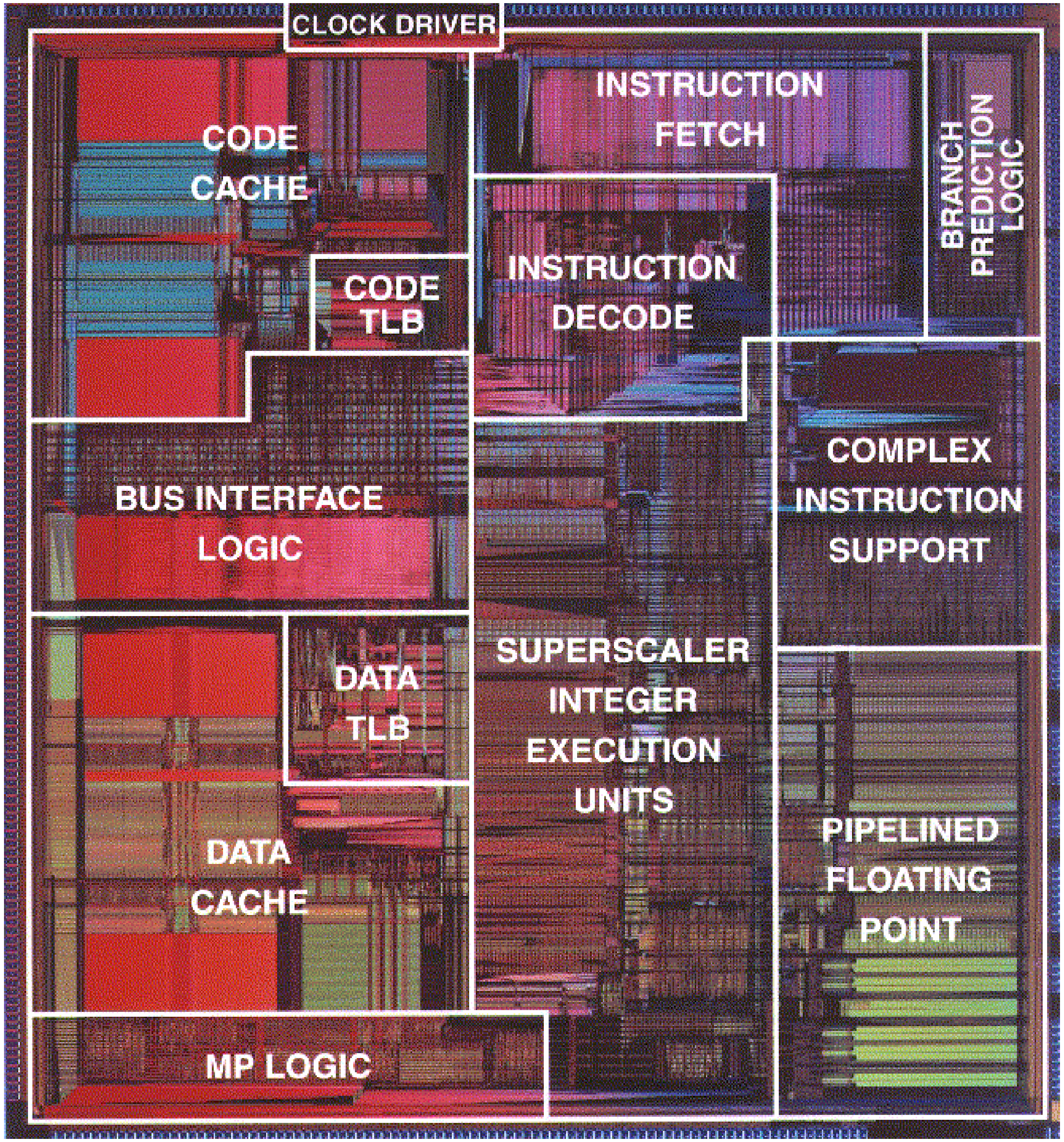
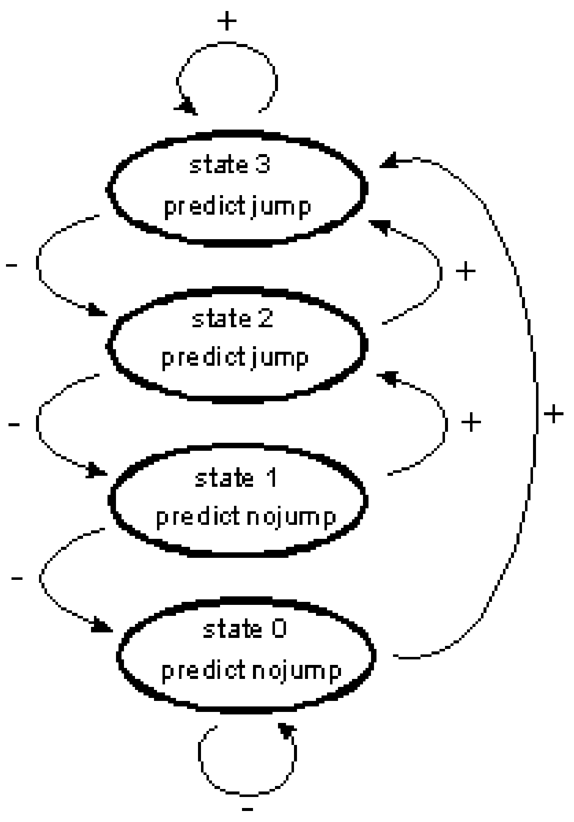
ELEC3020: Lecture 4-3

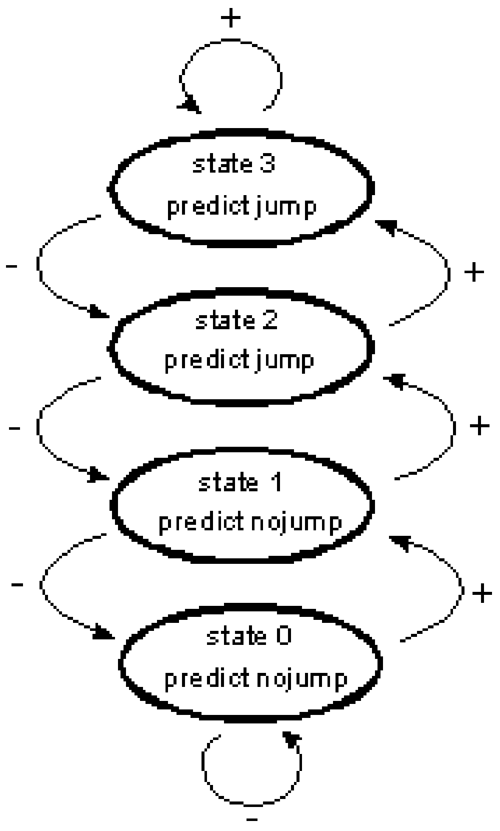
Branch prediction in the Pentium family

Pentium Die Photo



The Pentium





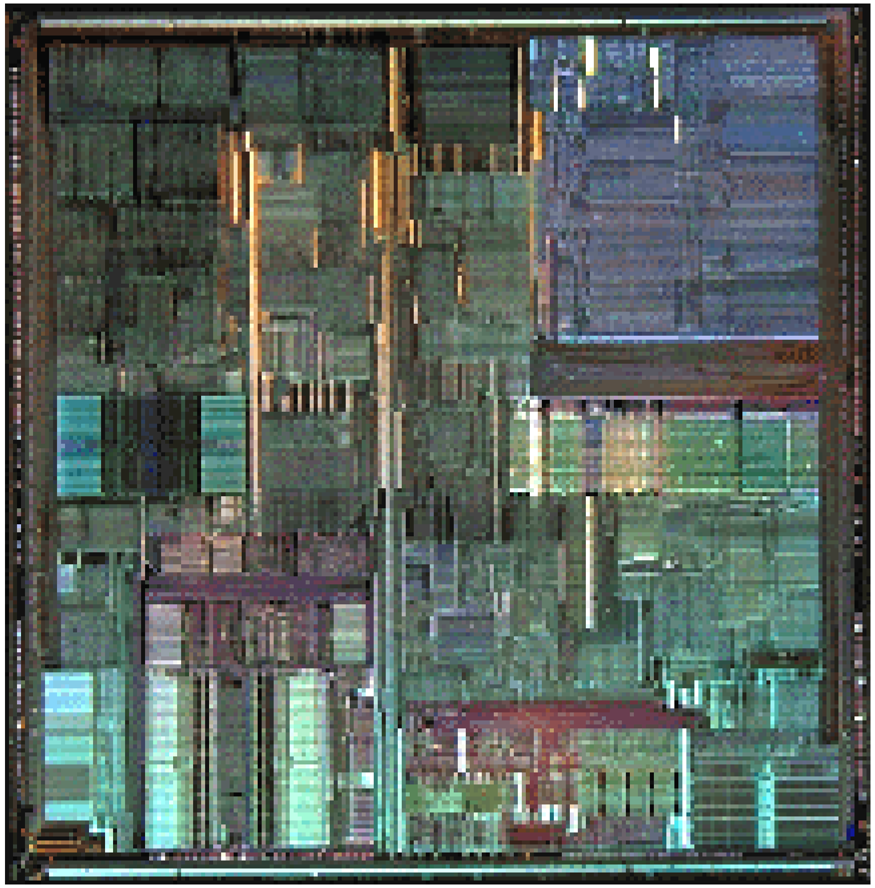
- **asymmetric design in the Pentium:**
- The state follows the +arrows when the branch instruction jumps, and the -arrows when not jumping. The branch instruction is predicted to jump next time if in state 2 or 3, and to not jump when in state 0 or 1.

- **symmetric design:**
- This is how the branch prediction *should* work. The state is incremented when jumping (+arrows) and decremented when not jumping (-arrows).

Pentium

- Yes, it appears to be slightly broken.
- The complication is that the designers have equated state 0 with 'vacant BTB entry'. This makes sense because state 0 is predicted to not jump anyway. But since it cannot distinguish between state 0 and a vacant BTB entry it will go to state 3 next time the branch jumps rather than to state 1.
- The convincing proof came is that tight loops behave differently. In a small loop the microprocessor doesn't have enough time to update the BTB entry for a branch instruction before it meets the same branch instruction again. In order to avoid a delay, it bypasses the BTB and reads the branch prediction state directly from the pipeline. And in this case it is actually able to go from state 0 to state 1, correctly.

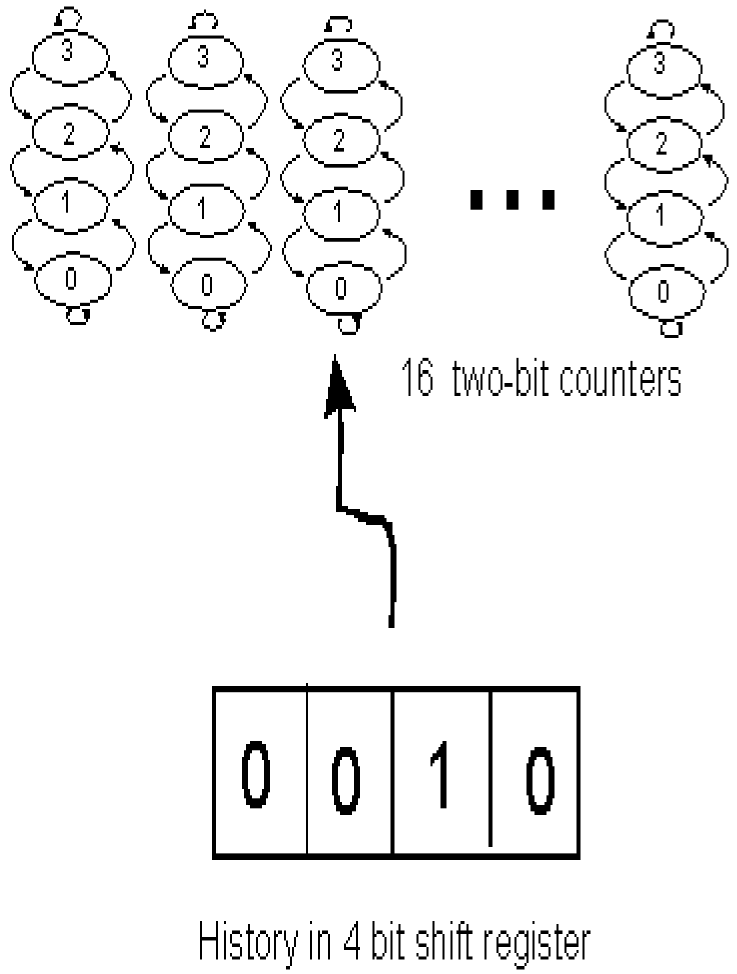
Pentium Pro Die Photo



Pentium Pro

- The later processors in the Pentium family: the Pentium MMX, Pentium Pro, Pentium II, Celeron, and Xeon, all have a much more advanced branch prediction mechanism.
- The improvement in the later processors comes from two-level branch prediction. The new first level is a shift register that stores the history of the last four events for any branch instruction. This gives sixteen possible bit patterns. You get a pattern of 0000 if the branch did not jump the last four times, and a pattern of 1111 after four times of jumping. The second level in the branch prediction mechanism is constituted of sixteen 2-bit counters as in the Pentium. It uses the 4-bit pattern in the first level to choose which of the sixteen counters to use in the second level.

Pentium Pro



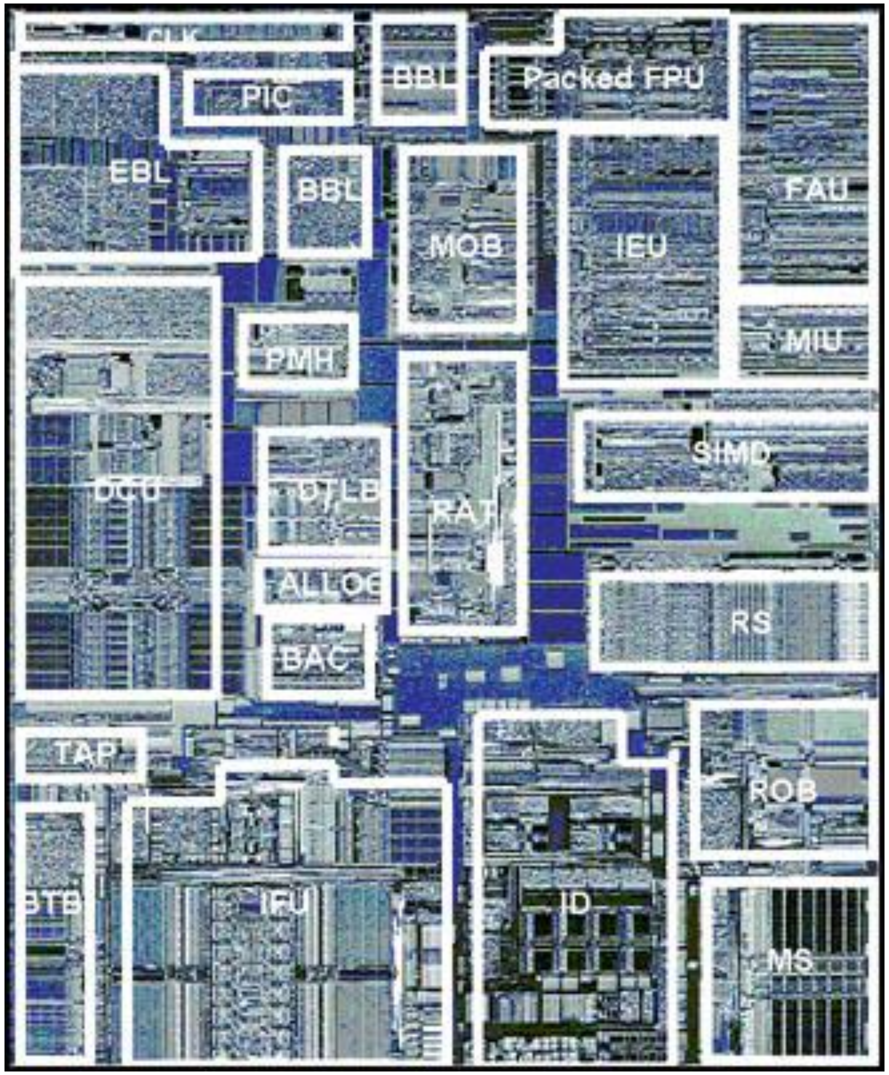
- Level two consists of 16 two-bit counters.
- Level one is a four bit shift register storing the history of the last four events. This four bit pattern is used to select which of the 16 two-bit counters to use for the next prediction.

Two-level prediction

- This mechanism can learn to recognize repetitive patterns. Imagine a branch that jumps every second time. You can write this pattern as 01010101 where 0 means no-jump and 1 means jump. After 0101 always comes an 0. Every times this happens, the counter with the binary number [0101] will be decremented until it reaches its lowest state. It has now learned that after 0101 comes an 0 and will therefore make this prediction correctly the next time. Similarly, counter number [1010] will be incremented until state three so that it will always predict a 1 after 1010. The remaining fourteen counters for this branch are never used as long as the pattern is the same. This mechanism is quite powerful as it can handle complex repetitive patterns like 00101-00101-00101. In fact, it can handle any repetitive pattern with a period of up to five, most patterns of period six and seven, and even some patterns with periods as high as sixteen. To see if a pattern of period *n* can be handled without misprediction, write down the *n* 4-bit sub-sequences in the pattern. If they are all different, then you will have no mispredictions after an initial learning time of two periods.
- It is also good at handling *deviations* from a regular pattern. If a branch instruction has an *almost regular* pattern with occasional deviations, then the processor will soon learn what the deviations look like, so that it can handle almost any kind of recurrent deviation with only one misprediction.
- Furthermore, it can handle a situation where you alternate between two different repetitive patterns. Assume that you have given the processor one repetitive pattern until it has learned to handle it without mispredictions. Then another pattern. And then return to the first pattern. If the two patterns do not have any 4-bit subsequences in common, then they do not use the same counters, so the processor doesn't have to re-learn the first pattern. Therefore, it can handle the transitions back and forth between the two patterns with a minimum of mispredictions.

[most of these notes are taken directly from http://www.x86.org/articles/branch/branchprediction.htm]

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

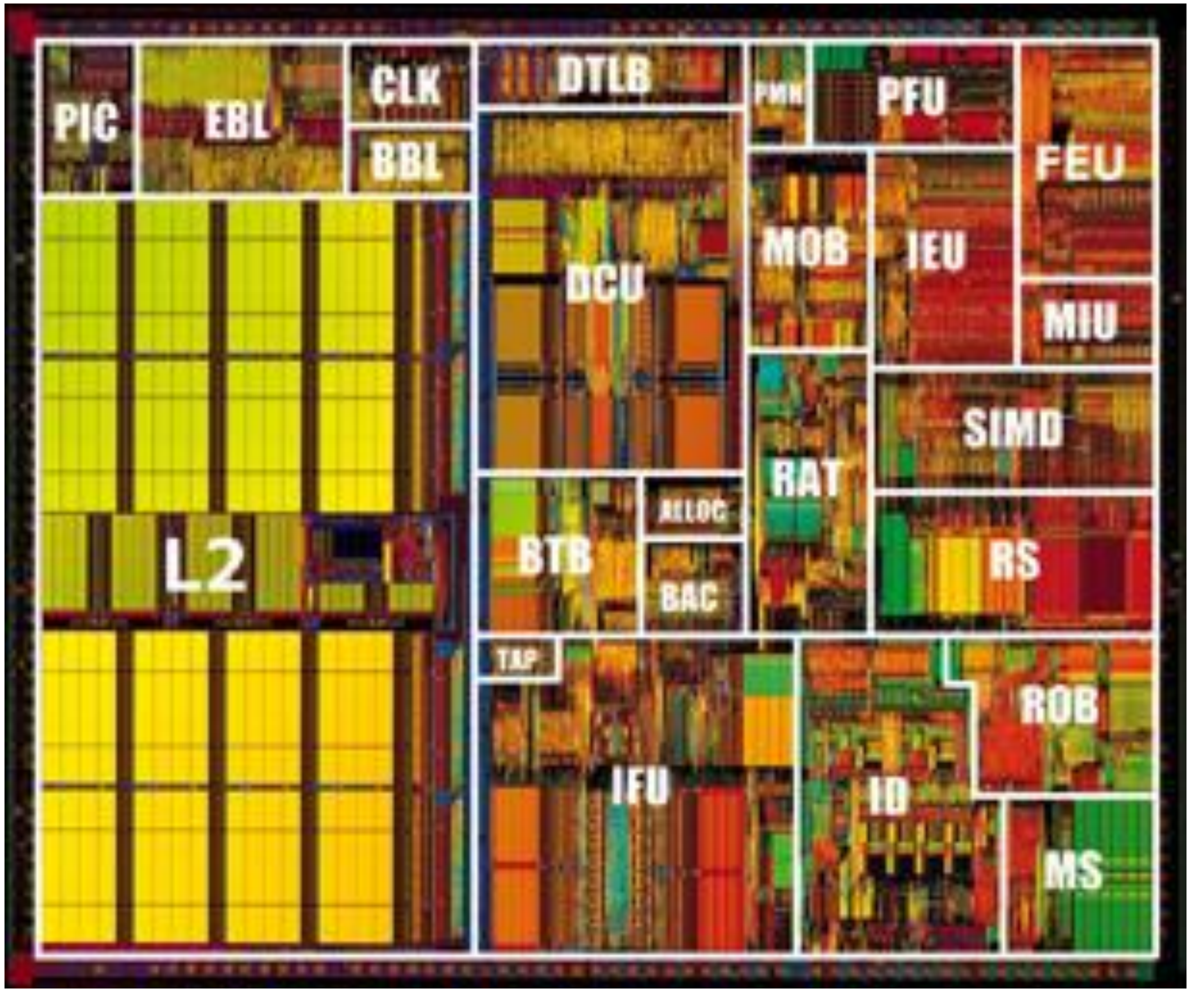
PMH - Page Miss Handler
DTLB - Data TLB
BAC - Branch Address Calculator
RAT - Register Alias Table
SIMD - Packed Fl. Pt.
RS - Reservation Station
BTB - Branch Target Buffer
IFU - Instruction Fetch Unit

ID - Instruction Decode
ROB - Reorder Buffer
MS - Micro-instruction Sequencer

1st Pentium III, Katmai: 9.5 M transistors, 12.3 * 10.4 mm in 0.25-mi. with 5 layers of aluminum

See: http://parasol.tamu.edu/people/dzhang/Courses/614/lec/Lec18-dynamic3.ppt

Later PIII with L2 cache



Later P3 die

Bottom centre region - Logic for the front-end of the pipeline resides here.

IFU - Instruction Fetch Unit. Instruction fetch logic and a 16K Byte 4-way set-associative level one instruction cache resides in this block. Instruction data from the IFU is then forwarded to the ID.

BTB - Branch Target Buffer. This block is responsible for dynamic branch prediction based on the history of past branch decisions paths.

BAC - Branch Address Calculator. Static branch prediction is performed here to handle the BTB miss case.

TAP - Testability Access Port. Various testability and debug mechanisms reside within this block.

Bottom right region - Instruction decode, scheduling, dispatch, and retirement functionality is contained within this region.

ID - Instruction Decoder. This unit is capable of decoding up to 3 instructions per cycle.

MS - Micro-instruction Sequencer. This holds the microcode ROM and sequencer for more complex instruction flows. The microcode update functionality is also located here.

RS - Reservation Station. Micro-instructions and source data are held here for scheduling and dispatch to the execution ports. Dispatch can happen out-of-order and is dependent on source data availability and an available execution port.

ROB - Re-Order Buffer. This supports a 40-entry physical register file that holds temporary write-back results that can complete out of order. These results are then committed to a separate architectural register file during in-order retirement.

Top right region - This primarily consists of the execution datapath

SIMD - SIMD integer execution unit for MMX™ Technology instructions.

MIU - Memory Interface Unit. This is responsible for data conversion and formatting for floating point data types.

IEU - Integer Execution Unit. This is responsible for ALU functionality of scalar integer instructions. Address calculations for memory referencing instructions are also performed here along with target address calculations for jump related instructions.

FEU - Floating point Execution Unit. This performs floating point related calculations for both existing scalar instructions along with support for some of the new SIMD-FP instructions.

PFU - Packed Floating point arithmetic Unit. This contains arithmetic execution data-path functionality for SIMD-FP specific instructions.

Top center region - Functionality in this region is split into assorted functions including data cache access, and allocation.

ALLOC - Allocator. Allocation of various resources such as ROB, MOB, and RS entries is performed here prior to micro-instruction dispatch by the RS.

RAT - Register Alias Table. During resource allocation the renaming of logical to physical registers is performed here.

MOB - Memory Order Buffer. Acts as a separate schedule and dispatch engine for data loads and stores. Also temporarily holds the state of outstanding loads and stores from dispatch until completion.

DTLB - Data Translation Look-aside Buffer. Performs the translation from linear addresses to physical address required for support of virtual memory.

PMH - Page Miss Handler. Hardware engine for performing a page table walk in the event of a TLB miss.

DCU - Data Cache Unit. Contains the non-blocking 16K Byte 4-way set-associative level one data cache along with associated fill and write back buffering.

Left region - Functionality in this area includes bus interface circuitry and a level 2 cache.

BBL - Back-side Bus Logic. Logic for interface to the back-side bus for accesses to the internal unified level two processor cache.

EBL - External Bus Logic. Logic for interface to the external front-side bus.

PIC - Programmable Interrupt Controller. Local interrupt controller logic for multi-processor interrupt distribution and boot-up communication.

CLK - Clocking. Contains phase lock loop and other clocking control circuitry.

L2 - Level 2 Cache. 256K unified level two cache.