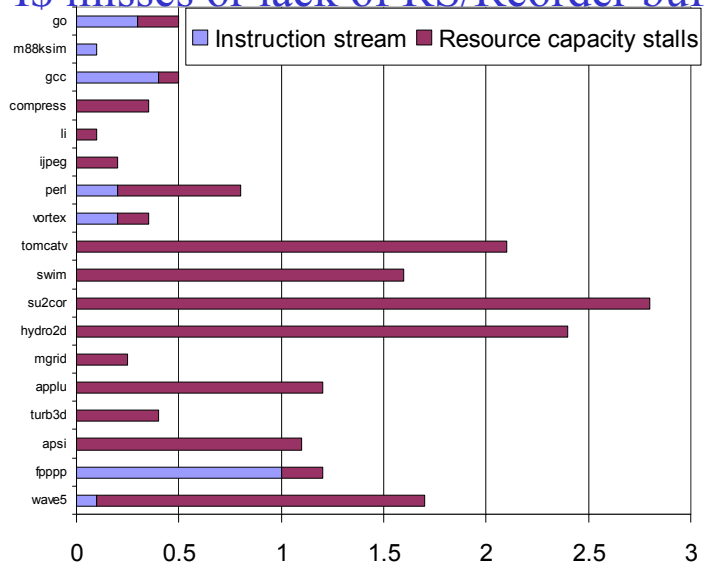
Pentium III Power Dissipation



Max: ~20W
Typical: ~15W

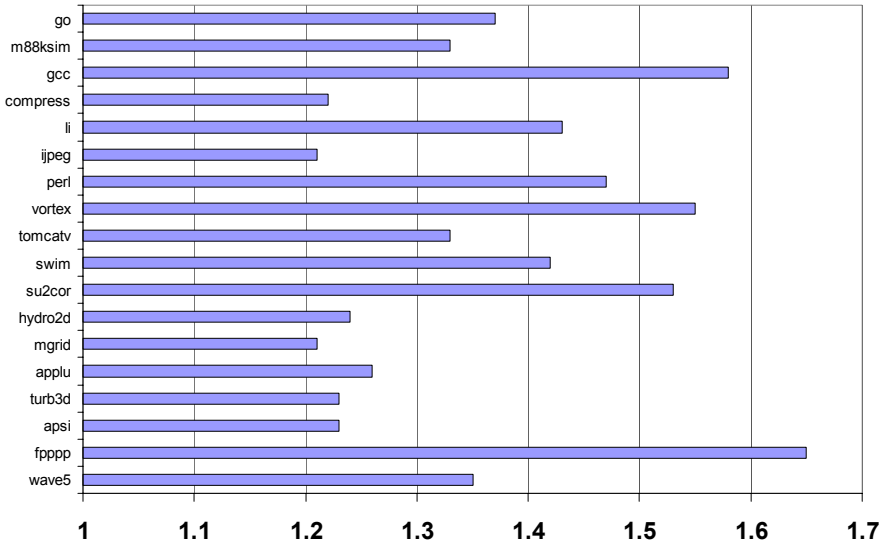
P6 Performance: Stalls at decode stage

I$ misses or lack of RS/Reorder buf. entry



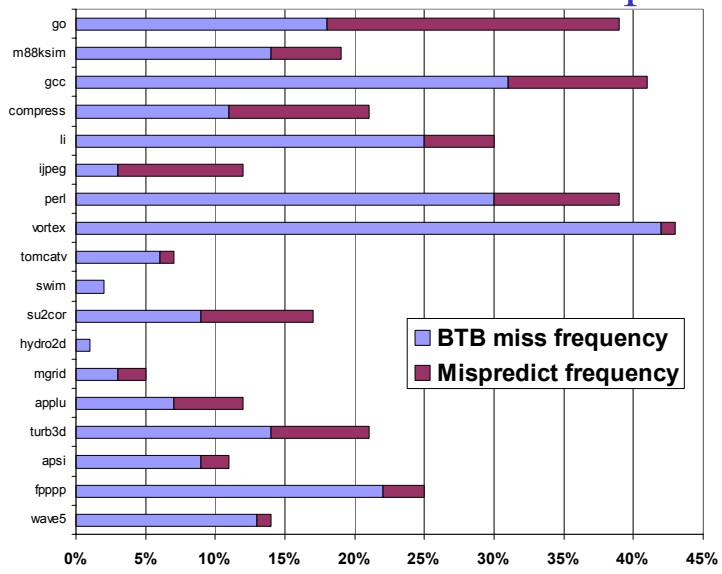
0.5 to 2.5 Stall cycles per instruction: 0.98 avg. (0.36 integer)

P6 Performance: uops/x86 instr



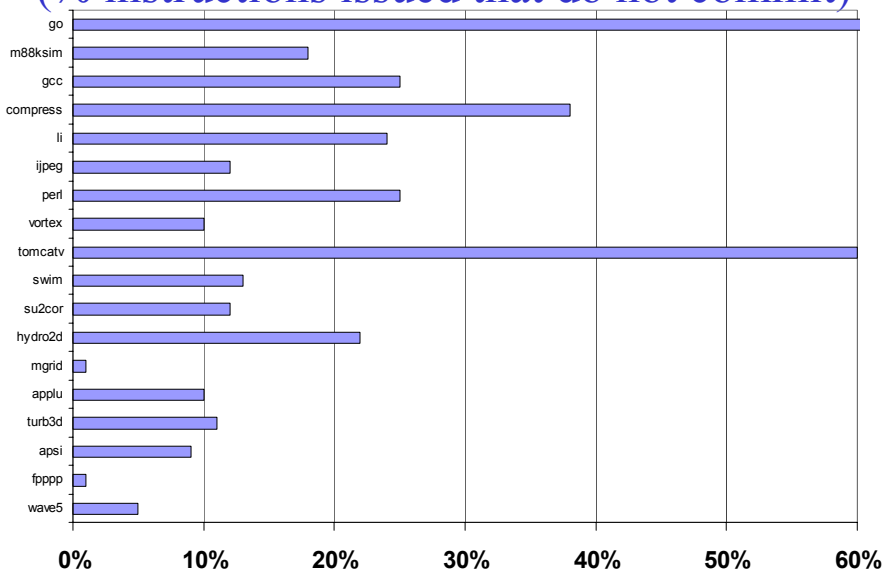
1.2 to 1.6 uops per IA-32 instruction: 1.36 avg. (1.37 integer)

P6 Performance: Branch Mispredict Rate



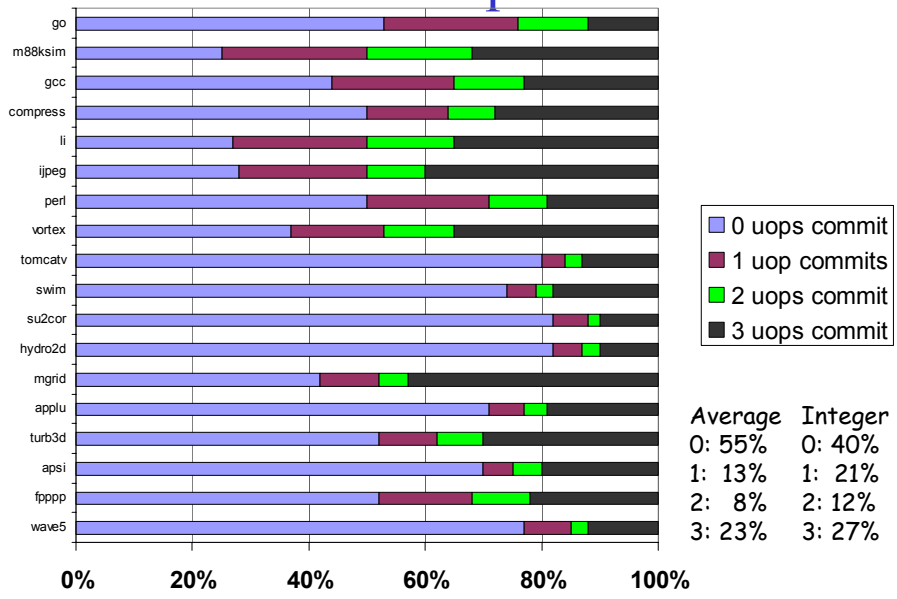
10% to 40% Miss/Mispredict ratio: 20% avg. (29% integer)

P6 Performance: Speculation rate
(% instructions issued that do not commit)



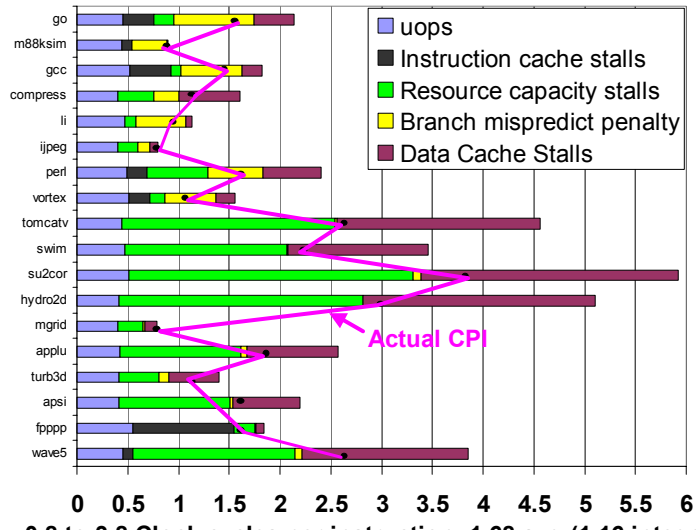
1% to 60% instructions do not commit: 20% avg (30% integer)

P6 Performance: uops commit/clock



Average	Integer
0: 55%	0: 40%
1: 13%	1: 21%
2: 8%	2: 12%
3: 23%	3: 27%

P6 Dynamic Benefit?
Sum of parts CPI vs. Actual CPI



0.8 to 3.8 Clock cycles per instruction: 1.68 avg (1.16 integer)

Ratio of sum of parts vs. actual CPI: 1.38X avg. (1.29X integer)

Pentium 4

- Still translate from 80x86 to micro-ops
- P4 has better branch predictor, more FUs
- Instruction Cache holds micro-operations vs. 80x86 instructions
  - no decode stages of 80x86 on cache hit ("Trace Cache")
- Faster memory bus: 400 MHz v. 133 MHz
- Caches
  - Pentium III: L1I 16KB, L1D 16KB, L2 256 KB
  - Pentium 4: L1I 12K uops, L1D 8 KB, L2 256 KB
  - Block size: PIII 32B v. P4 128B; 128 v. 256 bits/clock
- Clock rates:
  - Pentium III 1 GHz v. Pentium IV 1.5 GHz
  - 14 stage pipeline vs. 24 stage pipeline

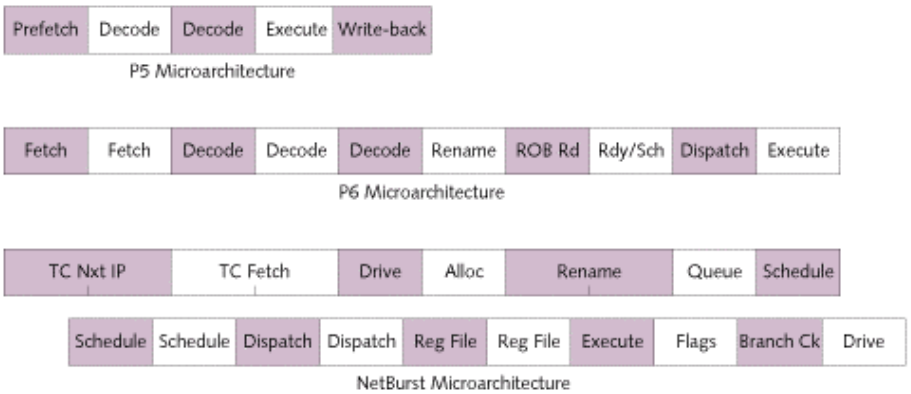
Trace Cache

- IA-32 instructions are difficult to decode
- Conventional Instruction Cache
  - Provides instructions up to and including taken branch
- Trace cache, records uOps instead of x86 Ops
- Builds them into groups of six sequentially ordered uOps per line
  - Allows more ops per line
  - Avoids clock cycle to get to target of branch

Pentium 4 features

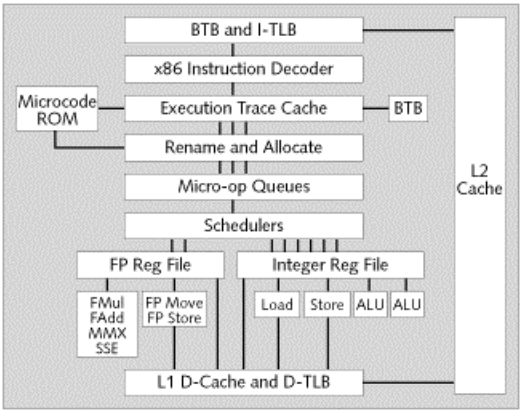
- Multimedia instructions 128 bits wide vs. 64 bits wide => 144 new instructions
  - When used by programs??
  - Faster Floating Point: execute 2 64-bit Fl. Pt. Per clock
  - Memory FU: 1 128-bit load, 1 128-store /clock to MMX regs
- Using RAMBUS DRAM
  - Bandwidth faster, latency same as SDRAM
  - Cost 2X-3X vs. SDRAM
- ALUs operate at 2X clock rate for many ops
- Pipeline doesn't stall at this clock rate: uops replay
- Rename registers: 40 vs. 128; Window: 40 v. 126
- BTB: 512 vs. 4096 entries (Intel: 1/3 improvement)

Pentium, Pentium Pro, P4 Pipeline



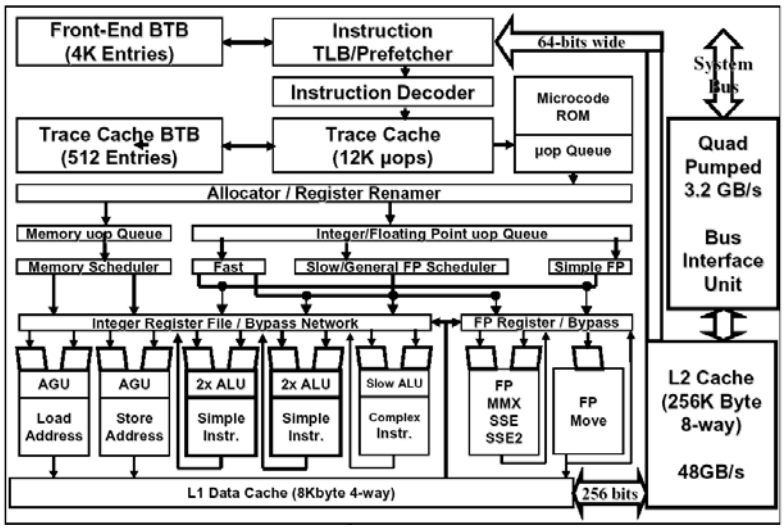
- Pentium (P5) = 5 stages
  Pentium Pro, II, III (P6) = 10 stages (1 cycle ex)
  Pentium 4 (NetBurst) = 20 stages (no decode)

Block Diagram of Pentium 4 Microarchitecture

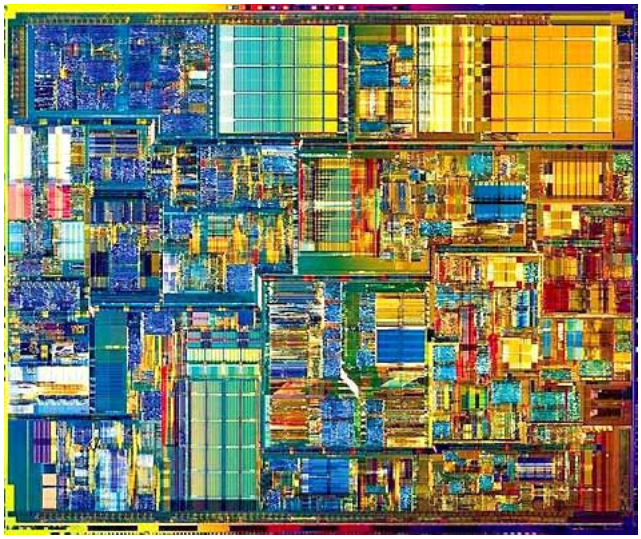


- BTB = Branch Target Buffer (branch predictor)
- I-TLB = Instruction TLB, Trace Cache = Instruction cache
- RF = Register File; AGU = Address Generation Unit
- "Double pumped ALU" means ALU clock rate 2X => 2X ALU F.U.s

Pentium 4 Block Diagram

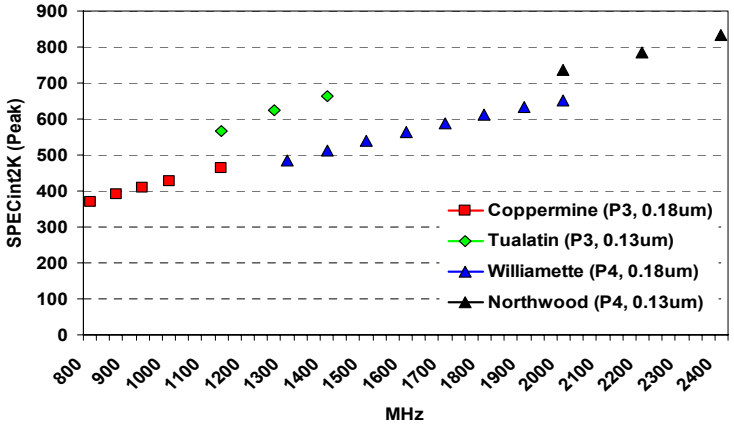


Pentium 4 Die Photo

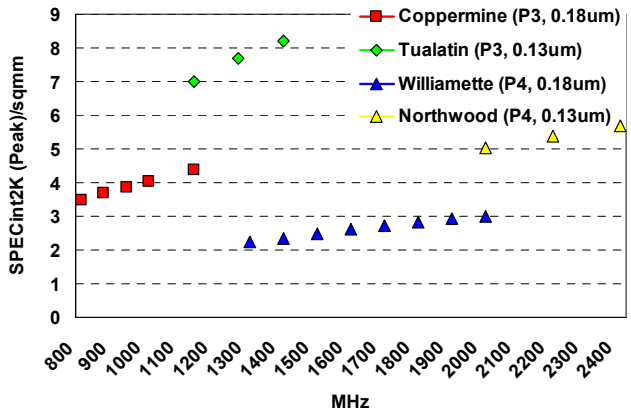


- 42M xistors
  - PIII: 26M
- 217 mm^2
  - PIII: 106 mm^2
- L1 Execution Cache
  - Buffer 12,000 Micro-Ops
- 8KB data cache
- 256KB L2$

Pentium III vs. Pentium 4: Performance



Pentium III vs. Pentium 4: Performance / mm^2



Williamette: 217mm^2, Northwood: 146mm^2, Tualatin: 81mm^2, Coppermine: 106mm^2

For next time

- Static Scheduling