Pentium vs Pentium II

(Research Essay #1)

Seok Bum Ko

kosby@ele.uri.edu

ELE548 Computer Architecture

12 February, 1999

1. Introduction

The Pentium is a widely-used personal computer microprocessor from the Intel Corporation. First offered in 1993, the Pentium quickly replaced Intel's 486 microprocessor as the microchip-of-choice in manufacturing a PC. The Pentium actually contains two processors on one chip that contains 3.1. million transistors.

The Pentium Pro, released in 1995, was designed for PC servers and workstations that needed to serve multiple users or needed the speed required for graphics-intensive applications. In addition to the microprocessor, the Pentium Pro includes another microchip containing cache memory that, being closer to the processor than the computer's main memory (RAM), speeds up computer operation. The Pentium Pro contains 5.5 million transistors.

The Pentium II is a Pentium Pro with Intel's MMX technology included. It comes in microprocessor clock speeds of 233 MHz (millions of cycles per second), 266 MHz, and 300 MHz. It's suitable for applications that include motion video and 3-D images. Among the Pentium II's features are a 512 KB (kilobyte) level-two (L2) memory cache and a 32 KB L1 cache, twice that of the Pentium Pro processor. The L2 cache can include error correcting code (ECC).

The Pentium III processor integrates the best features of the P6 microarchitecture processors - Dynamic Execution performance, a multi-transaction system bus, and Intel MMX media enhancement technology. In addition, the Pentium III processor offers Streaming SIMD Extensions -- 70 new instructions enabling advanced imaging, 3D, streaming audio and video, and speech recognition applications.

2. Comparison

2.1) Pentium

Intel's superscalar successor to the 486. It has the following features: Dual 32 bit 486-type integer pipelines with dependency checking; can execute a maximum of two instructions per cycle; Pipelined floating-point; 16 kilobyte of on-chip cache; Branch prediction; A 64 bit memory interface; 3.1 million transistors on a 262.4 mm2 die; ~2.3 million transistors in the core logic; clock rate 66MHz; heat dissipation 16W; integer performance 64.5 SPECint92; floating-point performance 56.9 SPECfp92; 8 32 bit general-purpose registers; 8 80 bit floating-point registers. It is called "Pentium" because it is the fifth in the 80x86 line. It would have been called the 80586 had a US court not ruled that you can't trademark a number.

2.1.1) **Superscalar architecture**

The Pentium processor's superscalar architecture is implemented using two integer pipelines, commonly referred to as the U and V- pipes. The two pipelines differ from each other only in that the U-pipe can execute the entire instruction set, whereas the V-pipe is only able to execute "simple" instructions because it does not have a barrel shifter as part of its ALU. The most optimum performance occurs when two instructions are being executed in parallel (one in each pipe). The more often this can occur, the better.

Each pipeline consists of five stages as denoted by the block diagram. These stages are Prefetch, Decode 1, Decode 2, Execute, and Writeback.

Pipelined Scalar Execution(10 clocks)

IF	D1	D2	E	WB					
	IF	D1	D2	E	WB				
		IF	D1	D2	E	WB			
			IF	D1	D2	E	WB		
				IF	D1	D2	E	WB	
					IF	D1	D2	E	WB

Pipelined Superscalar Execution(7 clocks);Assumes optimum pairing

IF	D1	D2	E	WB		
IF	D1	D2	E	WB		
	IF	D1	D2	E	WB	
	IF	D1	D2	E	WB	
		IF	D1	D2	E	WB
		IF	D1	D2	E	WB

2.1.2) **Enhanced Bus Interface Unit**

The addition of the V-pipe and increased internal clock speeds to the Pentium processor have boosted performance so that as many as two instructions per clock cycle can now be executed. A natural effect of these enhancements is the need to increase the rate at which code and data can be read from and written to external memory and I/O, in order to prevent the execution pipelines from stalling. To achieve this goal, the Bus Interface Unit of the Pentium processor has been dramatically enhanced.

The Bus Interface Unit provides the physical interface between the Pentium processor and the rest of the system. There are several different functional subsystems that make up this unit, all of which have been enhanced, but three that specifically need attention:

Data bus transceivers, Bus control logic, and Write buffers

• Data bus transceivers

The Pentium processor's data bus is 64 bits wide and supports a 4-transfer (32-byte) burst cycle for reads and writes (writebacks). The increase from 32 to 64 bits was needed to supply code data to the dual integer pipelines, and was required to keep the external data bus from being the bottleneck in the processors pipelines. The data bus transceivers drive data on to the processor's local bus during write cycles (writebacks), and gate data into the processor during read cycles.

• Bus control logic

The bus control logic controls whether standard or burst bus cycles are run by the processor. Standard bus cycles are either 8, 16, or 32 bits wide depending on the instruction and are used to access I/O locations, non-cacheable memory locations, or cacheable memory write operations. Burst cycles are 32 bytes (four quad-words) and are run by the processor during cache line fills or cache write-back bus cycles to the data cache.

• Write buffers

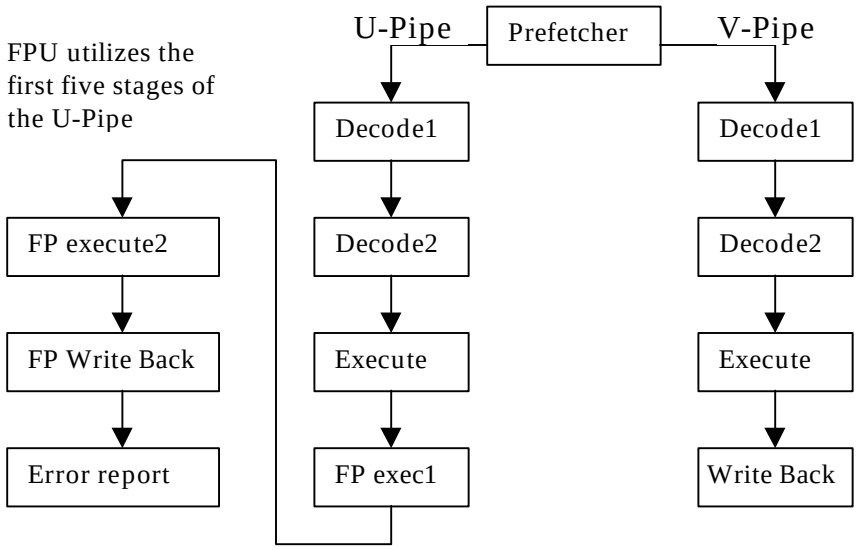
The Pentium processor has two write buffers, one corresponding to each of the integer pipelines enhancing performance of consecutive writes to memory. Each buffer is 64 bits wide and both can be filled simultaneously in one clock cycle. For example, if two simultaneous writes occur (one from each integer pipeline) that are both cache misses, both writes are placed in the write buffers on the same clock. If the Bus Unit is busy running a cycle, then the write is placed in the buffer and the execution unit is allowed to continue.

2.1.3) **The Floating Point Unit on the Pentium**

Pentium processor has been significantly improved over the Intel486 processor Floating Point Unit. It consists of an eight-stage pipeline which, under certain circumstances, permits several instructions to be executed simultaneously. The result is that it is capable of floating point performance up to five times that of the Intel486 DX2/66 processor, and up to ten times that of the Intel486 DX processor.

The Floating Point Pipeline utilizes the U-pipe of the integer pipeline, and adds three additional stages to it. The following diagram shows the additional stages of the Floating Point Unit, and the enhancements to the integer pipeline when executing floating point instructions. The first five stages are shared with the integer pipeline. When executing a floating point instruction the Prefetch, Decode 1, and Decode 2 stages of the integer pipeline perform the exact same functions for the floating point instruction as they do for an integer instruction. The last two stages of the integer pipeline perform slightly different functions when executing a floating point instruction. When executing a floating point instruction, the integer execute stage performs a register read, memory read, or memory write as required by the instruction. The integer writeback stage becomes the first floating point execute stage. In this stage, information from a register or memory is written into a floating point register. In addition, data is converted to floating point format before being loaded into the Floating Point Unit. No ALU function is performed for floating point operation as it is for integer operation, just the register read/write or memory write. The last three stages of the floating point pipeline reside within the Floating Point Unit itself. They are:

FP EX2 - Floating point operation performed
FP WB - Results are rounded and written to target floating point register
Error Report - Error detected, enters stage and Floating Point Unit status word is updated



The floating point pipeline is integrated into the integer pipeline and under normal circumstances, is capable of delivering up to 1 result/clock. However, the floating point exchange (FXCH) instruction can be paired with other simple floating point instructions as long as it is always the second instruction in the pair. This is because FXCH is the only floating point instruction that can be executed in the V-pipe. This provides an important performance boost as operands used by floating point instructions must be on the top of the floating point register stack in order to be accessed. This requirement results in a lot of FXCH instructions in floating point code, so the ability to execute them in parallel with other floating point instructions

provides a significant performance boost. Finally, the resources within the Floating Point Unit can be allocated such that pipelining within the Floating Point Unit itself is permitted. The following conditions apply:

1.The FDIV instruction cannot execute in parallel with any other instruction.
2.Two consecutive FMUL instructions cannot execute in parallel.
3.One FMUL instruction can execute in parallel with one or two FADD instructions.
4.Three FADD instructions can be executed in parallel.

2.1.4) **Branch Target Prediction**

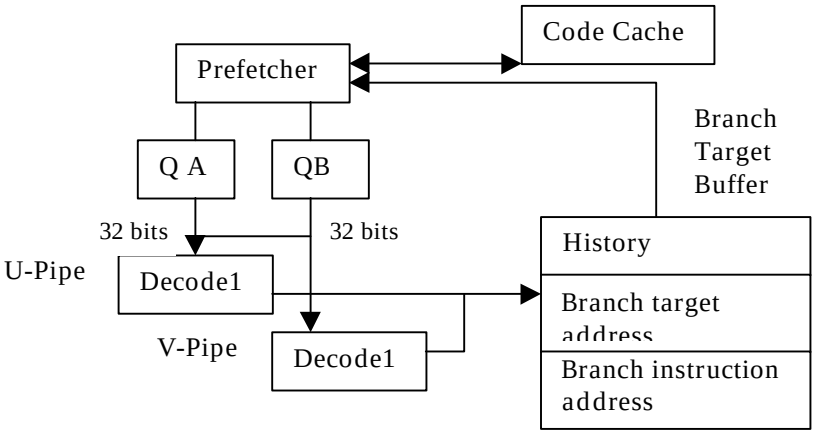
One of the inherent issues that occurs when instructions are prefetched and queued in a prefetch buffer is how to handle conditional branch instructions. Prefetchers in older processors assumed that the execution unit would want the next instruction from the next sequential memory location.

The problem

The problem with this assumption is that if the branch was taken, it altered the program flow and the instruction to be executed would not be the next sequential instruction in memory. The result is that the instructions in both the pipeline and the queue would be incorrect, and both would have to be flushed. The pipeline and queue would have to then be refilled with correct instructions from the Branch Target Address. While this "refilling" takes place the execution unit is stalled, resulting in performance degradation

The solution

The answer to this dilemma was to add Branch Prediction. Branch Prediction logic permits the Pentium processor to make more intelligent decisions regarding what information to prefetch from memory. The prefetcher has two separate prefetch queues (A and B), but only one of them is used at a time.



When a branch instruction is fetched, the prediction logic predicts whether it will be taken or not (it does this by examining the execution history of the instruction). If the branch is not taken, it continues fetching instructions from the next sequential memory location of the current queue. If it predicts the branch will be taken, the prefetcher will switch to the other queue and begin fetching instructions from memory, starting at the branch target address. The prediction is accomplished using the Branch Target Buffer (BTB). The BTB is capable of storing the information for up to 256 unique branch instructions. The BTB contains the following information: Branch Instruction Address, Branch Target Address, History Bits. The initial prediction for any branch instruction is not taken, as no history has been established. The history bits are updated when the instruction is evaluated in the Execute stage of the pipeline. It takes two consecutive mispredictions for the branch condition to switch from taken to not taken or vice versa. The

initial prediction of a branch instruction is "not taken" (this is true even for Unconditional branches). As soon as a branch instruction is taken for the first time, the BTB is initialized with the Branch Instruction Address and the Branch Target Address, and the history bits are set to strongly taken. When predictions are correct, pipeline stalls are prevented. The prefetcher fetches code from the correct address (either the Branch Target Address if the branch is taken, or the next sequential address after the branch instruction address if not taken) and begins decoding the instructions. The prediction is verified in the "execute stage". If the prediction is incorrect, the pipeline is flushed and a stall occurs while the pipe is refilled. A three-clock penalty occurs for any branch instruction mispredicted in the U-pipe, and for conditional branches mispredicted in the V-pipe. A four-clock penalty occurs for unconditional branches mispredicted in the V-pipe. Most importantly, this feature is completely software transparent, and current existing code benefits from the performance gain.

2.1.5) **Separate 8K code and data caches**

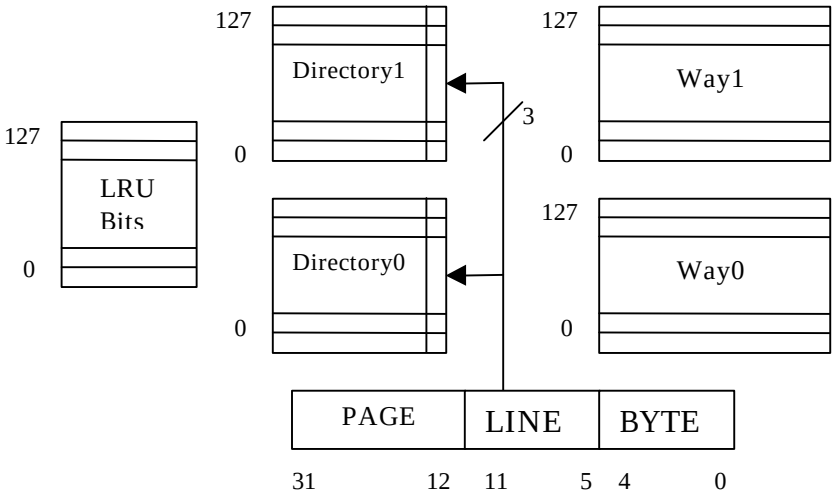
If you recall from the early part of this module we spoke about how the speeds of new Processors increased at a much faster rate than the speeds of memory devices. When processor speeds reached 20-25MHz reasonably priced DRAMs could no longer keep up without adding wait states. The result was that memory performance throttled the execution performance of the processors. In order to gain the benefit of these faster processors it was necessary to use faster RAM. Since the faster SRAMs are also much more costly, the desire was to use as little as possible while still reaping the benefits. This cost / performance tradeoff was implemented as cache memory. The idea is for the cache memory to hold copies of code / data that is accessed frequently during the execution of the code. In early processors this cache memory was always implemented externally, requiring an external bus cycle to be executed for every memory access. Starting with the Intel486 processors, the cache memory was moved onto the processor chip itself, eliminating the bus speed bottleneck.

Cache separation

Another problem that soon began to affect performance was that both code and data accesses were occurring to the same memory. This would create contention and often would result in the cache controller servicing the execution unit first (the data read / write), and then servicing the prefetcher's request. This could lead to starving the pipeline, resulting in poorer performance. To alleviate this problem, the Pentium processor includes separate cache memories—one for code and one for data. This permits the execution units to submit requests to the data cache at the same time the code prefetcher submits requests to the code cache. In the classic Pentium processor (no MMX technology) each cache memory is 8K in size with each cache line being 32 bytes wide. Each cache memory is 8Kbytes deep organized in two banks of 4Kbytes. Each bank is referred to as a "Way" (Way 0 and Way 1), and each Way is organized as 128 lines x 32 bytes. Since there are two "Ways", the Pentium Processors cache is referred to as "two way set associative". For more detail on two way set associativity, refer to the Pentium Processor Family Developers manual (located in the Resources section).

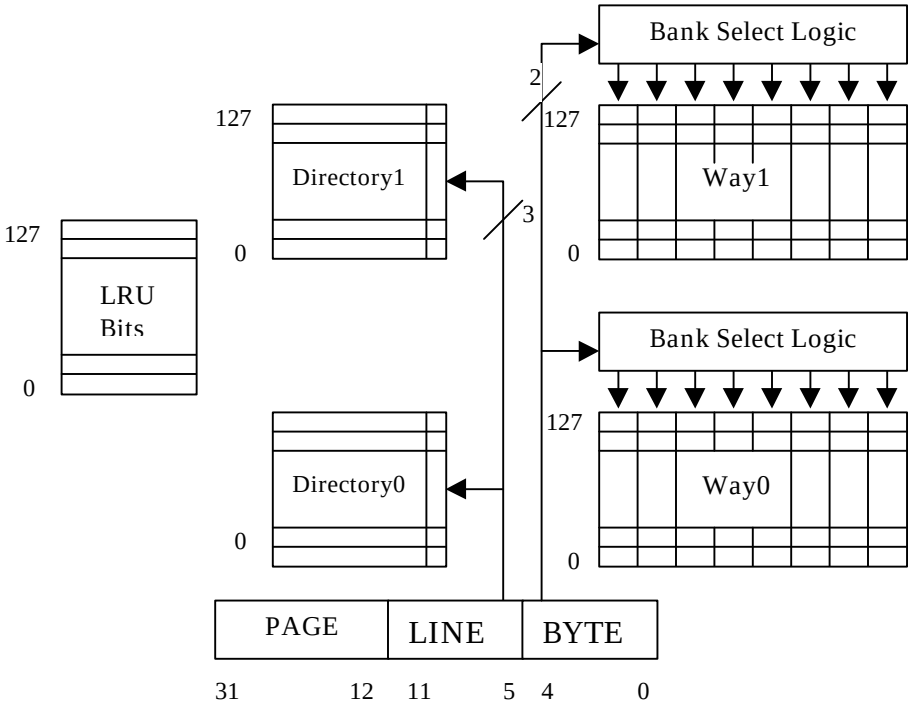
Code Cache Organization

Each Way has its own lookup directory (often called "Tag RAM") that stores the starting address of each line in the cache memory. Any given line of code from main memory can be stored either in way 0 or way 1, but must be stored in the corresponding line of the way. Since there are many more lines of code in main memory than there are in cache memory it's often necessary to replace an existing line in cache with another line from main memory. The algorithm used to determine which cache line to replace is called the "Least Recently Used" algorithm. A set of bits called LRU bits identify which line is the least recently used, indicating to the cache controller where to store the new line. The Code Cache provides the prefetcher with an entire cache line, 32 bytes at a time.



Data Cache Organization

The Data Cache is organized the same as the Code Cache, with two primary exceptions. First, The data in the cache is selectable on double word boundaries. This means that when data is read from the data cache it does not have to load an entire cache line (32 bytes) at a time, but rather can be loaded in increments of 4 bytes (a double word) at a time. Second, parity protection is done on a per-byte basis in the data cache.



ENHANCED ERROR DETECTION FEATURES: The 80486 introduced parity checking-based error detection as part of the processor, but the error detection was restricted to the external data bus only. Pentium carries this process further and does parity-based error detection

for both the external data bus and the external address bus. In addition, all internal data structures are parity protected, including the code and data caches, cache tags, TLBs, microcode ROM, and the branch target buffer. For ultimate security, fault tolerant systems can configure one Pentium chip to check another on a cycle-by-cycle basis, signaling any inconsistency.

SYSTEM MANAGEMENT MODE: Intel had long promised developers that the System Management Mode (SMM) would be made a standard part of future processors and no longer be restricted to chips of the SL series. Pentium is the first chip to live up to the promise. As in the 80386 SL and 80486 SL, the SMM is a unique processor operating mode beyond all other normal processor modes. The SMM capability is targeted mainly toward portable applications-such as notebook computers. The SMM mode allows the implementation of power-saving functions, such as powering down peripherals when they are not in use and restarting them when they are accessed.

A 64-BIT DATA BUS: Pentium has a 64-bit high-speed data bus with sophisticated protocols. The bus can transfer up to 8 bytes of data at a time. Each byte can be enabled separately. Parity checking is available. Pentium uses even parity and parity is checked only for enabled bytes. The Pentium bus contains several new features over the 80486 data bus which help increase throughput. The flip side is that the complexity of the bus controllers required also goes up.

OTHER ENHANCEMENTS: Among the other enhancements in Pentium are some new instructions such as the CPUID instruction which can be used to identify the CPU. Calling this instruction with a 0 in the EAX causes the string 'Genuine Intel' to be returned. The exception model has been extended. Finally, Pentium can support pages which are 4 MB in size in addition to the normal 4 KB size pages.

2.2)Pentium II

2.2.1) **Dynamic Execution**

First used in the Pentium Pro processor, Dynamic Execution is an innovative combination of three processing techniques designed to help the processor manipulate data more efficiently. These are multiple branch prediction, data flow analysis, and speculative execution.
Dynamic execution enables the processor to be more efficient by manipulating data rather than simply processing a list of instructions. The way software programs are written can influence a processor's performance. For example, software performance will be adversely affected if the processor is frequently required to stop what it is doing and "jump" or "branch" elsewhere in the program. Delays can also occur when the processor isn't able to process a new instruction until the current instruction is completed. Dynamic execution allows the processor to streamline and predict the order of instructions.

• Multiple Branch Prediction

Predicts the flow of the program through several branches: Using a multiple branch prediction algorithm, the processor can anticipate jumps in the instruction flow. It predicts where the next instructions can be found in memory with an amazing 90% or greater accuracy. This is made possible because, while the processor is fetching instructions, it's also looking at instructions further ahead in the program. This technique accelerates the flow of work sent to the processor.

• Data Flow Analysis

Analyzes and schedules instructions to be executed in an optimal sequence, independent of the original program order: Using data flow analysis, the processor looks at decoded software instructions and determines if they're available for processing or if they're dependent on other instructions. The processor then determines the optimal sequence for processing and executes the instructions in the most efficient manner.

• Speculative Execution

Increases the rate of execution by looking ahead of the program counter and executing instructions that are likely to be needed: When the processor executes instructions (up to five at a time), it does so using "speculative execution." This keeps the Pentium II processor's superscalar processing power as busy as possible to boost software performance. Since the software instructions being processed are based on predicted branches, the results are stored as "speculative results." Once their final state can be determined, the instructions are returned to their proper order and committed to permanent machine state.

2.2.2) **MMX Technology**

The Pentium II processor incorporates Intel MMX (MultiMedia Extension) technology—Intel's most significant enhancement to the Intel architecture in the last 10 years. MMX technology improves video compression / decompression, image manipulation, encryption and I / O processing—all of which are used in a variety of today's office suites and advanced business media, communications and Internet capabilities.

<How It Works>

Single Instruction, Multiple Data (SIMD) Technique

Today's multimedia and communication applications often use repetitive loops that, while occupying 10 percent or less of the overall application code, can account for up to 90 percent of the execution time. SIMD permits one instruction cycle to act on multiple data pieces thus reducing run time (hence speeding up the program execution). Perhaps the best analogy of SIMD is that of a Drill Sergeant. The Drill Sergeant could address each recruit individually with "about face" (Non-MMX Technology). Or the Drill Sergeant could (loudly) say "About face" to all the recruits simultaneously and all recruits would turn at the same time (MMX Technology).

New Instructions

Intel engineers have also added 57 powerful new instructions specifically designed to manipulate and process video, audio and graphical data more efficiently. These instructions are oriented to the highly parallel, repetitive sequences often found in multimedia operations. While the Pentium II processor's MMX technology is binary compatible with that used in the Pentium processor with MMX technology, it is also combined synergistically with the Pentium II processor's advanced core architecture. Powerful MMX technology instructions take full advantage of the efficient processing techniques of Dynamic Execution—delivering the best in media and communications capabilities.

2.2.3) **Dual Independent Bus**

In order to meet the demands of applications and to provide headroom for future generations of processors, Intel has developed the Dual Independent Bus architecture to address bandwidth limitations in the current PC platform architecture. The Dual Independent Bus architecture was first implemented in the Pentium Pro processor and will become broadly available with the Pentium II processor. Intel created the dual independent bus architecture to aid processor bus bandwidth. Having two (dual) independent buses enables the Pentium II processor to access data from either of its buses simultaneously and in parallel, rather than in a singular sequential manner as in a single bus system. The Pentium and Pentium with MMX Technology have a single bus to Cache and L2. Because the Pentium II is so much faster, Intel added a faster, second bus to cache. These two pathways to cache and main memory operate independently. When combined with dynamic execution (described above), DIB dramatically improves performance**.**

<How it works>

• Two buses make up the Dual Independent Bus architecture:
The "L2 cache bus" and the processor-to-main-memory "system bus."
• The Pentium II processor can use both buses simultaneously.
• The Dual Independent Bus architecture enables the L2 cache of the 400MHz Pentium II processor, for example, to run more than three times as fast as the L2 cache of Pentium

processors. As the frequency of future Pentium II processors increases, so will the speed of the L2 cache.

• The pipelined system bus enables multiple simultaneous transactions (instead of singular sequential transactions), accelerating the flow of information within the system and boosting overall performance.

Together these Dual Independent Bus architecture improvements offer up to three times the bandwidth performance over a single bus architecture processor. In addition, the Dual Independent Bus architecture supports the evolution of today's 100MHz system bus. This high-bandwidth bus technology is designed to work in concert with the high-performance processing power of the Pentium II processor.

2.2.4) **Single Edge Contact**

The Single Edge Contact (S.E.C.) cartridge is Intel's innovative packaging design that enables even higher performance to be delivered to mainstream systems. Using this technology, the core and L2 cache are fully enclosed in a plastic and metal cartridge. These sub-components are surface mounted directly to a substrate inside the cartridge to enable high-frequency operation. The S.E.C. cartridge technology allows the use of widely available, high-performance industry BSRAMs for the dedicated L2 cache, enabling high-performance processing at mainstream price points. Also, this cartridge technology allows the Pentium II processor to use the same high performance Dual Independent Bus architecture used in the Pentium Pro processor.

The Pentium II processor connects to a motherboard via a single edge connector instead of the multiple pins used in existing PGA packages. Similarly, the 242-contact slot connector connector replaces the PGA socket used in prior systems. Future versions of the Pentium II processor will also be compatible with the 242-contact slot connector.

3. References

1. Hay98 John P. Hayes, 1998 Computer architecture and organization, McGraw-Hill
2. Hen96 John L. Hennesy and David A Patterson, 1996 Computer Architecture A quantitative approach, Morgan Kaufman
3. http://www.intel.com
4. Tome Shanley, Pentium pro and pentium II system architecture(PC system architecture series), Mindshare Inc.

Table 1. CPU Comparison

Processor	Introduced	Clock speeds (MHz)	System bus speed (MHz)	L1 cache	Integrated L2 cache	Speed of L2 cache interface (MHz)	Special	Process technology	Packaging
Intel 386	17-Oct-85	16, 20, 25, 33	Speed of processor	None	None	N/A	None	1µ BiCMOS	Pin-grid array
Intel 486	10-Apr-89	25, 33, 50, 66, 75, 100	25, 33, 50	8K, 16K	None	Speed of system bus	None	1µ, 0.8µ BiCMOS	Pin-grid array
Intel Pentium	**22-Mar-93**	**60, 66, 75, 90, 100, 120, 133, 150, 166, 200**	**50, 60, 66**	**16K**	**None**	**Speed of system bus**	**None**	**0.8µ, 0.6µ, 0.35µ BiCMOS**	**Pin-grid array**
Intel Pentium Pro	1-Nov-95	150, 166, 180, 200	60, 66	16K	256K, 512K, 1MB	Speed of processor	None	0.6µ, 0.35µ BiCMOS	Dual-cavity-pin grid array
Intel Pentium with MMX	8-Jan-97	166, 200, 233	66	32K	None	Speed of system bus	MMX	0.35µ CMOS	Pin-grid array (Socket 7-compatible)
Intel Pentium II (Klamath)	**7-May-97**	**233, 266, 300**	**66**	**32K**	**512K (off-chip but in SEC cartridge)**	**Half the speed of the processor**	**MMX**	**0.35µ CMOS**	**Single-edge-contact cartridge**
Intel Pentium II, 66Mhzbus Deschutes	26-Jan-98	333	66	32K	512K (off-chip but in SEC cartridge)	Half the speed of the processor	MMX	0.25µ CMOS	Single-edge-contact cartridge
Intel Pent-iumII, 100 Mhz bus Deschutes	15-Apr-98	350, 400	100	32K	512K (off-chip but in SEC cartridge)	Half the speed of the processor	MMX	0.25µ CMOS	Single-edge-contact cartridge
Intel Celeron	Apr15,1998	266	66	32K	None	N/A	MMX	0.25µ CMOS	Single-edge processor package
Intel Slot 2 processor	1st half of 1998	Unknown	100	N/A	Up to 2MB	Speed of processor	MMX	0.25µ CMOS	Unknown
AMD K6	2-Apr-97	166, 200, 233, 266	66	64K	None	Speed of system bus	Licensed MMX	0.35µ, 0.25µ CMOS	Pin-grid array (Socket 7-compatible)
AMD K6 3D	1st half of 1998	300, 350	66, 100	64K	None	Speed of system bus	Licensed MMX, 3-D	0.25µ CMOS	Pin-grid array (Socket 7-compatible)
AMD K6+ 3D	2nd half of 1998	350, 400	66, 100	64K	256K	Speed of processor	Licensed MMX, 3-D	0.25µ CMOS	Pin-grid array (Socket 7-compatible)
Cyrix 6x86	5-Feb-96	133 (PR 166), 150 (PR 200)	60, 66, 75	16K	None	Speed of system bus	None	0.35µ CMOS	Pin-grid array (Socket 7-compatible)
Cyrix MediaGX	20-Feb-97	133, 150, 180, 233	66	16K	None	N/A	MediaGX 2, MMx 3	0.35µ CMOS	Ball-grid array
Cyrix 6x86MX	30-May-97	150 ,166, 188,200, 233,266, 300,350	60, 66, 75, 83, 100	64K	None	Speed of system bus	MMX, 3-D	0.35µ, 0.25µ CMOS	Pin-grid array (Socket 7-compatible)
Cyrix MXi	2nd half of 1998	PR300, PR400	133	64K	None	Speed of system bus	MMX, 3-D	0.25µ CMOS	Pin-grid array (Socket 7-compatible)
IDT WinChip	13-Oct-97	180, 200, 220, 240	60, 66	64K	None	Speed of system bus	MMX	0.35µ CMOS	Pin-grid array (Socket 7-compatible)
IDT C6+	1st half of 1998	300	66, 75, 100	64K	None	Speed of system bus	MMX, 3-D	0.25µ CMOS	Pin-grid array (Socket 7-compatible)
IDT C6+ with L2 cache	2nd half of 1998	300	66, 100	64K	256K	Speed of processor	MMX, 3-D	0.25µ CMOS	Pin-grid array (Socket 7-compatible)