PowerPC 750

[PowerPC 750 User’s Manual]

General

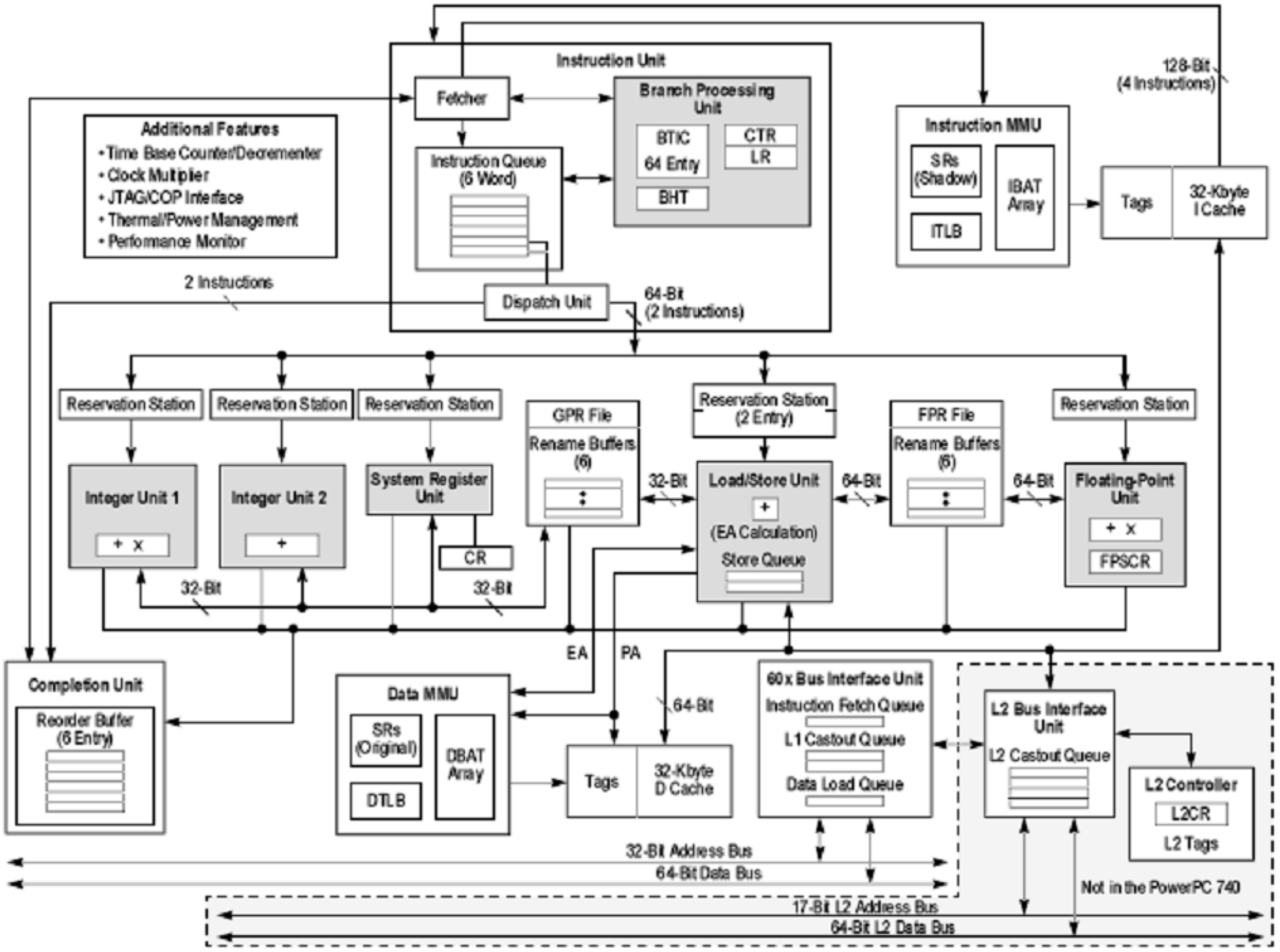
- PowerPC 750 is an implementation of PowerPC microprocessor family of **reduced instruction set computer (RISC)** microprocessors
- 750 implements the 32-bit portion of the PowerPC architecture
- Provides 32-bit effective addresses
  - Integer data types of 8, 16, and 32 bits
  - Floating-point data types of 32 and 64 bits
- High-performance, **superscalar** microprocessor
  - As many as **four instructions can be fetched** from the instruction cache per cycle
  - As many as **two instructions can be dispatched** per clock
  - As many as **six instructions can execute** per clock
  - Six independent execution units and two register files

PowerPC Instructions

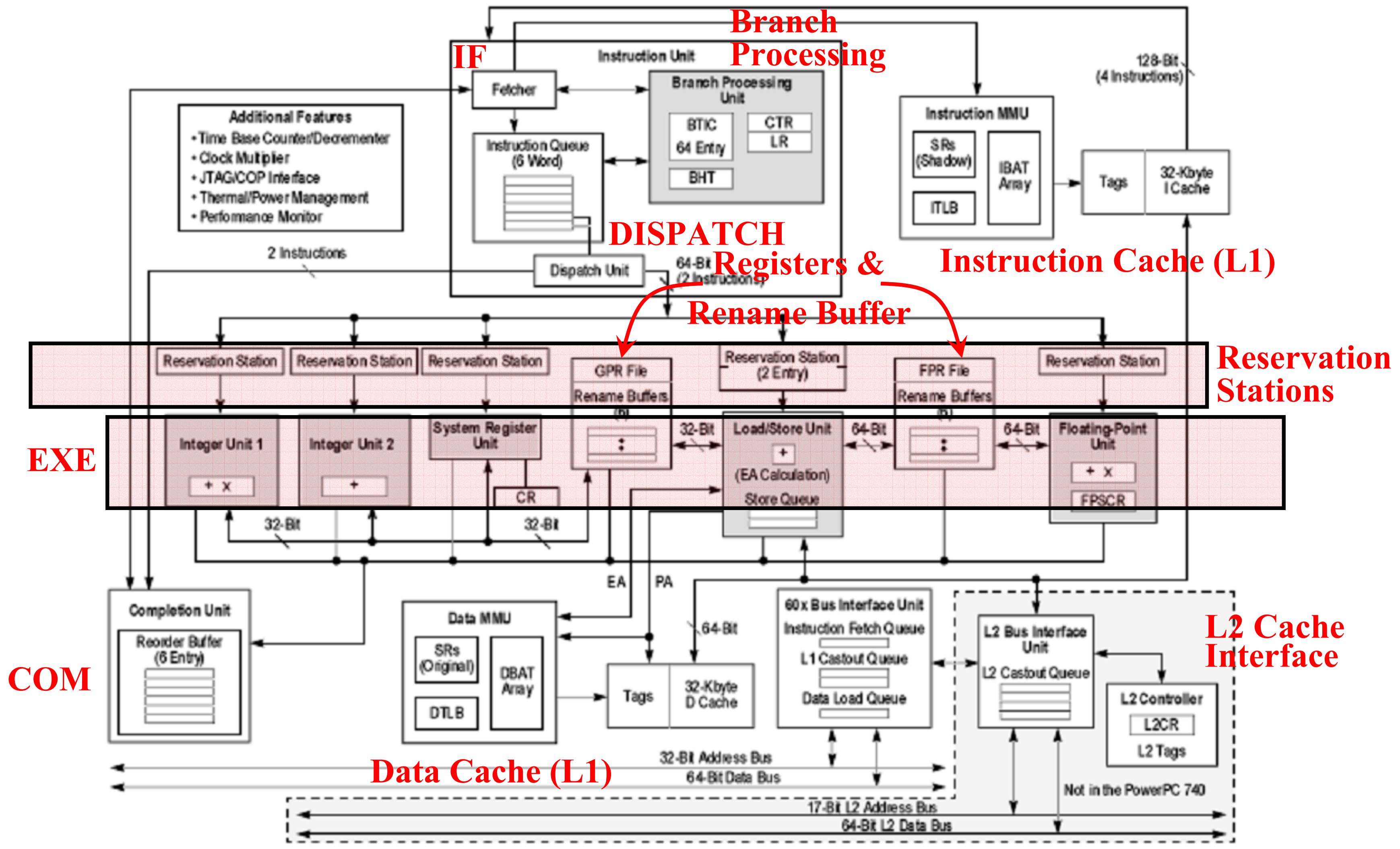
- All PowerPC instructions are encoded as single-word (32-bit)
- Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses.
- This fixed instruction length and consistent format greatly simplifies instruction pipelining

- Integer instructions
  - Integer arithmetic, compare, logical, rotate and shift instructions
- Floating-point instructions
  - Floating-point arithmetic, multiply/add, rounding and conversion, compare, status and control instructions
- Load/store instructions
  - Integer load and store instructions and Floating-point load and store
  - Primitives used to construct atomic memory operations (lwarx and stwcx. instructions)
- Flow control instructions
  - branching, condition register logical, trap, and other instructions that affect the instruction flow
- Processor control instructions:
  - These instructions are used for synchronizing memory accesses and management of caches, TLBs, and the segment registers.
- Memory control instructions
  - These instructions provide control of caches, TLBs, and SRs.

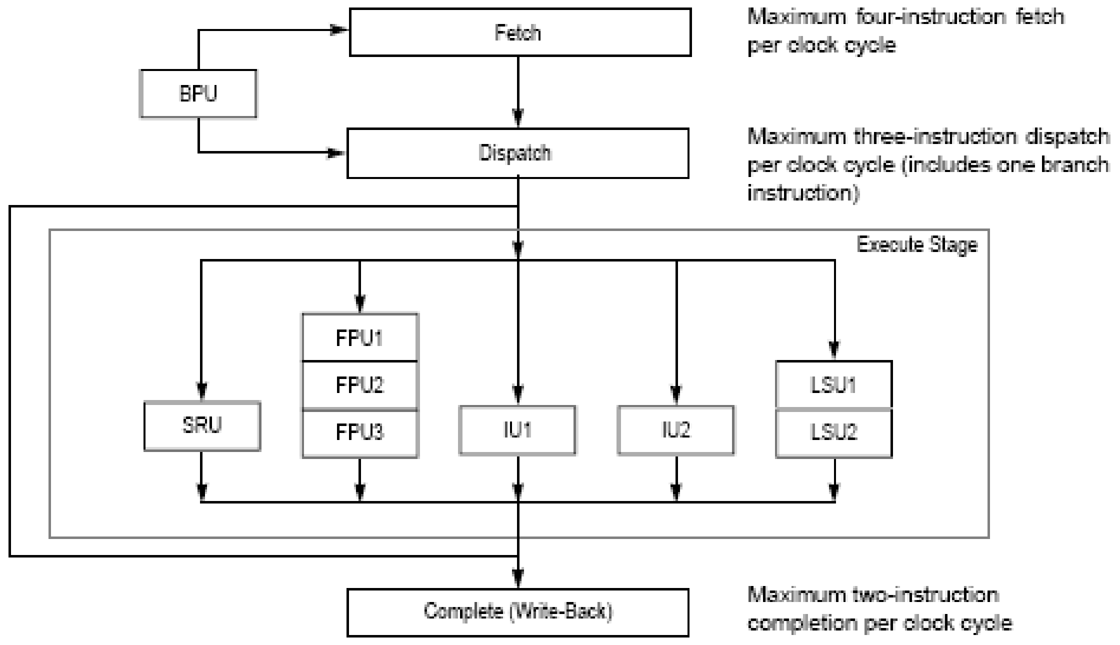
PowerPC 750 Microprocessor Block Diagram



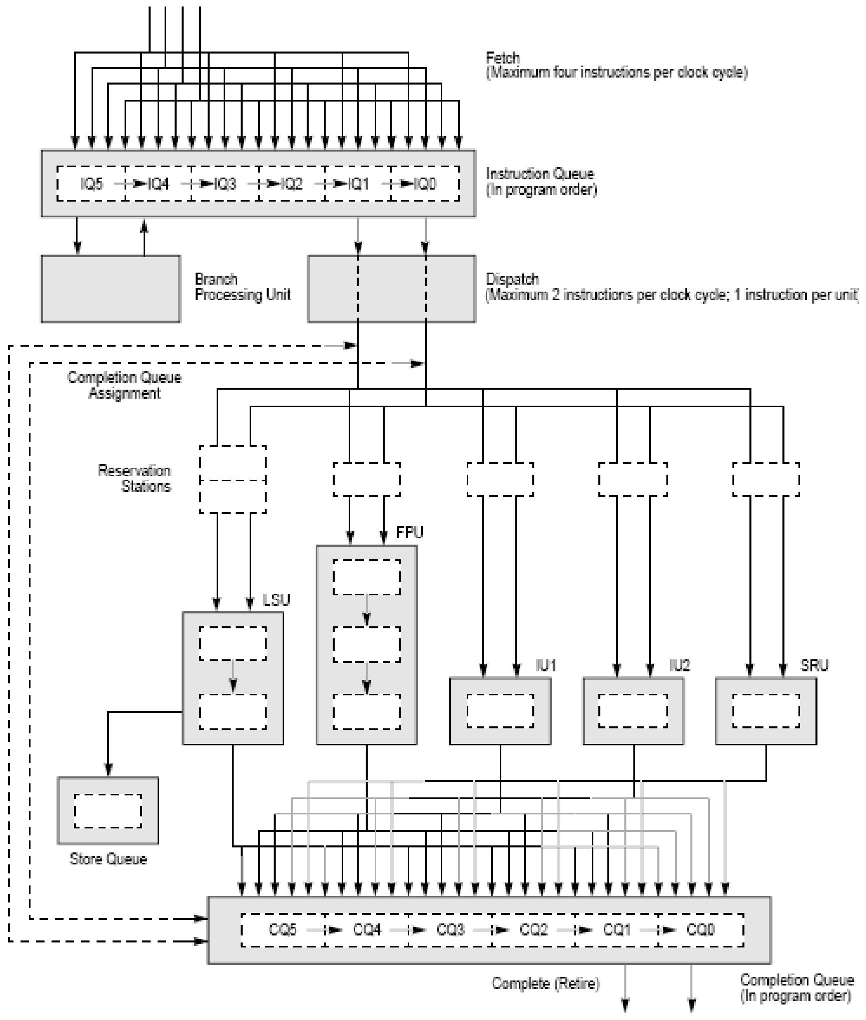
PowerPC 750 Microprocessor Block Diagram



Superscalar Pipeline


Fetch
Maximum four-instruction fetch per clock cycle
BPU
Dispatch
Maximum three-instruction dispatch per clock cycle (includes one branch instruction)
Execute Stage
FPU1
FPU2
FPU3
SRU
IU1
IU2
LSU1
LSU2
Complete (Write-Back)
Maximum two-instruction completion per clock cycle

Instruction Flow



Fetch

- Clock cycles necessary to request instructions from the memory system
- Where exactly:
  1. the branch target instruction cache
  2. the on-chip instruction L1 cache
  3. the L2 cache

Decode/Dispatch

- The time it takes to fully decode the instruction and dispatch it from the instruction queue to the appropriate execution unit
- Instruction dispatch requires the following:
  - Instructions can be dispatched only from the two lowest instruction queue entries, IQ0 and IQ1.
  - A maximum of two instructions can be dispatched per clock cycle (although an additional branch instruction can be handled by the BPU)
  - Only one instruction can be dispatched to each execution unit per clock cycle
  - There must be a vacancy in the specified execution unit.
  - A rename register must be available for each destination operand specified by the instruction
  - For an instruction to dispatch, the appropriate execution unit must be available
  - There must be an open position in the completion queue. If no entry is available, the instruction remains in the IQ.

Execution Units

- Two integer units (IUs) that share thirty-two GPRs for integer operands
  - IU1 can execute any integer instruction
  - IU2 can execute all integer instructions except multiply and divide
  - Single-entry reservation station for each
- One three-stage floating point unit (FPU)
  - both single- and double-precision operations
  - Hardware support for denormalized numbers
  - Single-entry reservation station
  - Thirty-two 64-bit FPRs for single- or double-precision operands

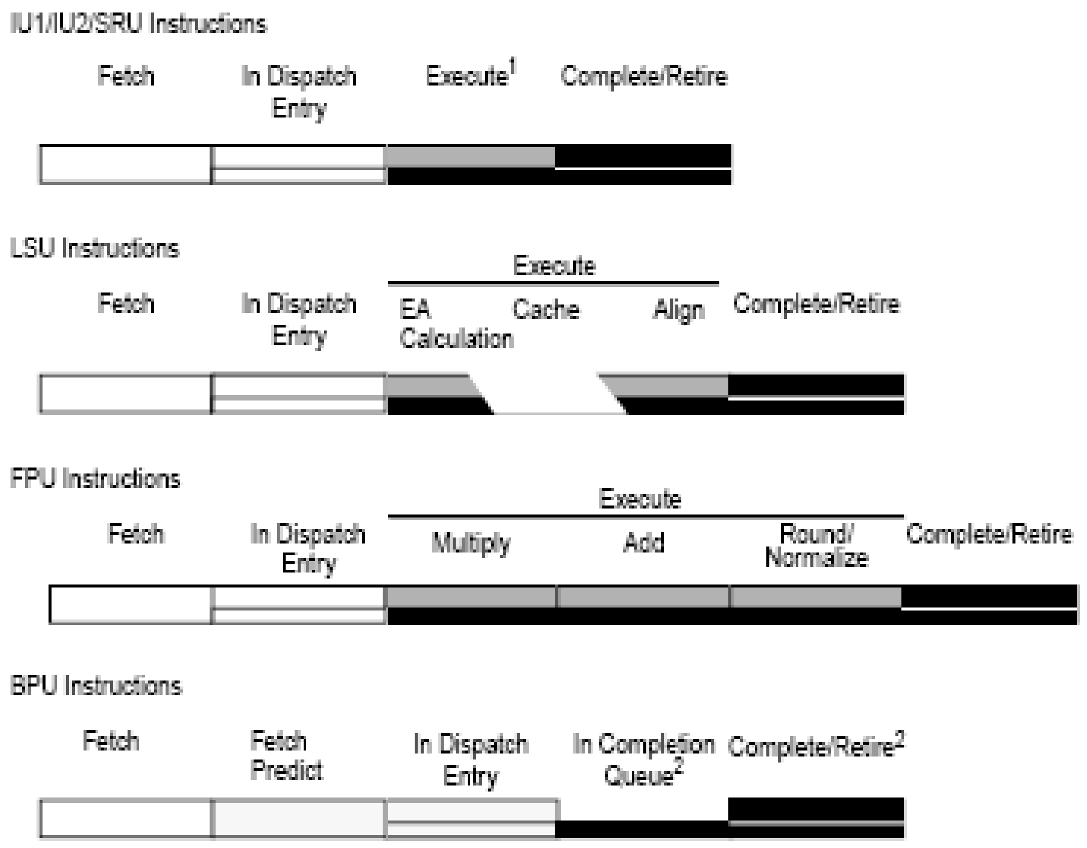
Execution Units (Cont'd)

- Two-stage LSU
  - Two-entry reservation station
  - Single-cycle, pipelined cache access
  - Dedicated adder performs EA calculations
  - Performs alignment and precision conversion for floating-point data
  - Performs alignment and sign extension for integer data
  - Three-entry store queue
  - Supports both big- and little-endian modes
- SRU handles miscellaneous instructions
  - Executes CR logical and Move to/from SPR instructions (mtspr and mfspr)
  - Single-entry reservation station

Completion

- Completion unit retires an instruction from the six-entry reorder buffer (completion queue) when
  1. All instructions ahead of it have been completed, and
  2. The instruction has finished execution, and
  3. No exceptions are pending
- Guarantees sequential programming model (precise exception model)
- Monitors all dispatched instructions and retires them in order
- Tracks unresolved branches and flushes instructions from the mispredicted branch
- Retires as many as two instructions per clock

Pipeline Stages



1 Several integer instructions, such as multiply and divide instructions, require multiple cycles in the execute stage.

2 Only those branch instructions that update the LR or CTR take an entry in the completion queue.

Rename Buffers

- 750 provides rename registers for holding instruction results before the completion commits them to the architected register
- There are six GPR rename registers, six FPR rename registers, and one each for the CR, LR, and CTR
- When an instruction is dispatched to its execution unit, a rename register for the results of that instruction is assigned
- Dispatcher also provides a tag to the execution unit identifying the rename register that forwards the required data for an instruction
- When the source data reaches the rename register, execution can begin
- Results are transferred from the rename registers to the architected registers by the completion unit when an instruction is retired from completion queue
- Results of squashed instructions are flushed from the rename registers

Branch Prediction Unit

- Featuring both static and dynamic branch predictions
  - Only one is used at any given time
- Static branch prediction
  - Static branch prediction is defined by the PowerPC architecture and involves encoding the branch instructions
  - The PowerPC architecture provides a field in branch instructions (the BO field) to **allow software to hint** whether a branch is likely to be taken
  - Rather than delaying instruction processing until the condition is known, the 750 uses the instruction encoding to predict whether the branch is likely to be taken and begins fetching and executing along that path
- Dynamic branch prediction: Branch history table (BHT)
  - 512-entry with two bits per entry
  - Not-taken, strongly not-taken, taken, strongly taken

Branch Target Cache

- Used to reduce time required for fetching target instructions when branch is predicted to be taken
- Branch Target Instruction Cache (BTIC)
  - 64-entry (16-set, four-way set-associative)
  - Cache of branch instructions that have been encountered in branch/loop code sequences
  - BTIC hit: instructions are fetched into the instruction queue a cycle sooner than it can be made available from the instruction cache
  - Typically, provides the first two instructions in the target stream

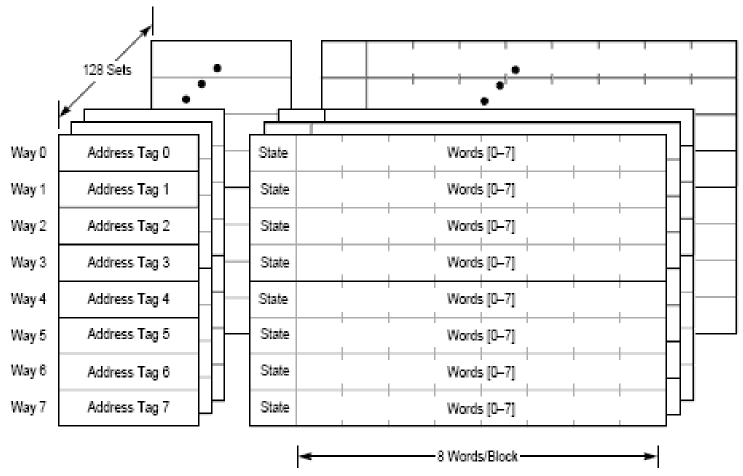
Multiple Branch Predictions

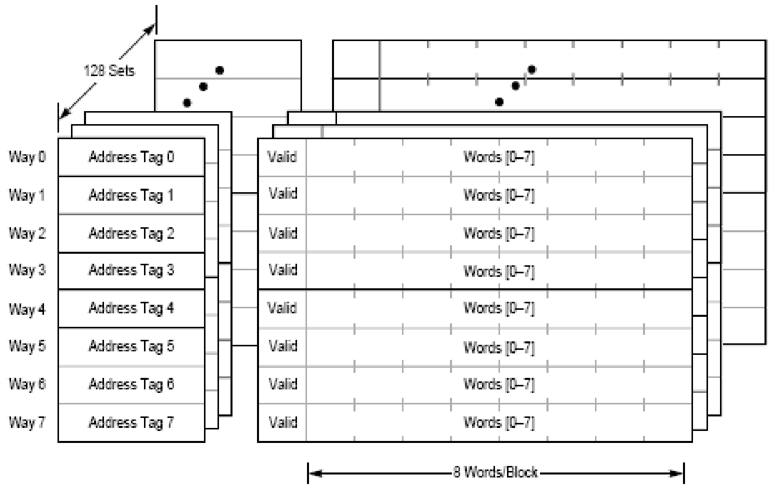
- The 750 executes through two levels of prediction
- Instructions from the first unresolved branch can execute, but they cannot complete until the branch is resolved.
- If a second branch instruction is encountered in the predicted instruction stream, it can be predicted
- Instructions can be fetched, but not executed, from the second branch
- No action can be taken for a third branch instruction until at least one of the two previous branch instructions is resolved

Cache

- Separate on-chip instruction and data caches
  - 32-Kbyte, eight-way set-associative instruction
  - Pseudo least-recently-used (PLRU) replacement
  - 32-byte (eight-word) cache block
  - Physically indexed/physical tags
  - Cache write-back or write-through operation per-block basis
  - Caches can be disabled in software
  - Caches can be locked in software
  - Data cache coherency (MEI) maintained in hardware
  - The critical double word is made available to the requesting unit
  - The cache is non-blocking

Data and Instruction Cache Organization





32-word blocks (5 bits)
128 sets (7 bits)
Tags (20 bits)
State: MEI

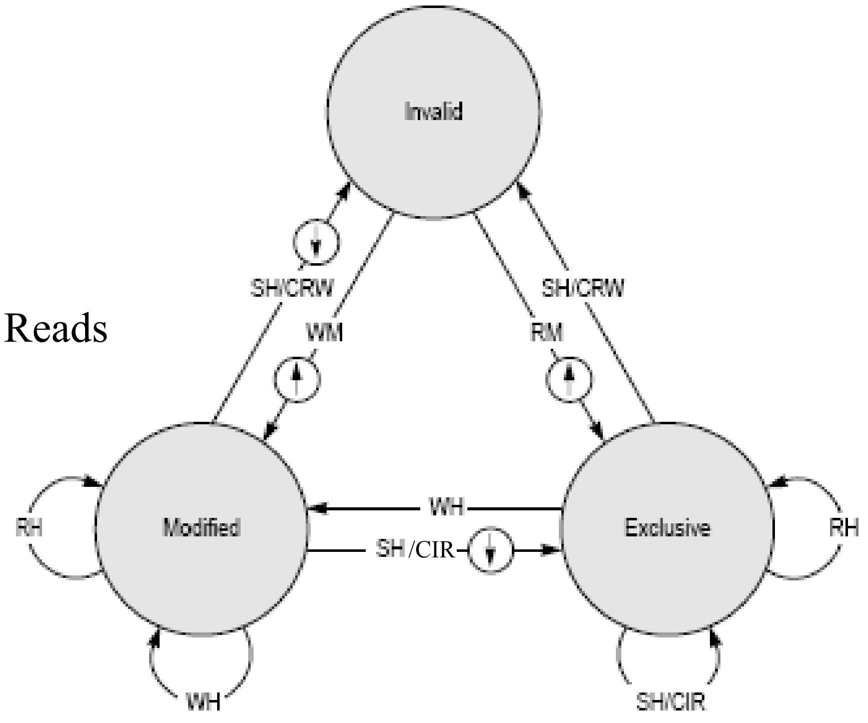
32-word blocks (5 bits)
128 sets (7 bits)
Tags (20 bits)
Valid/Not valid
Not snooped

Multiprocessing

- Multiprocessing support features include the following:
  - Hardware-enforced, three-state cache coherency protocol (MEI) for data cache.
  - Load/store with reservation instruction pair for atomic memory references, semaphores, and other multiprocessor operations
- The 750's three-state cache-coherency protocol (MEI) supports the Modified, Exclusive, and Invalid states
- A compatible subset of the MESI (modified/exclusive/shared/invalid) four-state protocol, and it operates coherently in systems with four-state caches

MEI Protocol

- Three states:
  - Invalid
  - Modified
  - Exclusive
- No shared state
- Caching-Inhibited Reads



Bus Transactions

SH = Snoop Hit
RH = Read Hit
RM = Read Miss
WH = Write Hit
WM = Write Miss
SH/CRW = Snoop Hit, Cacheable Read/Write
SH/CIR = Snoop Hit, Caching-Inhibited Read

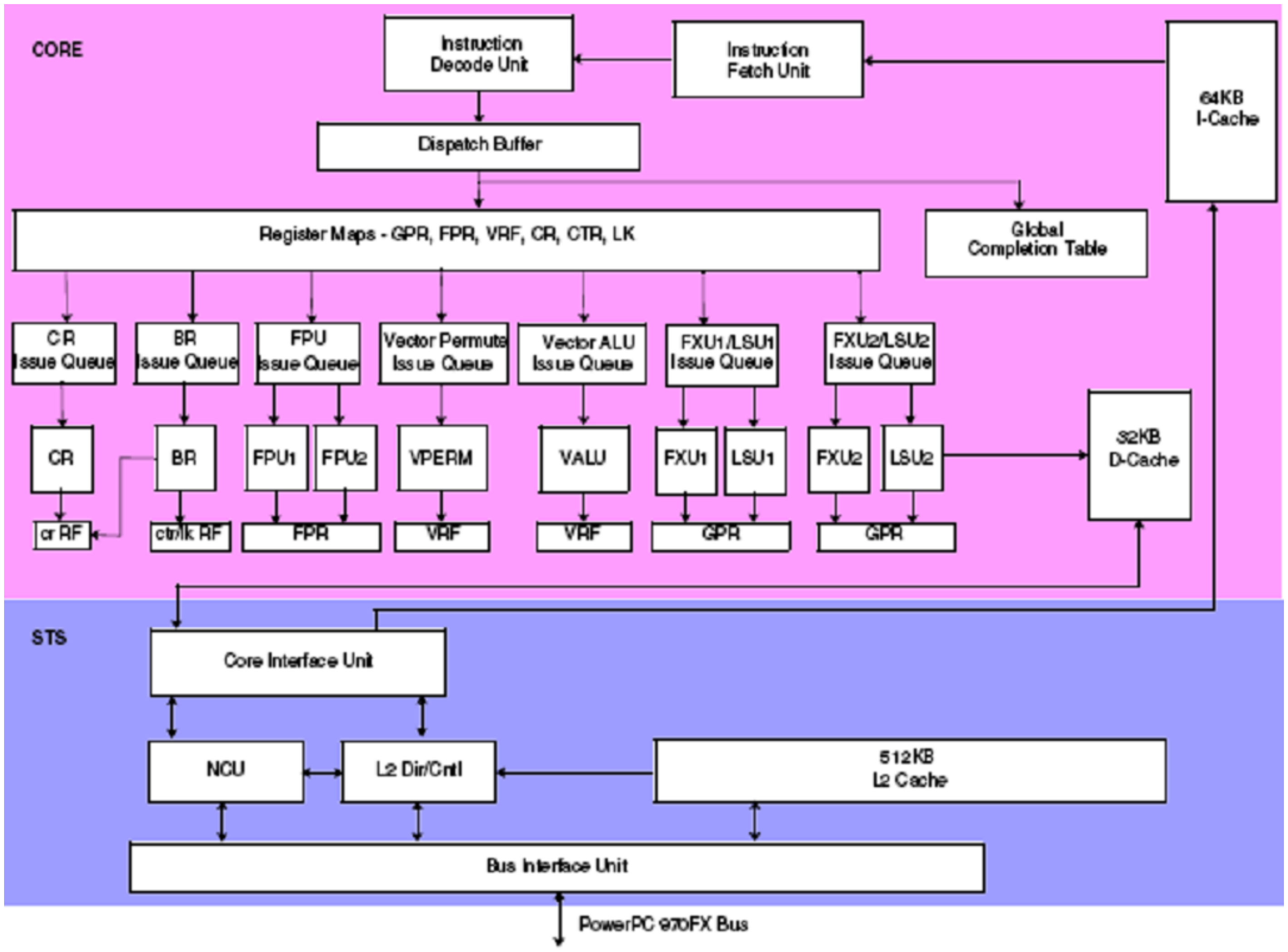
↓ = Snoop Push

↑ = Cache Block Fill

PowerPC 970FX

[PowerPC 970FX User’s Manual]

PowerPC 970FX Block Diagram



PowerPC 970FX

- 64-bit implementation of the PowerPC® AS Architecture (version 2.01)
- Vector/SIMD Multimedia eXtension
- Deeply pipelined design
  - 16 stages for most fixed-point register-register operations
  - 18 stages for most load and store operations (assuming an L1 D-cache hit)
  - Up to 25 stages for floating point operations
  - the VALU.
  - 19 stages for vector permute operations
- Dynamic instruction cracking
  - Some complex instructions are broken into two simpler, more RISC-like instructions**!**
  - Allows for simpler inner core dataflow

General

- Aggressive branch prediction
  - Prediction for up to two branches per cycle
  - Support for up to 16 predicted branches in flight
  - Prediction support for branch direction and branch addresses
- In-order dispatch of up to five operations into distributed issue queue structure
- Out-of-order issue of up to 10 operations into 10 execution pipelines
  - Two load or store operations
  - Two fixed-point register-register operations
  - Two floating-point operations
  - One branch operation
  - One condition register operation
  - One vector permute operation
  - One vector ALU operation
- Register renaming
- Cache coherency protocol: **MERSI** (modified/exclusive/recent/shared/invalid)

General (Cont'd)

- Large number of instructions in flight (theoretical maximum of 215 instructions)
- Up to 16 instructions in the instruction fetch unit (fetch buffer and overflow buffer)
- Up to 32 instructions in the instruction fetch buffer in instruction decode unit
- Up to 35 instructions in three decode pipe stages and four dispatch buffers
- Up to 100 instructions in the inner-core (after dispatch)
- Up to 32 stores queued in the store queue (STQ) (available for forwarding)
- Fast, selective flush of incorrect speculative instructions and results
- Specific focus on storage latency management
- Out-of-order and speculative issue of load operations
- Support for up to eight outstanding L1 cache line misses
- Hardware initiated instruction prefetching from L2 cache
- Software initiated data stream prefetching with support for up to eight active streams
- Critical word forwarding / critical sector first
- New branch processing / prediction hints for branch instructions

Instruction Fetch

- 64KB, direct-mapped instruction cache (I-cache)
  - 128-byte lines (broken into four 32-byte sectors)
  - Dedicated 32-byte read/write interface from L2 cache – critical sector first reload policy
- Four-entry, 128-byte, instruction prefetch queue above the I-cache; hardware-initiated prefetches
- Fetch 32-byte aligned block of eight instructions per cycle

Branch Prediction

- Scan all eight fetched instructions for branches each cycle
- Predict up to two branches per cycle
- Three-table prediction structure
  - Local (16K entries, 1-bit each); Taken/Not taken
  - Global (16K entries, 1-bit each) 11-bit history XORed with branch instruction address; Taken/ Not taken
  - Selector (16K entries, 1-bit each) indexed as above; Use local predictor/Use global predictor
- This combination of branch prediction tables has been shown to produce very accurate predictions on a wide range of workload types.

- 16-entry link stack for address prediction (with stack recovery)
  - predict the target address for a branch to link instruction that it believes corresponds to a subroutine return (pushed into the stack earlier)

- 32-entry count cache for address prediction (indexed by the address of Branch Conditional to Count Register (bcctr) instructions)

Instruction Decode and Preprocessing

- Three cycle pipeline to decode and preprocess instructions
  - Cracking one instruction into two internal operations
- Cracked and microcoded instructions have access to four renamed emulation GPRs (eGPRs), one renamed emulation FPR (eFPR), and one renamed emulation CR (eCR) field (in addition to architected facilities)
- 8-entry (16 bytes per entry) instruction fetch buffer (up to eight instructions in, five instructions out each cycle)

Instruction Dispatch and Completion Control

- Four dispatch buffers which can hold up to four dispatch groups when the global completion table (GCT) is full
- 20-entry global completion table
  - Group-oriented tracking associates a five operation dispatch group with a single GCT entry
  - Tracks internal operations from dispatch to completion for up to 100 operations
  - Capable of restoring the machine state for any of the instructions in flight
  - Very fast restoration for instructions on group boundaries (i.e., branches)
  - Slower for instructions contained within a group
- Supports precise exceptions

Branch and Condition Register Execution Pipelines

- One branch execution pipeline
  - Computes actual branch address and branch direction for comparison with prediction
  - Redirects instruction fetching if either prediction was incorrect
  - Assists in training/maintaining the branch table predictors, the link stack, and the count cache
- One condition register logical pipeline
  - Executes CR logical instructions and the CR movement operations
  - Executes some Move to/from Special Purpose Register (**mtspr and mfspr**) instructions also
- Out-of-order issue with bias towards oldest operations first

Data Stream Prefetch

- Eight (modeable) data prefetch streams supported in hardware. Eight hardware streams are only available if vector prefetch instructions are disabled
- Four vector prefetch streams supported using four of the eight hardware streams. The vector prefetch mapping algorithm supports the most commonly used forms of vector prefetch instructions

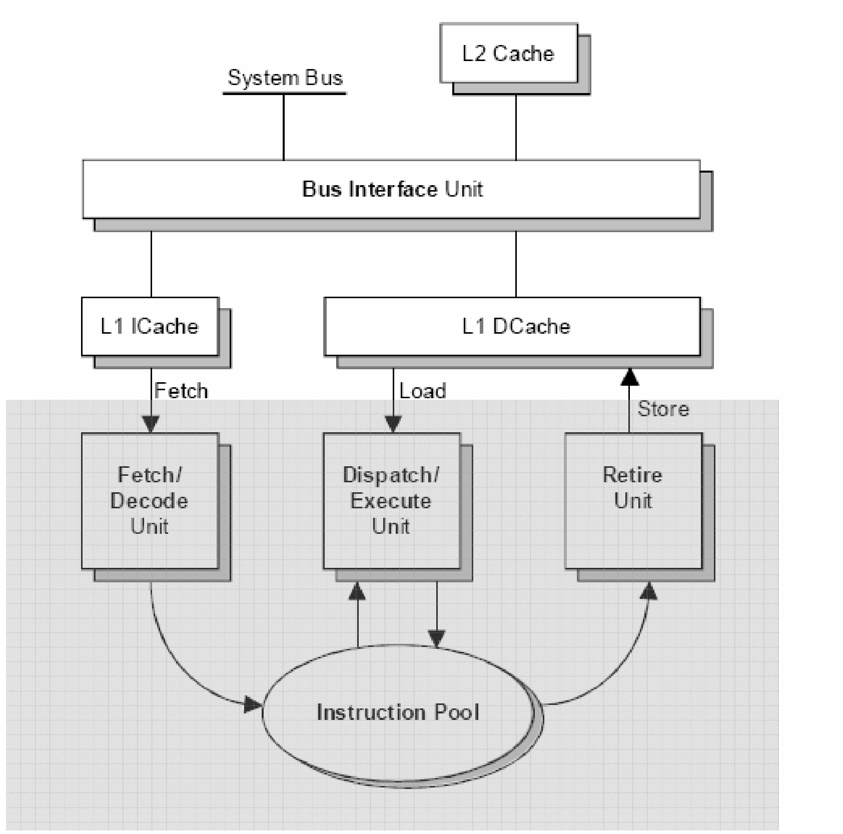
Intel P6

[P6 Family of Processors Hardware Developer's Manuals]

General

- The P6 family of processors is the generation of processors that succeeds the Pentium® line of Intel processors
- This processor family implements Intel’s dynamic execution microarchitecture
  - Multiple branch prediction
  - Data flow analysis
  - Speculative execution

P6 Three Engines and Interface with Memory

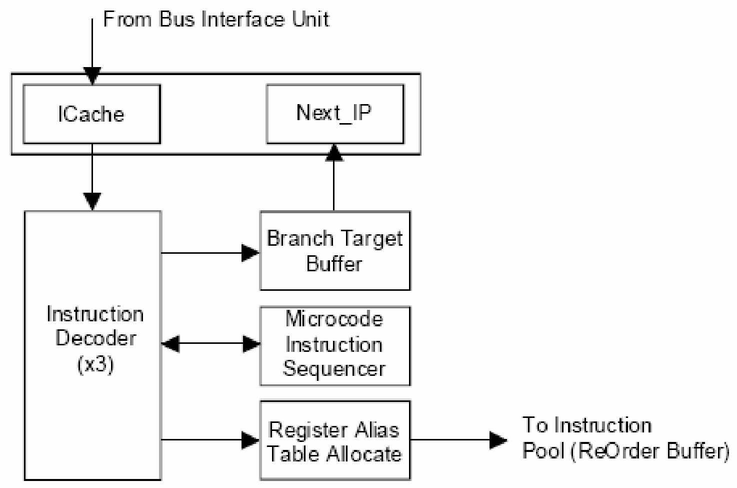


Major Units

- The FETCH/DECODE unit:
  - An in-order unit that takes as input the user program instruction stream from the instruction cache, and decodes them into a series of **μoperations** (μops) that represent the dataflow of that instruction stream. The pre-fetch is speculative
- The DISPATCH/EXECUTE unit:
  - An out-of-order unit that accepts the dataflow stream, schedules execution of the μops subject to data dependencies and resource availability and temporarily stores the results of these speculative executions
- The RETIRE unit:
  - An in-order unit that knows how and when to commit (“retire”) the temporary, speculative results to permanent architectural state
- The BUS INTERFACE unit:
  - The bus interface unit communicates directly with the L2 (second level) cache supporting up to four concurrent cache accesses. The bus interface unit also controls a transaction bus, with **MESI** snooping protocol, to system memory

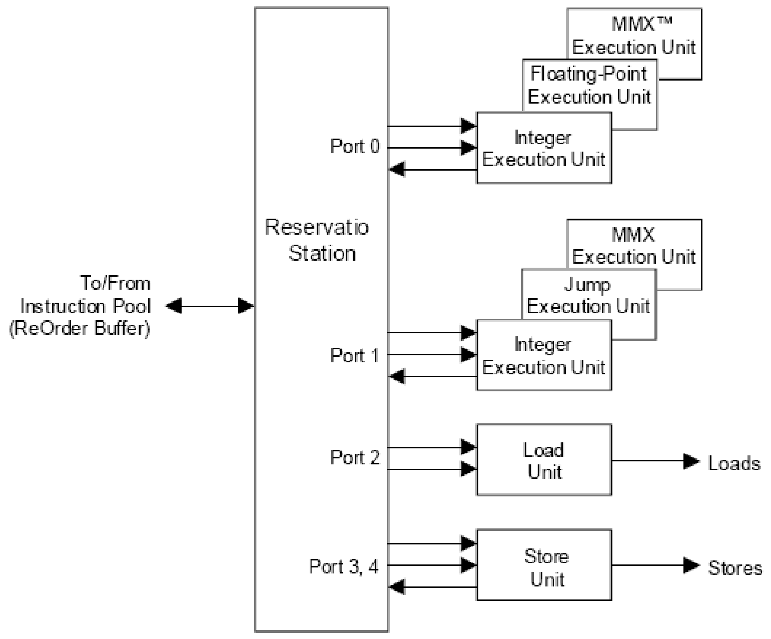
Inside Fetch

- The L1 Instruction Cache fetches the cache line corresponding to the index from the Next_IP and presents 16 aligned bytes to the decoder.
- The decoder converts the Intel Architecture instructions into triadic µops (two logical sources, one logical destination per µop)
- Most Intel Architecture instructions are converted directly into single µops, some instructions are decoded into one-to-four µops and the complex instructions require microcode
- The µops are queued, and sent to the Register Alias Table (RAT) unit, where the logical Intel Architecture-based register references are converte into references to physical registers in P6 family processors physical register references
- µopa are entered into the instruction pool
- The instruction pool is implemented as an array of Content Addressable Memory called the ReOrder Buffer (ROB).


From Bus Interface Unit
ICache
Next_IP
Instruction Decoder (x3)
Branch Target Buffer
Microcode Instruction Sequencer
Register Alias Table Allocate
To Instruction Pool (ReOrder Buffer)

Inside Dispatch/Execute

- The Dispatch unit selects μops from the instruction pool depending upon their status
- If the status indicates that a μop has all of its operands then the dispatch unit checks to see if the execution resource needed by that μop is also available
- If both are true, the Reservation Station removes that μop and sends it to the resource where it is executed
- The results of the μop are later returned to the pool
- There are five ports on the Reservation Station, and the multiple resources are accessed as shown



Inside Dispatch/Execute (Cont'd)

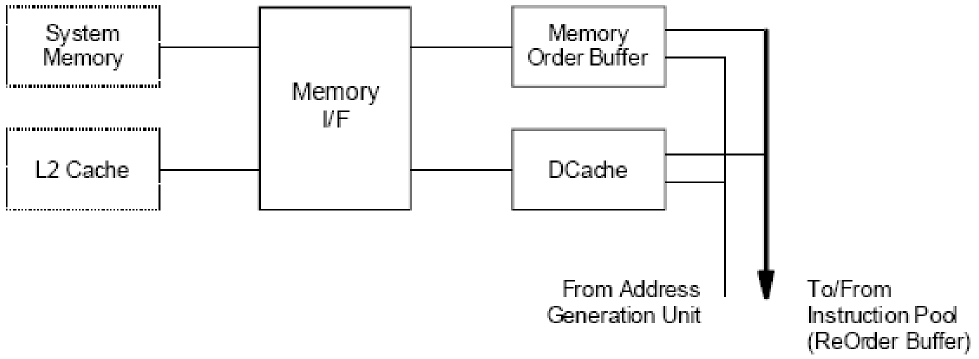
- The P6 family of processors can schedule (in an out-of-order fashion) at a peak rate of 5 μops per clock, one to each resource port, but a sustained rate of 3 μops per clock is more typical

- Note that many of the μops are branches
- The Branch Target Buffer (BTB) will correctly predict most of these branches
- Branch μops are tagged (in the in-order pipeline) with their fall-through address and the destination that was predicted for them
- But if mispredicted, then the Jump Execution Unit (JEU) changes the status of all of the μops behind the branch to remove them from the instruction pool
- In that case the proper branch destination is provided to the BTB which restarts the whole pipeline from the new target address

Inside Retire

- The Retire Unit is also checking the status of μops in the instruction pool
- Once removed, the original architectural target of the μops is written as per the original Intel Architecture instruction.
- The Retire Unit must also re-impose the original program order on them
- The Retire Unit must first read the instruction pool to find the potential candidates for retirement and determine which of these candidates are next in the original program order
- Then it writes the results of this cycle's retirements to the Retirement Register File (RRF). The Retire Unit is capable of retiring 3 μops per clock.

Bus Interface Unit


System Memory
L2 Cache
Memory I/F
Memory Order Buffer
DCache
From Address Generation Unit
To/From Instruction Pool (ReOrder Buffer)

- Loads are encoded into a single µop.
- Stores therefore require two µops, one to generate the address and one to generate the data. These µops must later re-combine for the store to complete.
- Stores are never performed speculatively since there is no transparent way to undo them
- Stores are also never re-ordered among themselves
- A store is dispatched only when both the address and the data are available and there are no older stores awaiting dispatch.

Bus Interface Unit (Cont’d)

- A study of the importance of memory access reordering concluded:
  - Stores must be constrained from passing other stores, for only a small impact on performance.
  - Stores can be constrained from passing loads, for an inconsequential performance loss.
  - Constraining loads from passing other loads or stores has a significant impact on performance.
- The Memory Order Buffer (MOB) allows loads to pass other loads and stores by acting like a reservation station and re-order buffer
- It holds suspended loads and stores and re-dispatches them when a blocking condition (dependency or resource) disappears

Intel Itanium

[Intel Itanium Processor Hardware Developer's Manuals and

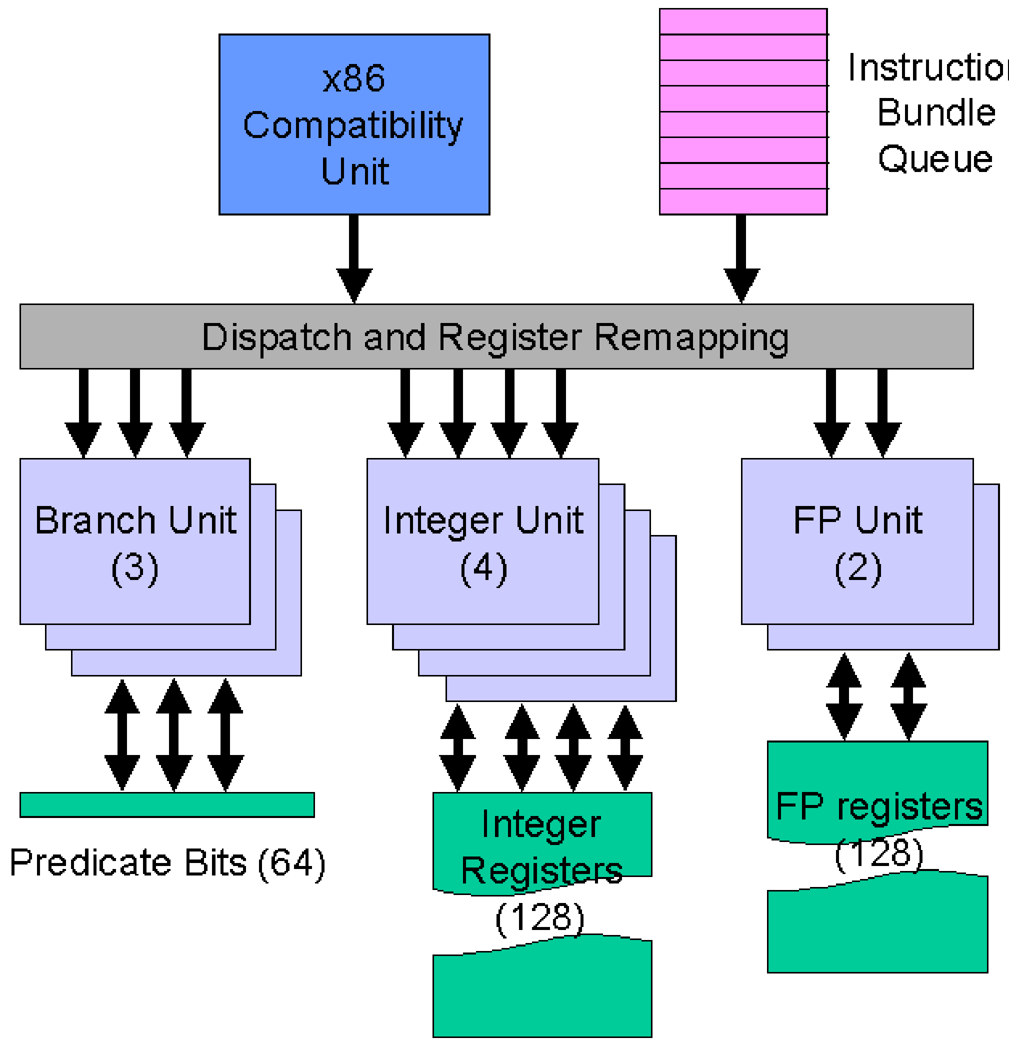
64-Bit CPUs: What You Need to Know,

extremetech.com]

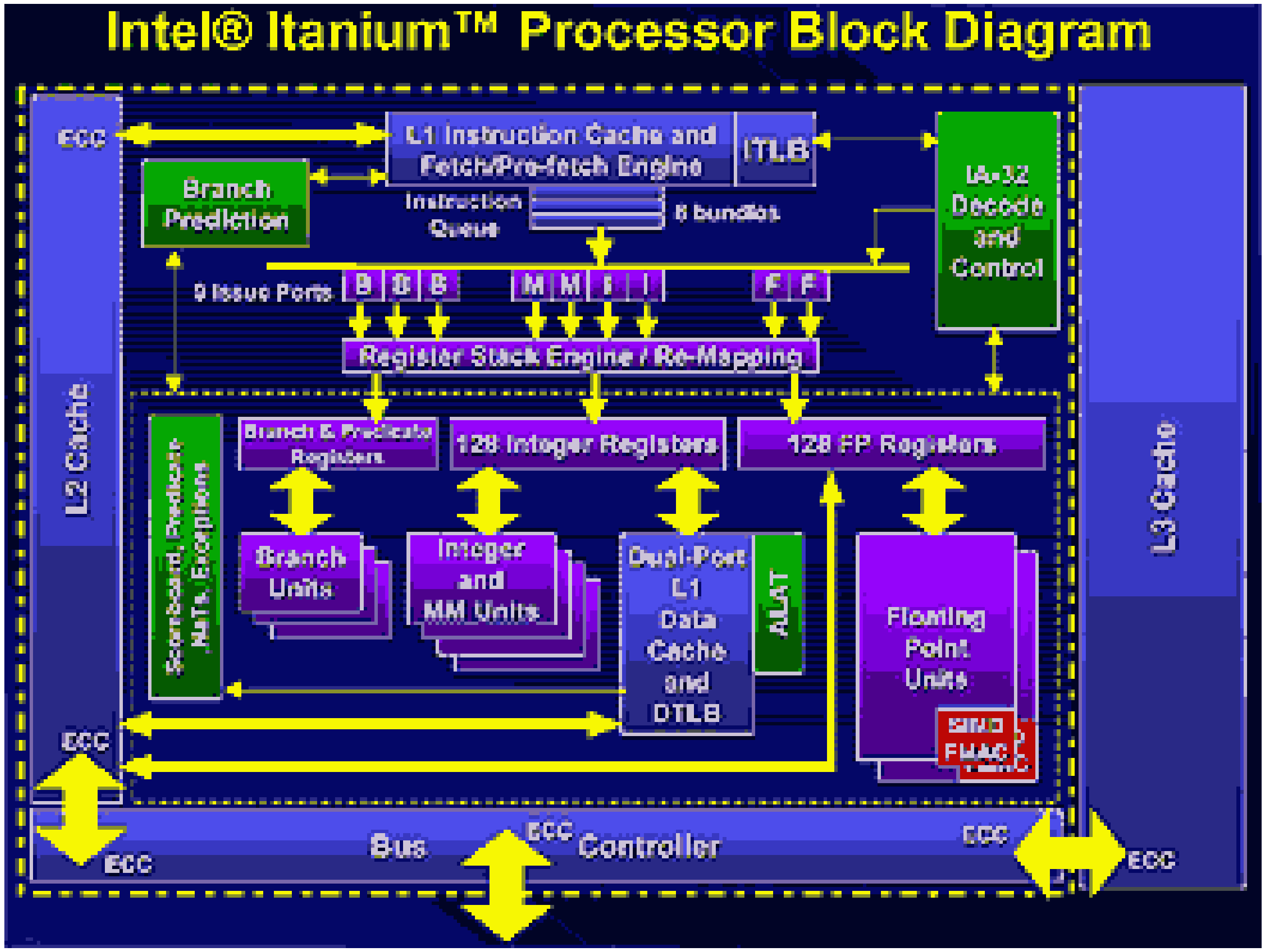
General

- First Implementation of Itanium Instruction Set
- EPIC: Explicitly Parallel Instruction Computing
- 6-wide 10-stage deep pipeline
- 4 inter units
- 4 multimedia units
- 2 load/store units
- 3 branch units
- 4 floating point units
- Can fetch, issue, execute, and retire 6 instructions per cycle

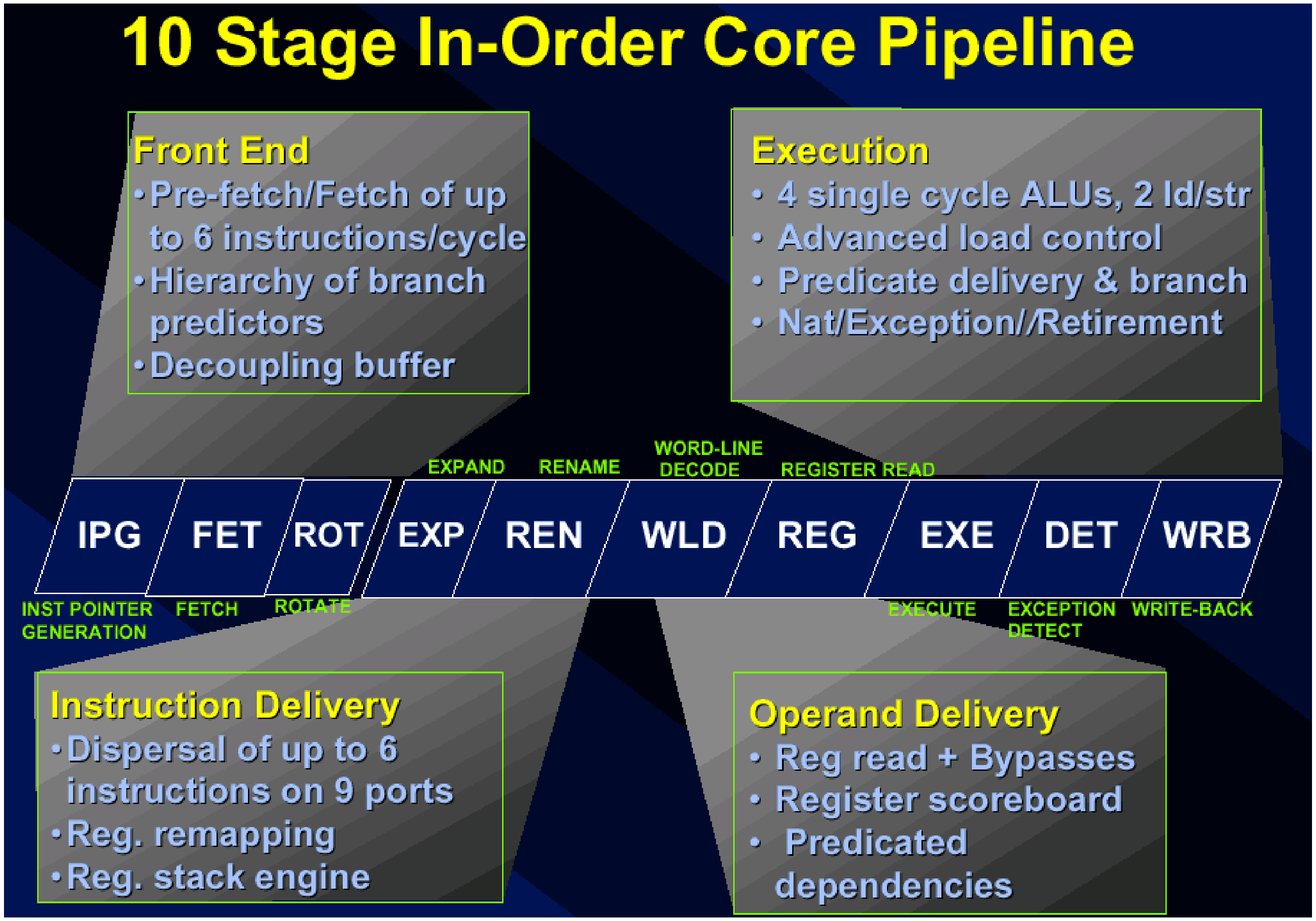
Itanium basic Block Diagram



Itanium Block Diagram



Itanium Pipeline



Pipeline Stages

- Front end (3 stages):
  - Pre-fetch and Fetch
  - Hierarchy of branch predictors
  - Decoupling Buffer
- Backend (7 stages):
  - Instruction delivery (2 stages)
    - 6 instructions into 9
    - Register remapping
    - Register Save Engine
  - Operand Delivery (2 stages)
    - Register file read and bypass
    - Register Scoreboard
    - Predicated dependencies
  - Execute (3 stages)
    - Execute and retire

Registers

- The Itanium processor has a massive register set
  - 128 general-purpose integer registers (each 64 bits wide)
  - 128 floating-point registers (each 82 bits wide)
  - 64 1-bit predicate registers
  - 8 branch registers, and a whole bunch of other registers
- Pushing and popping 128 registers for subroutine calls is very time-consuming (and usually not necessary anyway)
- IA-64 gets around the constant pushing and popping by using register frames
- The first 32 of the 128 integer registers are global, available to all tasks at all times. The other 96, though, can be framed, rotated, or both

Instructions, Bundles, and Groups

- Instructions are 41 bits long!
  - 7 bits to specify one of 128 general-purpose (or floating-point) registers; two source-operand fields and a destination field eat up 21 bits
  - Another 6 bits specify the 64 combinations of predication
- Instructions are delivered to the processor in "bundles”
  - Bundles are 128 bits: three 41-bit instructions (making 123 bits), plus one 5-bit template
- Instruction groups: collections of instructions that can theoretically all execute all at once.
  - It's the responsibility of the compiler to get this right
  - Groups can be of any arbitrary length, from one instruction up to millions of instructions that can all run at once without interfering with each other. A bit in the template identifies the end of a group
  - Processors will do their best to dispatch all the instructions in a group at once
- A bundle is not a group

- All IA-64 instructions fall into one of four categories
  - integer, load/store, floating-point, and branch operations

Predicated Instructions

- Almost all IA-64 instructions can be conditional
  - Predicated on literally anything
  - Beyond simple Z (zero), V (overflow), S (sign), and N (negative) flags
  - Has 64 free-form predicate bits, each considered a separate predicate register
  - You can set or clear a predicate bit any way you like, and its condition sticks indefinitely
  - Any subsequent instruction anywhere in the program can check that bit (or multiple bits) and behave accordingly