Processor: Superscalars – Case Studies: Intel P6, Pentium 4, Itanium, and IBM PowerPC 620

Z. Jerry Shi
Assistant Professor of Computer Science and Engineering
University of Connecticut

* Slides adapted from Blumrich&Gschwind/ELE475'03, Peh/ELE475'*

Intel P6 Microarchitecture In a Nutshell

- 1995 product → Pentium Pro, Pentium II, Pentium III, ideas reused in Pentium 4.
- Application of concepts we learnt in class:
  - 3-way superscalar
  - OOO, speculative
  - Micro-ops
  - Branch prediction with BTB
  - Reservation stations
  - Renaming through Reorder Buffer

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

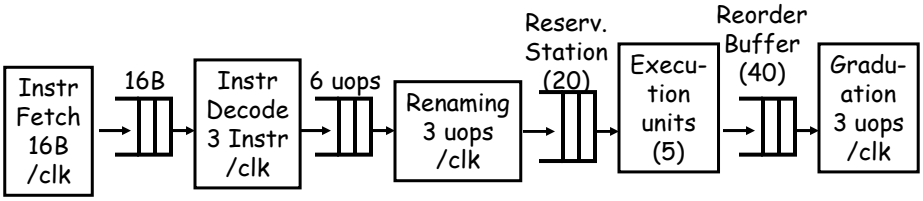
- Q: How pipeline 1 to 17 byte 80x86 instructions?
- A: P6 doesn't pipeline 80x86 instructions
- P6 decode unit translates the Intel instructions into 72-bit micro-operations (~ MIPS)
- Sends micro-operations to reorder buffer & reservation stations
- Many instructions translate to 1 to 4 micro-operations
- 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

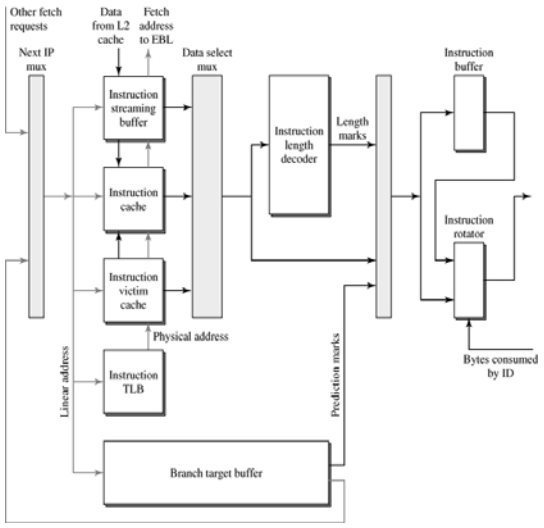
Parameter	80x86	microops
Max. instructions issued/clock	3	6
Max. instr. complete exec./clock		5
Max. instr. committed/clock		3
Window (Instrs in reorder buffer)		40
Number of reservations stations		20
Number of rename registers		40
Number of integer functional units (FUs)		2
Number of floating point FUs		1
Number of SIMD floating point FUs		1
Number of memory FUs		1 load + 1 store

P6 Pipeline

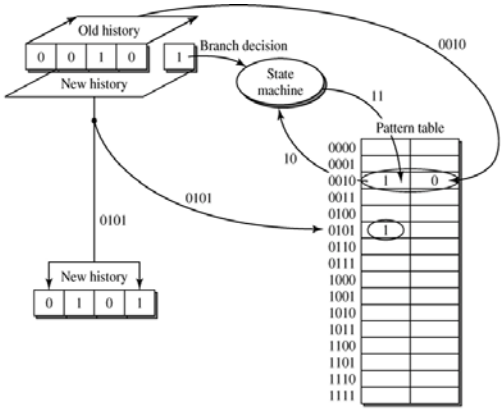
- 8 stages are used for in-order instruction fetch, decode, and issue
  - Takes 1 clock cycle to determine length of 80x86 instructions + 2 more to create the micro-operations (uops)
- 3 stages are used for out-of-order execution in one of 5 separate functional units
  - Integer, FP, branch, memory address,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



Instruction Fetch



Branch Prediction

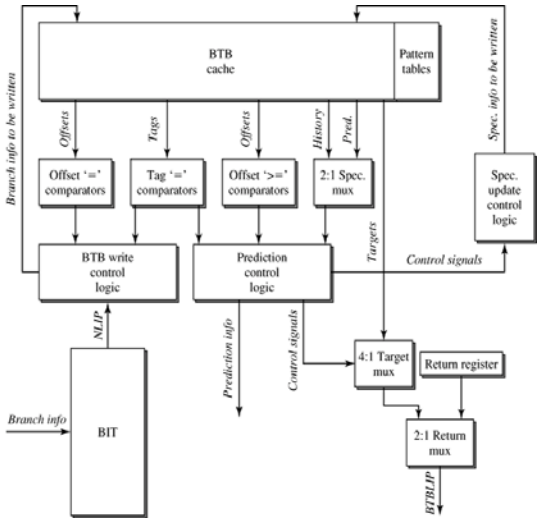


Two processes occur in parallel:

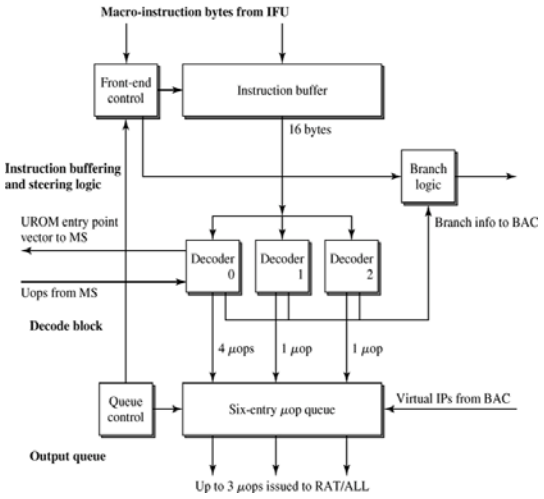
1. The new history is used to access the pattern table to get the new prediction bit. This prediction bit is written into the BTB in the next phase.

2. The old history is used to access the pattern table to get the state that has to be updated. The updated state is then written back to the pattern table.

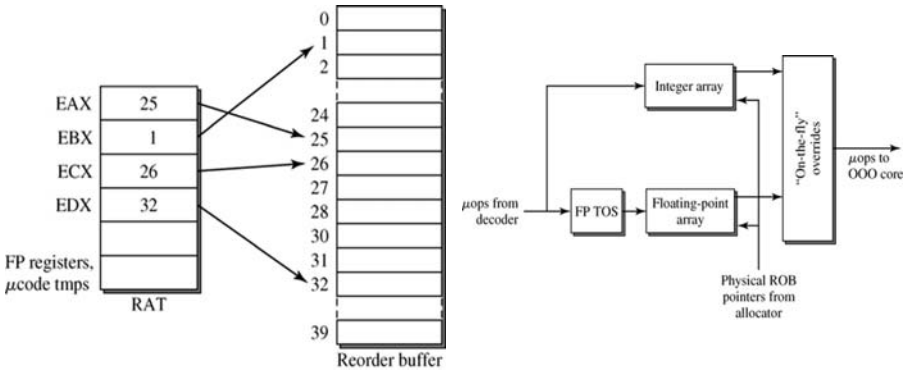
Branch Target Buffer



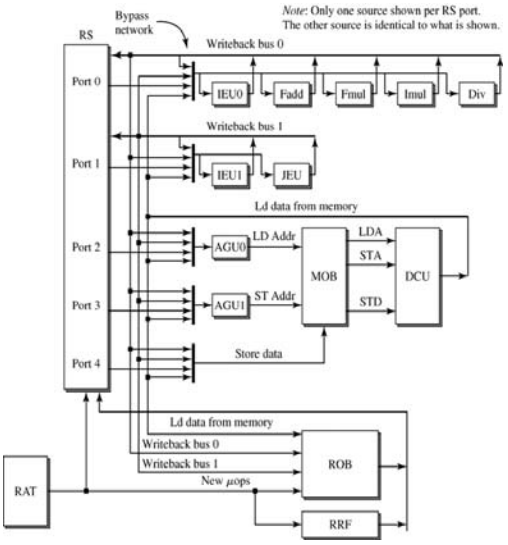
Instruction Decode



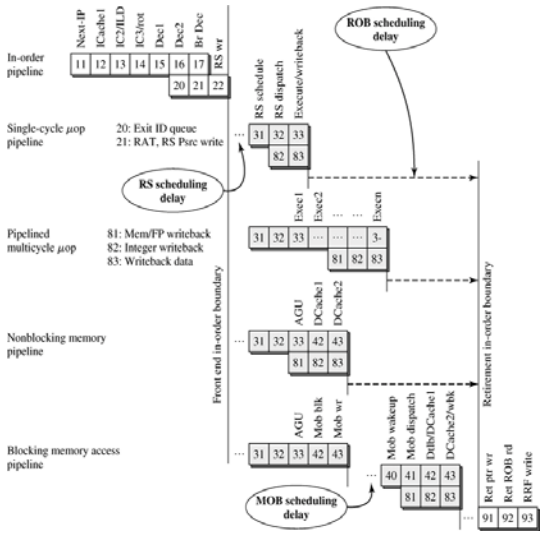
Register Renaming: Register Alias Table



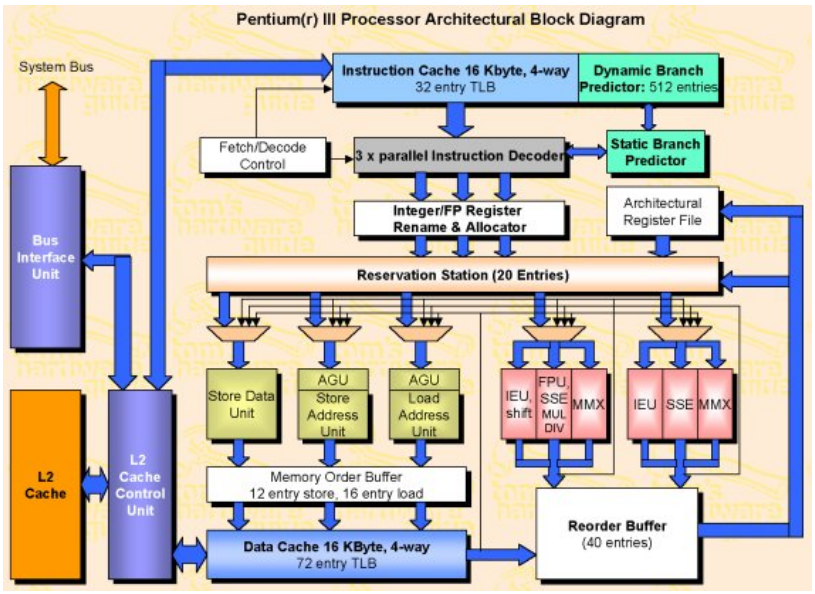
Instruction Dispatch & Execute: Reservation Stations



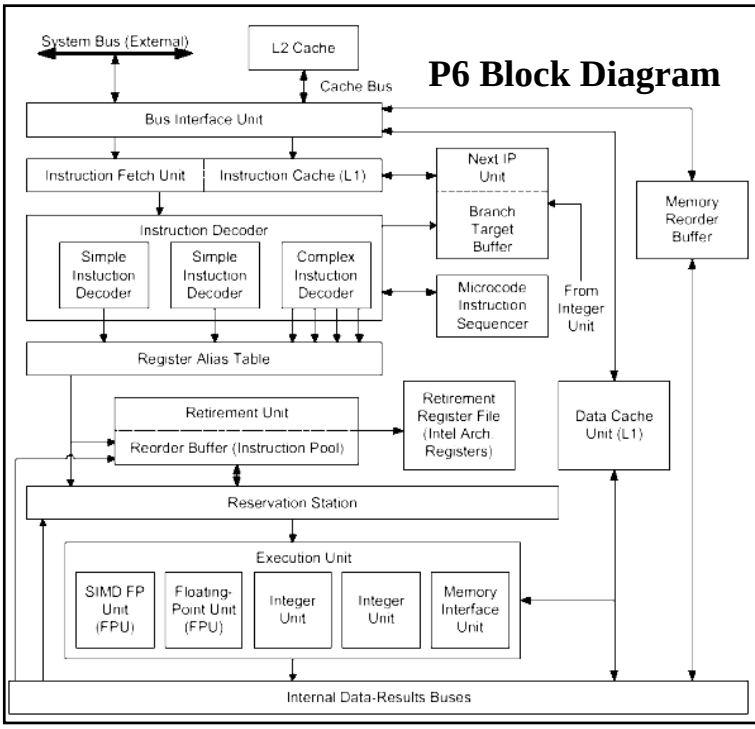
P6 Pipeline (Look at ROB stages)



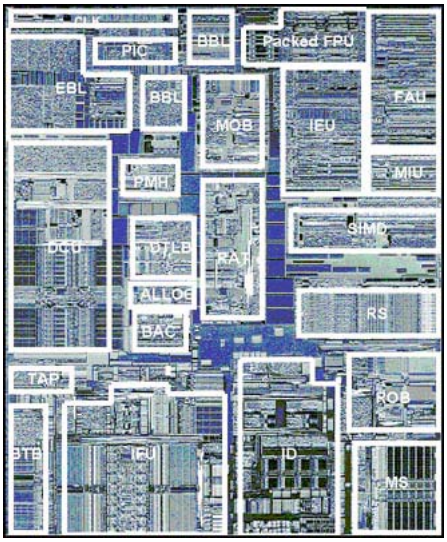
Pentium III Overview



P6 Block Diagram



Pentium III Die Photo

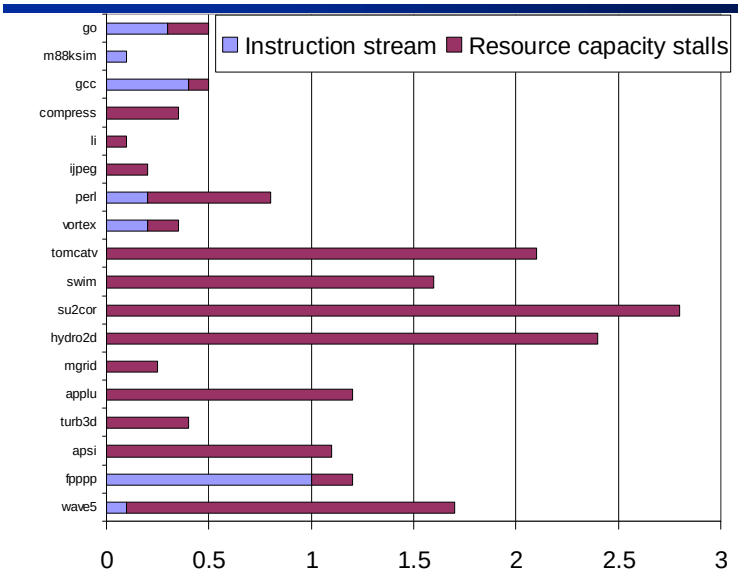


1st Pentium III, Katmai: 9.5 M transistors, 12.3 x 10.4 mm, 250 nm CMOS with 5 layers of Al

- 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

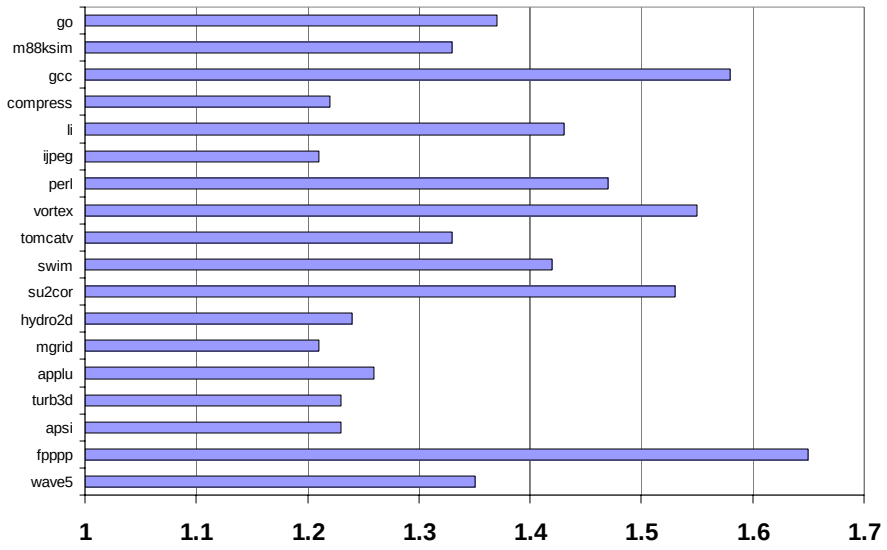
P6 Performance: Stalls at decode stage

I-cache misses or lack of RS/Reorder buf. entry



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

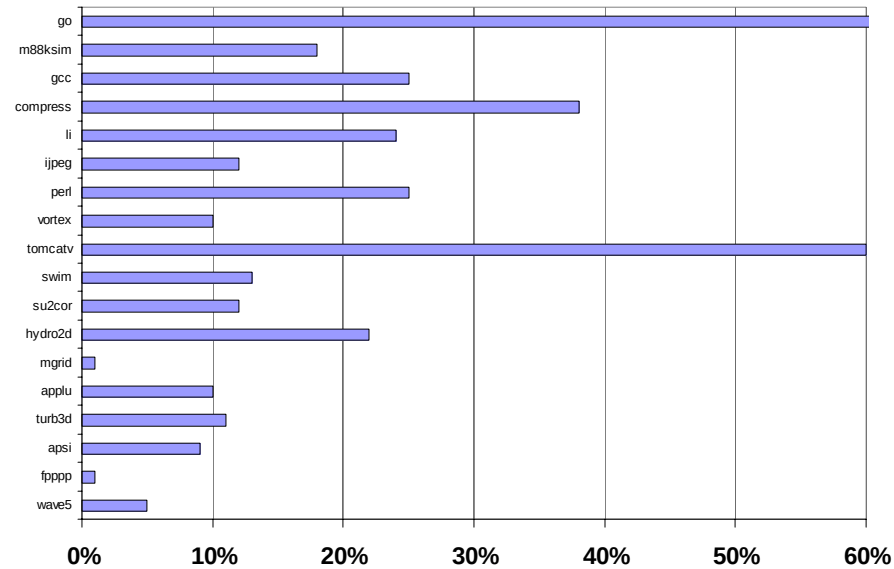
P6 Performance: uops/x86 instr



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

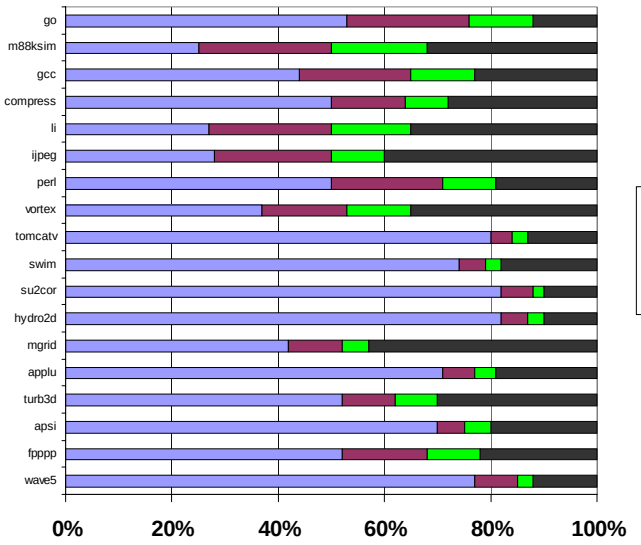
P6 Performance: Speculation rate

(% instructions issued that do not commit)



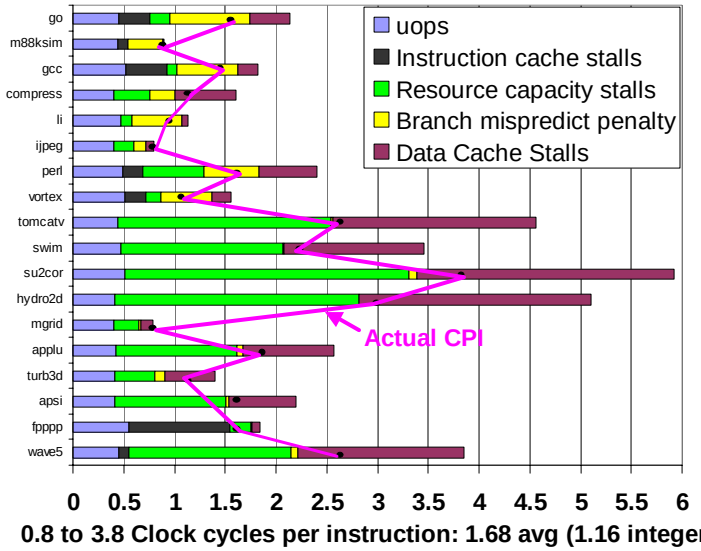
1% to 60% instructions do not commit: 20% avg (30% integer)

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 functional units
- 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: L1-I 16KB, L1-D 16KB, L2 256 KB
  - Pentium 4: L1-I 12K uops, L1-D 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

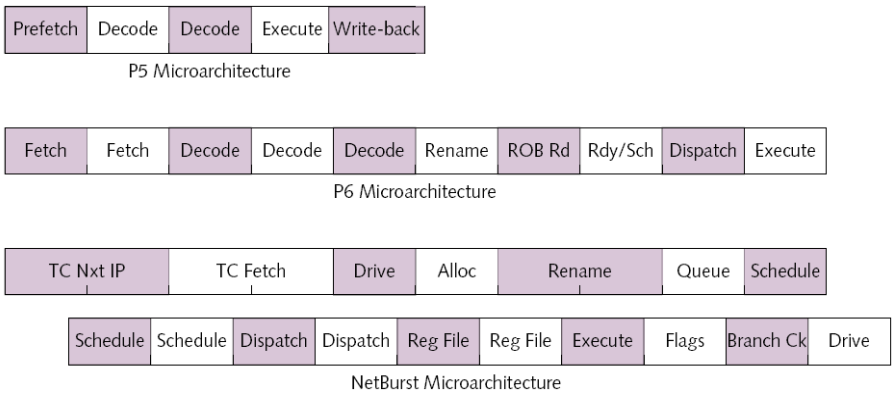
Trace Cache

- 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

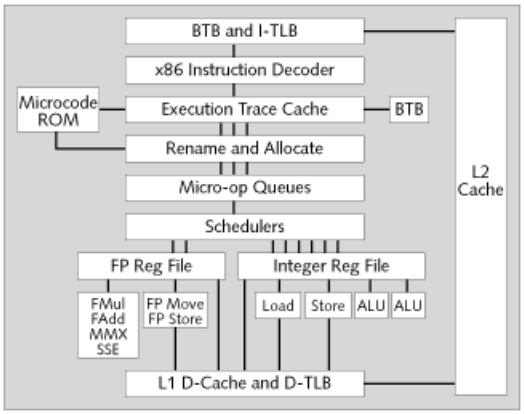
- Multimedia instructions 128 bits wide vs. 64 bits wide => 144 new instructions
  - When used by programs??
  - Faster Floating Point: execute 2 64-bit Fl. Pt. Per clock
  - Memory FU: 1 128-bit load, 1 128-store /clock to MMX regs
- Using RAMBUS DRAM
  - Bandwidth faster, latency same as SDRAM
  - 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



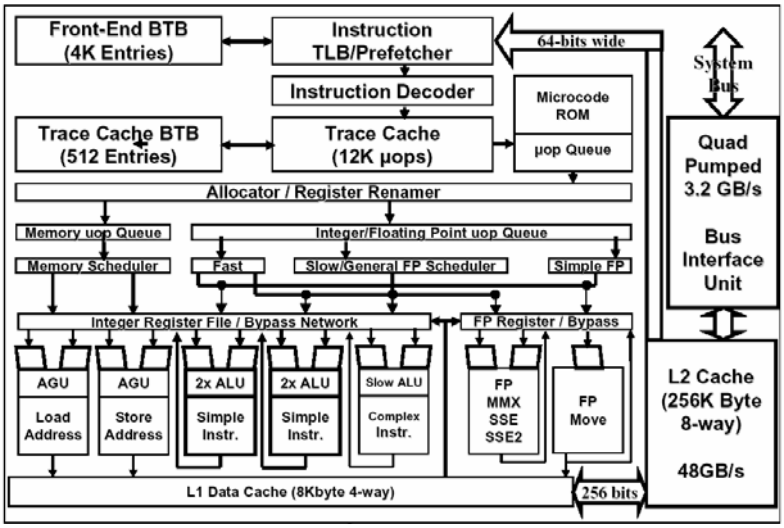
- P.N. Glaskowsky, "Pentium 4 (partially) previwed," MPR, August 2000.
- Pentium (P5) = 5 stages
  Pentium Pro, II, III (P6) = 10 stages (1 cycle ex)
  Pentium 4 (NetBurst) = 20 stages (no decode)

Block Diagram of Pentium 4 Microarchitecture

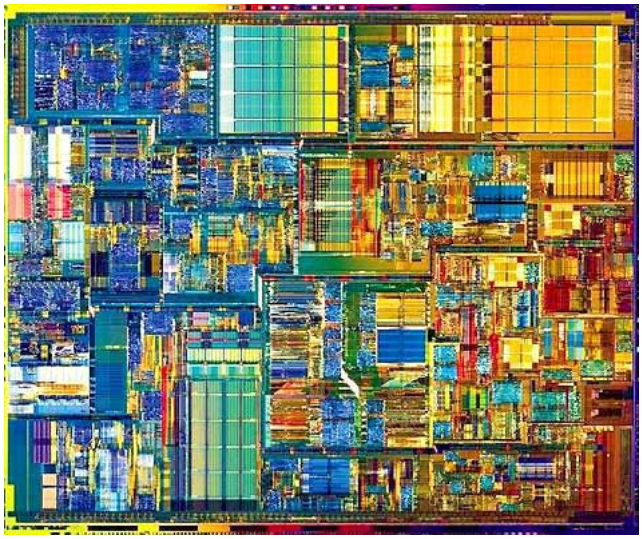


- BTB = Branch Target Buffer (branch predictor)
- I-TLB = Instruction TLB, Trace Cache = Instruction cache
- RF = Register File; AGU = Address Generation Unit
- "Double pumped ALU" means ALU clock rate 2X => 2X ALU F.U.s

Pentium 4 Block Diagram

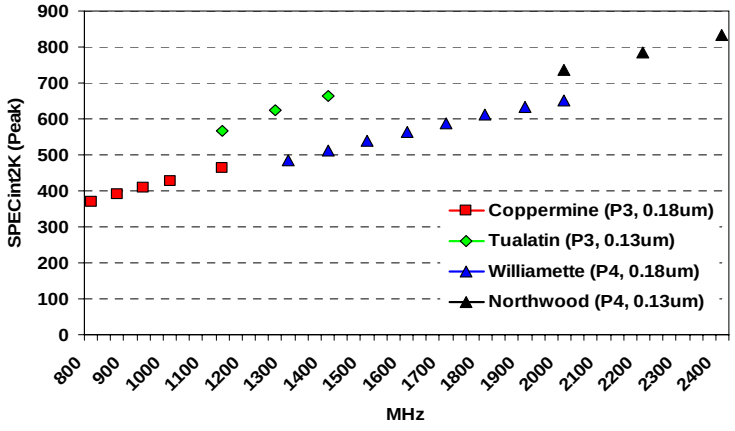


Pentium 4 Die Photo

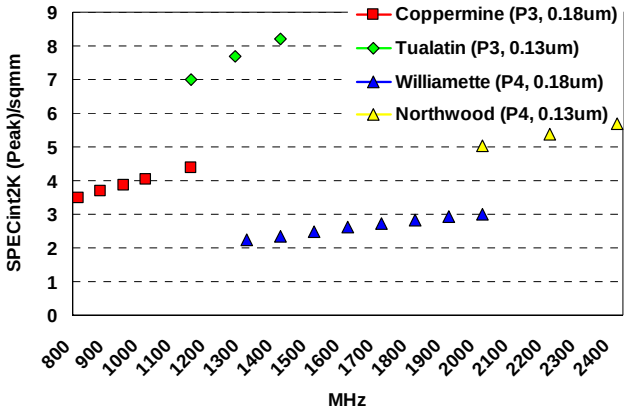


- 42M xistors
  - PIII: 26M
- 217 mm^2
  - PIII: 106 mm^2
- L1 Execution Cache
  - Buffer 12,000 Micro-Ops
- 8KB data cache
- 256KB L2$

Pentium III vs. Pentium 4: Performance



Pentium III vs. Pentium 4: Performance / mm²



Williamette: 217mm², Northwood: 146mm², Tualatin: 81mm², Coppermine: 106mm²

Intel/HP IA-64: Explicitly Parallel Instruction Computer (EPIC)

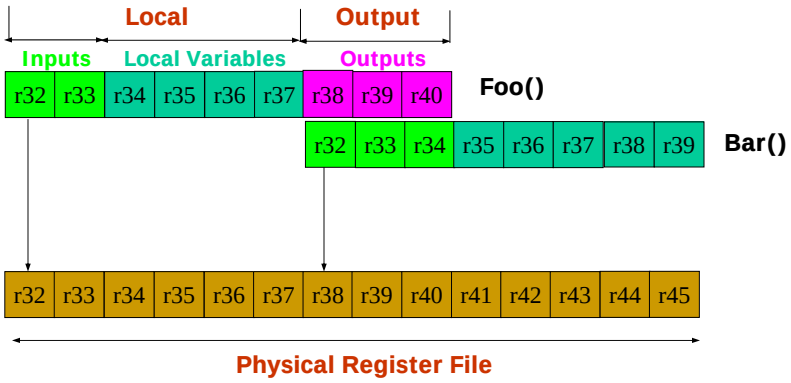
- IA-64: instruction set architecture; EPIC is type
  - EPIC = 2nd generation VLIW?
- Itanium™ is name of first implementation (June 2001)
  - Highly parallel and deeply pipelined hardware at 800Mhz
  - 6-wide, 10-stage pipeline at 800Mhz on 0.18 µ process
- Itanium2 – Sept 2002 (1GHz), Sept 2003 (1.5GHz)
- 128 64-bit integer registers + 128 82-bit floating point registers
- Hardware checks some dependencies (interlocks => binary compatibility over time)
- Predicated execution (select 1 out of 64 1-bit flags)

IA-64 Registers

- The integer registers designed to assist procedure calls using a register stack
  - Similar to SPARC's register windows.
  - Registers 0-31 are always accessible and addressed as 0-31
  - Registers 32-128 are used as a register stack and each procedure is allocated a set of registers (from 0 to 96)
  - The new register stack frame is created for a called procedure by renaming the registers in hardware;
  - a special register called the current frame pointer (CFM) points to the set of registers to be used by a given procedure
- 8 64-bit Branch registers used to hold branch destination addresses for indirect branches
- 64 1-bit predicate registers

Register Stack Frame Mapping

- Example (from Settle, et al., CGO'03)



IA-64 Registers

- Both the integer and floating point registers support register rotation for registers 32-128.
- Register rotation eases the task of allocating registers in software pipelined loops
- Avoid the need for unrolling and for prologue and epilogue code for a software pipelined loop
  - Makes the SW-pipelining usable for loops with smaller numbers of iterations

Explicitly Parallel Instruction Computer (EPIC)

- Instruction group: a sequence of consecutive instructions with no register data dependences
  - All the instructions in a group could be executed in parallel (if no structural hazards and if any dependences through memory were preserved)
  - Instruction group can be arbitrarily long
  - Compiler must explicitly indicate the boundary between one instruction group and another by placing a stop between two instructions that belong to different groups
- IA-64 instructions are encoded in bundles, which are 128 bits wide.
  - Each bundle consists of a 5-bit template field and 3 instructions, each 41 bits in length
- 3 Instructions in 128 bit “groups”; field determines if instructions dependent or independent

Five Types of Execution in Bundle

Execution Unit Slot	*Instruction type*	*Instruction Description*	*Example Instructions*
I-unit	A	Integer ALU	add, subtract, and, or, cmp
	I	Non-ALU Int	shifts, bit tests, moves
M-unit	A	Integer ALU	add, subtract, and, or, cmp
	M	Mem access	Loads, stores for int/FP regs
F-unit	F	Floating point	Floating point instructions
B-unit	B	Branches	Conditional branches, calls
L+X	L+X	Extended	Extended immediates, stops

Template Examples

Template	Slot 0	Slot 1	Slot 2
0	M	I	I
1	M	I	I
2	M	I	I
3	M	I	I
…	…	…	…
28	M	F	B
29	M	F	B

Stop bits

Predication Support

- Nearly all instructions are predicated
  - Conditional branches are predicated jumps!
- Compare/Test instructions set predicates
  - Ten different comparison tests + 2 predicate destinations
  - Written with result of comparison + complement

Speculation Support

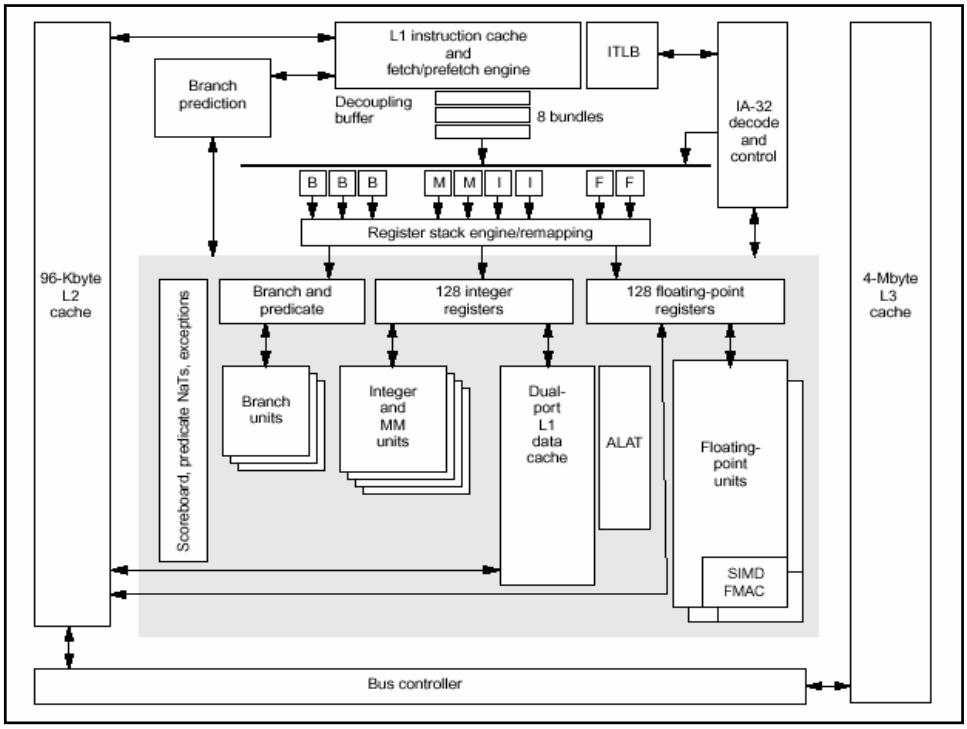
- All INT registers have a 1-bit NaT (Not A Thing)
  - This is a poison bit (as discussed earlier)
  - Speculative loads generate these
  - All other instructions propagate them
- Deferred exceptions
  - Nonspeculative exceptions receive a NAT as a source operand there is an unrecoverable exception
  - Chk.s instructions can detect and branch to recovery code

Memory Reference Support

- *Advanced Loads* allow speculative memory references
  - Move loads ahead of potentially dependent stores
  - ALAT table is allocated with register destination + memory address
  - Stores associatively lookup the table when they execute
    - Invalidate ALAT entries with same memory address
- Before using the value of the advanced load
  - Explicit check is needed to see if ALAT entry is valid
  - If it fails, can re-load the value or perform cleanup operation

Itanium™ Machine Characteristics

Frequency	800 MHZ
Transistor count	25.4M CPU; 295M L3
Process	0.18u CMOS, 6 metal layers
Package	Organic Land Grid Array
Machine width	6 instructions/clock (4 ALU/MM, 2 Ld/St, 2 FP, 3 Br)
Registers	14 ported 128 GR & 128 FR; 64 Predicates
Speculation	32 entry ALAT, Exception Deferral
Branch prediction	Multilevel 4-stage Prediction Hierarchy
FP compute bandwidth	3.2 GFlops (DP/EP); 6.4 GFlops (SP)
Memory →FP bandwidth	4 DP (8 SP) operands/clock
Virtual memory support	64 entry ITLB, 32/96 2-level DTLB, VHPT
L2/L1 cache	Dual ported 96K unified & 16KD; 16KI
L2/L1 latency	6 / 2 clocks
L3 cache	4MB, 4-way s.a., BW of 12.8 GB/sec;
System bus	2.1 GB/sec; 4-way glueless MP; scalable to large (512+ proc) systems



10 Stage In-Order Core Pipeline

Front End
- Pre-fetch/Fetch of up to 6 instructions/cycle
- Hierarchy of branch predictors
- Decoupling buffer

Execution
- 4 single cycle ALUs, 2 ld/str
- Advanced load control
- Predicate delivery & branch
- Nat/Exception//Retirement

IPG	FET	ROT	EXP	REN	WL.D	REG	EXE	DET	WRB
INST POINTER GENERATION	FETCH	ROTATE	EXPAND	RENAME	WORD-LINE DECODE	REGISTER READ	EXECUTE	EXCEPTION DETECT	WRITE-BACK

Instruction Delivery
- Dispersal of up to 6 instructions on 9 ports
- Reg. remapping
- Reg. stack engine

Operand Delivery
- Reg read + Bypasses
- Register scoreboard
- Predicated dependencies

Itanium processor 10-stage pipeline

- Front-end (stages IPG, Fetch, and Rotate): prefetches up to 32 bytes per clock (2 bundles) into a prefetch buffer, which can hold up to 8 bundles (24 instructions)
  - Branch prediction is done using a multilevel adaptive predictor like P6 microarchitecture
- Instruction delivery (stages EXP and REN): distributes up to 6 instructions to the 9 functional units
  - Implements registers renaming for both rotation and register stacking.

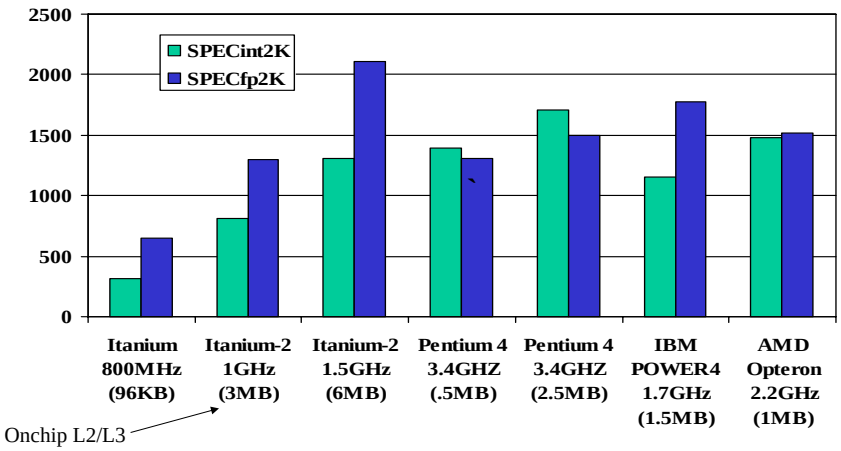
Itanium processor 10-stage pipeline

- Operand delivery (WLD and REG):
  - Accesses register file
  - Performs register bypassing
  - Accesses and updates a register scoreboard
    - Scoreboard used to detect when individual instructions can proceed, so that a stall of 1 instruction in a bundle need not cause the entire bundle to stall
  - Checks predicate dependences.

Itanium processor 10-stage pipeline

- Execution (EXE, DET, and WRB)
  - Executes instructions through ALUs and load/store units
  - Detects exceptions and posts NaTs
  - Retires instructions and performs write-back
  - Deferred exception handling via poison bits (NaTs)
- Predicate Delivery
  - Predicates generated in EXE delivered in DET and feed into retirement, branch execution, dependency detect
  - All instructions read operands and execute
  - Canceled at retirement

Peformance of IA-64 Itanium?

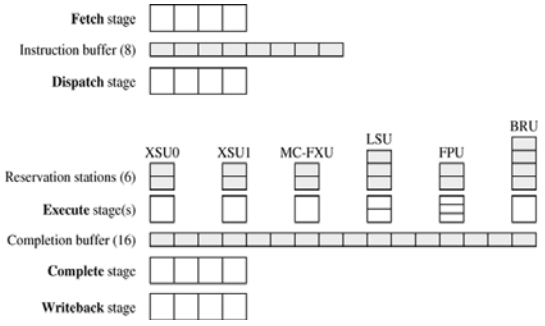


IBM PowerPC 620

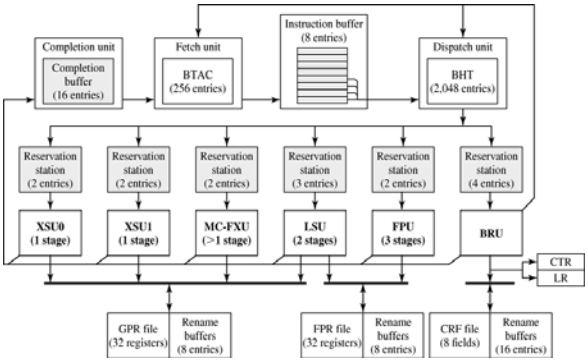
- Core design reused for POWER3, POWER4, POWER5: IBM servers
- Core used in game machines: PS2, XBOX

- 4-way superscalar
- OOO – BTB (BTAC), BHT.
- Distributed rsvn stations
- Register renaming with separate rename buffers

PowerPC 620 pipeline



PowerPC 620 architecture



PowerPC 620 -> POWER3 -> POWER4 -> POWER5

- More aggressive branch prediction algo
- More functional units (still 4-way, but more units and rsvn tables)
- More rename registers
- More processor cores