Extract from: Great Microprocessors of the Past and Present

http://www.cs.uregina.ca/~bayko/cpu.html

The Zilog Z-80 - End of an 8-bit line (July 1976)

The Z-80 was intended to be an improved 8080 (designed by ex-Intel engineers), and it was - vastly improved. It also used 8 bit data and 16 bit addressing, and could execute all of the 8080 (but not 8085) op codes, but included 80 more, instructions (1, 4, 8 and 16 bit operations and even block move and block I/O). The register set was doubled, with two banks of data registers (including A and F) that could be switched between. This allowed fast operating system or interrupt context switches. The Z-80 also added two index registers (IX and IY) and 2 types of relocatable vectored interrupts (direct or via the 8-bit I register).

Clock speeds ranged from the original Z-80 2.5MHz to the Z80-H (later called Z80-C) at 8MHz.

Like many processors (including the 8085), the Z-80 featured many undocumented instructions. In some cases, they were a by-product of early designs (which did not trap invalid op codes, but tried to interpret them as best they could), and in other cases chip area near the edge was used for added instructions, but fabrication made the failure rate high. Instructions that often failed were just not documented, increasing chip yield. Later fabrication made these more reliable.

But the thing that really made the Z-80 popular in designs was the memory interface - the CPU generated its own RAM refresh signals, which meant easier design and lower system cost, the deciding factor in its selection for the TRS-80 Model 1. That and its 8080 compatibility, and CP/M, the first standard microprocessor operating system, made it the first choice of many systems.

Embedded varients of the Z-80 were also produced. Hitachi produced the 64180 (1984) with added components (two 16 bit timers, two DMA controllers, three serial ports, and a segmented MMU mapping a 20 bit (1M) address space to any three variable sized segments in the 16 bit (64K) Z-80 memory map), a design Zilog and Hitachi later refined to produce the Z-180 and HD64180Z (1987?) which were compatible with Z-80 peripheral chips, plus variants (Z-181, Z-182). The Z-280 was a 16 bit version introduced about July, 1987 (loosely based on the ill-fated Z-800), with a paged (like Z-180) 24 bit (16M) MMU (8 or 16 bit bus resizing), user/supervisor modes and features for multitasking, a 256 byte (4-way) cache, 4 channel DMA, and a huge number of new op codes tacked on (total of almost 3,500, including previously undocumented Z-80 instructions), though the size made some very slow. Internal clock could be run at twice the external clock (ex. 16MHz CPU with a 8MHz bus), and additional on-chip components were available. A 16/32 bit Z-380 version also exists (1994) with added 32-bit linear addressing mode (not Z-80 compatible).

The Z-8 (1979) was an embedded processor with on-chip RAM (actually a set of 124 general and 20 special purpose registers) and ROM (often a BASIC interpreter), and is available in a variety of custom configurations up to 20MHz. Not actually related to the Z-80.

Zilog Corporation:

http://www.zilog.com/
20th Anniversary of the TRS-80:
http://www.radioshack.com/trs_80/

The 650x, Another Direction (1975)

Shortly after Intel's 8080, Motorola introduced the 6800. Some of the designers left to start MOS Technologies (later bought by Commodore), which introduced the 650x series which included the 6501 (pin compatible with the 6800, taken off the market almost immediately for legal reasons) and the 6502 (used in early Commodores, Apples and Ataris). Like the 6800 series, varients were produced which added features like I/O ports (6510 in the Commodore 64) or reduced costs with smaller address buses (6507 13-bit 8K address bus in the Atari 2600). The 650x was little endian (lower address byte could be added to an index register while higher byte was fetched) and had a completely different instruction set from the big endian 6800. Apple designer Steve Wozniak described it as the first chip you could get for less than a hundred dollars (actually a quarter of the 6800 price) - it became the CPU of choice for many early home computers (8 bit Commodore and Atari products).

Unlike the 8080 and its kind, the 6502 (and 6800 had very few registers. It was an 8 bit processor, with 16 bit address bus. Inside was one 8 bit data register, two 8 bit index registers, and an 8 bit stack pointer (stack was preset from address 256 ($100 hex) to 511 ($1FF)). It used these index and stack registers effectively, with more addressing modes, including a fast zero-page mode that accessed memory addresses from address 0 to 255 ($FF) with an 8-bit address that speeded operations (it didn't have to fetch a second byte for the address).

Back when the 6502 was introduced, RAM was actually faster than microprocessors, so it made sense to optimize for RAM access rather than increase the number of registers on a chip. It also had a lower gate count (and cost) than its competitors.

The 650x also had undocumented instructions.

The CMOS 65C02/65C02S fixed some original 6502 design flaws, and the 65816 (officially W65C816S, both designed by Bill Mensch of Western Design Center Inc.) extended the 650x to 16 bits internally, including index and stack registers, with a 16-bit direct page register (similar to the 6809), and 24-bit address bus (16 bit registers plus 8 bit data/program bank registers). It included an 8-bit emulation mode. Microcontroller versions of both exist, and a 32-bit version (the 65832) is planned. Various licensed versions are supplied by GTE (16 bit G65SC802 (pin compatible with 6502), and G65SC816 (support for VM, I/D cache, and multiprocessing)) and Rockwell (R65C40), and Mitsubishi has a redesigned compatible version. The 6502 remains surprisingly popular largely because of the variety of sources and support for it.

The 6502-based Apple II line (not backwards compatible with the Apple I) was among the first microcomputers introduced and became the longest running PC line, eventually including the 65816-based Apple IIgs The 6502 was also used in the Nintendo entertainment system (NES), and the 65816 is in the 16-bit successor, the Super NES.

The Western Design Center, Inc.:
http://www.wdesignc.com

DEC PDP-11, benchmark for the first 16/32 bit generation. (1970)

The DEC PDP-11 was the most popular in the PDP (Programmed Data Processors) line of minicomputers, a successor to the previously popular PDP-8 (the PDP-10 (1967) was a higher capacity 36-bit mainframe-like version of the PDP-8, much adored and rumoured to have souls), and remained in production until the decision to discontinue the line as of September 30, 1997 (over 25 years - see note on the DEC Alpha intended lifetime).

The PDP-11 had eight general purpose 16-bit registers (R0 to R7 - R6 was also the SP and R7 was the PC). It featured powerful register oriented (little-endian, byte addressable) addressing modes. Since the PC was treated as a general purpose register, constants were loaded using an indirect mode on R7 which had the effect of loading the 16 bit word following the current instruction, then incrementing the PC to the next instruction before fetching. The SP could be accessed the same way (and any register could be used for a user stack (useful for FORTH)). A CC (or PSW) register held results from every instruction that executed.

Adjascent registers could be implicitly grouped into a 32 bit register for multiply and divide results (Multiply result stored in two registers if destination is an even register, not if it’s odd. Divide source must be grouped - quotient is stored in high order (low number) register, remainder in low order).

A floating point unit could be added which contains six 64 bit accumulators (AC0 to AC5, can also be used as six 32-bit registers - values can only be loaded or stored using the first four registers).

PDP-11 addresses were 16 bits, limiting program space to 64K, though an MMU could be used to expand total address space (18-bits and 22-bits in different PDP-11 versions).

The LSI-11 (1975-ish) was a popular microprocessor implementation of the PDP-11 using the Western Digital MCP1600 microprogrammable CPU, and the architecture influenced the Motorola 68000, NS 320xx, and Zilog Z-8000 microprocessors in particular. There was also a 32-bit PDP-11 plan as far back as its 1969 introduction. The PDP-11 was finally replaced by the VAX architecture, (early versions included a PDP-11 emulation mode, and were called VAX-11).

PDP-11 FAQ:
http://www.village.org/pdp11/faq.pages/faq.html

Motorola 68000, a refined 16/32 bit CPU (1979)

The initial 8MHz 68000 was actually a 32 bit architecture internally, but had only a 16 bit data bus and

24 bit address bus to fit in a 64 pin package (address and data shared a bus in the 40 pin packages of the 8086 and Z-8000). Later the 68008 reduced the data bus to 8 bits and address to 20 bits, and the 68020 was fully 32 bit externally. Addresses were computed as 32 bits (without using segment registers) - unused upper bits in the 68000 or 68008 bits were ignored, but some programmers stored type tags in the upper 8 bits, causing compatibility problems with the 68020's 32 bit addresses. Lack of forced segments made programming the 68000 easier than some competing processors, without the 64K size limit on directly accessed arrays or data structures.

Looking back it was a logical design decision, since most 8 bit processors featured direct 16 bit addressing without segments.

The 68000 had sixteen 32-bit registers, split into eight data and address registers. One address register was reserved for the Stack Pointer. Data registers could be used for any operation, including offset from an address register, but not as the source of an address itself. Operations on address registers were limited to move, add/subtract, or load effective address.

Like the Z-8000, the 68000 featured a supervisor and user mode (each with its own Stack Pointer). The Z-8000 and 68000 were similar in capabilities, but the 68000 was 32 bit units internally (16 bit ALUs, two in parallel for 32-bit data, one for addresses), making it faster and eliminating forced segments. It was designed for expansion, including specifications for floating point and string operations (floating point was added in the 68040 (1991), with eight 80 bit floating point registers compatible with the 68881/2 coprocessors). Like many other CPUs of the time, the 68000 could fetch the next instruction during execution (a 2 stage pipeline).

The 68010 added virtual memory support (the 68000 couldn't restart interrupted instructions) and a special loop mode - small decrement-and-branch loops could be executed from the instruction fetch buffer. The 68020 (1984) expanded external data and address bus to 32 bits,simple 3-stage pipeline, and added a 256 byte cache, while the 68030 brought the MMU onto the chip (it supported two level pages (logical, physical) rather than the segment/page mapping of the Intel 80386 and IBM S/360 mainframe). The 68040 (1989/90) added fully cached Harvard busses (4K(?) each for data and instructions), 6 stage pipeline, and on chip FPU.

The 68060 (late 1994) expanded the design to a superscalar version, like the Intel Pentium and NS320xx (Swordfish) series before it. Like the Nx586, AMD K5, and Intel's "Pentium Pro", the the third stage of the 10-stage 68060 pipeline translates the 680x0 instructions to a decoded RISC-like form (stored in a 16 entry buffer in stage four), and uses resource renaming (with fourty rename registers) to reorder instructions. There is also a branch cache, and branches are folded into the decoded instruction stream like the AT&T Hobbit and other more recent processors, then dispatched to two pipelines (three stages: Decode, addr gen, operand fetch) and finally to two of three execution units - 2 integer, 1 floating point) before reaching two 'writeback' stages. Cache sizes are doubled over the 68040.

The 68060 also also includes many innovative power-saving features (3.3V operation, execution unit pipelines could actually be shut down, reducing power consumption at the expense of slower execution, and the clock could be reduced to zero) so power use is lower than the 68040 (4-6 watts vs. 3.9-4.9). Another innovation is that simple register-register instructions which don't generate addresses may use the the address stage ALU to execute 2 cycles early.

The embedded market became the main market for the 680x0 series after workstation venders (and the

Apple Macintosh) turned to faster load-store processors, so a variety of embedded versions were introduced. Later, Motorola designed a successor called Coldfire (early 1995), in which complex instructions and addressing modes (added to the 68020) were removed and the instruction set was recoded, simplifying it at the expense of compatibility (source only, not binary) with the 680x0 line.

The Coldfire 52xx architecture resmbles a stripped (single pipeline) 68060, The 5 stage pipeline is literally folded over itself - after two fetch stages and a 12-byte buffer, instructions pass through the decode and address generate stages, then loop back so the decode becomes the operand fetch stage, and the address generate becomes the execute stage (so only one ALU is required for address and execution calculations). Simple (non-memory) instructions don't need to loop back. There is no translator stage as in the 68060 because Coldfire instructions are already in RISC-like form. The earlier 51xx version has a straight 68040-based 6 stage pipeline and includes 680x0 instructions (in user mode).

At a quarter the physical size and a fraction of the power consumption, Coldfire is about as fast as a 68040 at the same clock rate, but the smaller design allows a faster clock rate to be acheived.

Few people wonder why Apple chose the Motorola 68000 for the Macintosh, while IBM's decision to use Intel's 8088 for the IBM PC has baffled many. It wasn't a straightforward decision though. The Apple Lisa was the predecessor to the Macintosh, and also used a 68000 (eventually - 8086 and slower bitslice CPUs (which Steve Wozniak thought were neat) were initially considered before the 68000 was available). It also included a fully multitasking, GUI based operating system, highly integrated software, high capacity (but incompatible) 'twiggy' 5 1/4" disk drives, and a large workstation-like monitor. It was better than the Macintosh in almost every way, but was correspondingly more expensive.

The Macintosh was to include the best features of the Lisa, but at an affordable price - in fact the original Macintosh came with only 128K of RAM and no expansion slots. Cost was such a factor that the 8 bit Motorola 6809 was the original design choice, and some prototypes were built, but they quickly realised that it didn't have the power for a GUI based OS, and they used the Lisa's 68000, borrowing some of the Lisa low level functions (such as graphics toolkit routines) for the Macintosh.

Competing personal computers such as the Amiga and Atari ST, and early workstations by Sun, Apollo, NeXT and most others also used 680x0 CPUs (including one of the earliest workstations, the Tandy TRS-80 Model 16, which used a 68000 CPU and Z-80 for I/O and VM support).

Motorola:
    http://www.mot.com/
Motorola Microprocessors:
    http://www.mot.com/SPS/MMTG/mp.html
Absolute Mac US History:
    http://www.absolutemac.com/US/HISTOIRE/history.html
Amiga International:
    http://www.amiga.de/
Atari compatible Medusa computers:
    http://www.stud.ee.ethz.ch/~caschwan/medusa.html
Atari compatible C-Lab computers:
    http://www.ataricentral.com/st/c-lab/mkx.html

Intel 8086, IBM's choice (1978)

The Intel 8086 was based on the design of the 8080/8085 (source compatible with the 8080) with a similar register set, but was expanded to 16 bits. The Bus Interface Unit fed the instruction stream to the Execution Unit through a 6 byte prefetch queue, so fetch and execution were concurrent - a primitive form of pipelining (8086 instructions varied from 1 to 4 bytes).

It featured four 16 bit general registers, which could also be accessed as eight 8 bit registers, and four 16 bit index registers (including the stack pointer). The data registers were often used implicitly by instructions, complicating register allocation for temporary values. It featured 64K 8-bit I/O (or 32K 16-bit) ports and fixed vectored interrupts. There were also four segment registers that could be set from index registers.

The segment registers allowed the CPU to access 1 meg of memory through an odd process. Rather than just supplying missing bytes, as most segmented processors, the 8086 actually added the segment registers (X 16, or shifted left 4 bits) to the address. As a strange result of this unsuccessful attempt at extending the address space without adding address bits, it was possible to have two pointers with the same value point to two different memory locations, or two pointers with different values pointing to the same location, and limited typical data structures to less than 64K. Most people consider this a brain damaged design (a better method might have been that developed for the MIL-STD-1750 MMU).

Although this was largely acceptable for assembly language, where control of the segments was complete (it could even be useful then), in higher level languages it caused constant confusion (ex. near/far pointers). Even worse, this made expanding the address space to more than 1 meg difficult. The 80286 (1982?) expanded the design to 32 bits only by adding a new mode (switching from 'Real' to 'Protected' mode was supported, but switching back required using a bug in the original 80286, which then had to be preserved) which greatly increased the number of segments by using a 16 bit selector for a 'segment descriptor', which contained the location within a 24 bit address space, size (still less than 64K), and attributes (for Virtual Memory support) of a segment.

But all memory access was still restricted to 64K segments until the 80386 (1985), which included much improved addressing: base reg + index reg * scale (1, 2, 4 or 8 bits) + displacement (8 or 32 bit constant = 32 bit address (in the form of paged segments (using six 16-bit segment registers), like the IBM S/360 series, and unlike the Motorola 68030). It also had several processor modes (including separate paged and segmented modes) for compatibility with the previous awkward design. In fact, with the right assembler, code written for the 8008 can still be run on the most recent Pentium Pro. The 80386 also added an MMU, security modes (called "rings" of privledge - kernal, system services, application services, applications) and new op codes in a fashion similar to the Z-80 (and Z-280).

The 80486 (1989) added full pipelines, single on chip 8K cache, integrated FPU (based on the eight element 80-bit stack-oriented FPU in the 80387 FPU), and clock doubling versions (like the Z-280). The Pentium (late 1993) was superscalar (up to two instructions at once in dual integer units and single FPU) with separate 8K I/D caches.

The Pentium was the name Intel gave the 80586 version because it could not legally protect the name "586" to prevent other companies from using it - and in fact, the Pentium compatible CPU from NexGen is called the Nx586 (early 1995). Due to its popularity, the 80x86 line has been the most widely cloned processors, from the NEC V20/V30 (slightly faster clones of the 8088/8086 (could also run 8085 code)), AMD and Cyrix clones of the 80386 and 80486, to versions of the Pentium within less than two years of its introduction.

MMX (initially reported as MultiMedia eXtension, but later said by Intel to mean Matrix Math eXtension) is very similar to the earlier SPARC VIS or HP-PA MAX instructions) - they perform integer operations on vectors of 8, 16, or 32 bit words, using the 80 bit FPU stack elements as eight 64 bit registers (switching between FPU and MMX modes as needed - it’s very difficult to use them as a stack and as MMX registers at the same time). The P55C Pentium version (January 1997) is the first Intel CPU to include MMX instructions, followed by the AMD K6, and Pentium II. Cyrix also added these instructions in its M2 CPU (6x86MX, June 1997), as well as IDT with its C6.

Interestingly, the old architecture is such a barrier to improvements that most of the Pentium compatible CPUs (NexGen Nx586/Nx686, AMD K5, IDT-C6), and even the "Pentium Pro" (Pentium’s successor, late 1995) don’t clone the Pentium, but emulate it with specialized hardware decoders which convert Pentium instructions to RISC-like instructions which are executed on specially designed superscalar RISC-style cores faster than the Pentium itself. Intel also used BiCMOS in the Pentium and Pentium Pro to achieve clock rates competitive with CMOS load-store processors (the Pentium P55C (early 1997) version is a pure CMOS design).

Persistant rumours that IBM was developing hardware to translate Pentium instructions for the PowerPC as part of the PowerPC 615 CPU eventually died.

The Cyrix 6x86 (early 1996), initially manufactured by IBM before Cyrix merged with National Semiconductor, still directly executes 80x86 instructions in two pipelines, but partly out of order, making it faster than a Pentium. Cyrix also sells an integrated version with graphics and audio on-chip called the MediaGX. MMX instructions were added to the 6x86MX, and 3D graphics instructions to the 6x86MXi.

The Pentium Pro (code named "P6") is a 1 or 2-chip (CPU plus 256K or 512K L2 cache - I/D L1 cache (8K each) is on the CPU), 14-stage superpipelined processor. It uses extensive multiple branch prediction and speculative execution via register renaming. Three decoders (one for complex instructions, two for simpler ones (four or fewer micro-ops)) each decode one 80x86 instruction into micro-ops (one per simple decoder + up to four from the complex decoder = three to six per cycle). Up to five (usually three) micro-ops can be issued in parallel and out of order (six units - FPU, 2 integer, 2 address, 1 load/store), but are held and retired (results written to registers or memory) as a group to prevent an inconsistant state (equivalent to half an instruction being executed when an interrupt occurs, for example). 80x86 instructions may produce several micro-ops in CPUs like this (and the Nx586 and AMD K5), so the actual instruction rate is lower. In fact, due to problems handling instruction alignment in the Pentium Pro, emulated 16-bit instructions execute slower than on a Pentium. The Pentium II (April 1997) version of the Pentium Pro added MMX instructions, doubled cache to 32K, and was packaged in a processor card instead of an IC package.

The AMD K5 translates 80x86 code to ROPs (RISC OPerations), which execute on a RISC-style core based on the superscalar AMD 29K. Up to four ROPs can be dispatched to six units (two integer, one FPU, two load/store, one branch unit), and five can be retired at a time. The complexity led to low clock speeds for the K5, prompting AMD to buy NexGen and integrate its designs for the next generation K6.

The NexGen/AMD Nx586 (early 1995) is unique by being able to execute its micro-ops (called RISC86 code) directly, allowing optimised RISC86 programs to be written which are faster than an equivalent x86 program would be, but this feature is seldom used. It also features two 16K I/D L1 caches, a dedicated L2 cache bus (like that in the Pentium Pro 2-chip module) and an off-chip FPU (either separate chip, or later as in 2-chip module).

The Nx586 sucessor, the K6 (April 1997) actually has three caches - 32K each for data and instructions, and a half-size 16K cache containing instruction decode information. It also brings the FPU on-chip and eliminates the dedicated cache bus of the Nx586, allowing it to be pin-compatible with the P54C model Pentium. Another decoder is added (two complex decoders, compared to the Pentium Pro's one complex and two simple decoders) producing up to four micro-ops and issuing up to six (to seven units - load, store, complex/simple integer, FPU, branch, multimedia) and retiring four per cycle. It includes MMX instructions, licensed from Intel, and 3D graphics extensions are planned.

Centaur, a subsidiary of Integrated Device Technology, will produce the IDT-C6 (May 1997), which uses a much simpler (6-stage, 2 way integer/simple-FPU execution) desgn than Intel and AMD translation-based designs by using micro-ops more closely resembling 80x86 than RISC code, which allows for a higher clock rate and larger L1 (32K each I/D) and TLB caches in a lower cost, lower power consumption design. Simplifications include replacing branch prediction (less important with a short pipeline) with an eight entry call/return stack, depending more on caches. The FPU unit includes MMX support (the C6+ version added a second FPU/MMX unit and 3D graphics enhancements).

Intel, with partner Hewlett-Packard, has begun development of a next generation 64-bit processor (code named Merced or IA-64). It is expected to be a variable length instruction group (VLIG or what HP/Intel call EPIC (Explicit Parallel Instruction Computing)) with instruction dependencies grouped from 1 to 9+, in 128 bit bundles of three instructions plus three "template bits" which indicate dependancies. Most instructions are predicated, a design very similar to the TI 320C6x DSP, but with 128 general 64-bit and 128 floating point registers, and 64 predicate bits (a type of condition code). To reduce page faults, speculative load instructions execute a load, but does not trap if there is an exception until a second instruction completes it - if the second instruction is predicated and never executes, then a page fault is avoided, and loads can be rescheduled more flexibly. It's expectged to be compatible in some way with both the PA-RISC and 80x86 - it will include 80x86 data and segment registers, with additional instructions to switch between instruction/register sets and transfer data between 80x86 and IA-64 registers. It is expected to translate 80x86 instructions into VLIW instructions (or directly to decoded instructions) the same way that Pentium Pro and AMD K5/K6 CPUs do, but with a larger number of instructions issued using the VLIW design, it should be faster. However, native IA-64 code should be even faster, and this may finally produce the incentive to let the 80x86 architecture finally fade away.

So why did IBM chose the 8086 series for the IBM 5150 PC (1981) when most of the alternatives were so much better? Apparently IBM's own engineers wanted to use the 68000, and it was used later in the forgotten IBM Instruments 9000 Laboratory Computer, but IBM already had rights to manufacture the 8086, in exchange for giving Intel the rights to its bubble memory designs. Apparently IBM was using 8086s in the IBM Displaywriter word processor.

Other factors were the 8-bit 8088 (1979) version, which could use existing low cost 8085-type components, and allowed the computer to be based on a modified 8085 design. 68000 components were not widely available, though it could use 6800 components to an extent. After the failure and expense of the IBM 5100 (1974/5/6? - their first attempt at a peronal computer - discrete random logic CPU with no bus, built in BASIC and APL as the OS), cost was a large factor in the design of the PC.

The availability of CP/M-86 is also likely a factor, since CP/M was the operating system standard for the computer industry at the time. However Digital Research founder Gary Kildall was unhappy with the legal demands of IBM, so Microsoft, a programming language company, was hired instead to provide the operating system (initially known at varying times as QDOS, SCP-DOS, and finally 86-DOS, it was purchased by Microsoft from Seattle Computer Products and renamed MS-DOS).

Digital Research did eventually produce CP/M 68K for the 68000 series, making the operating system choice less relevant than other factors.

Intel bubble memory was on the market for a while, but faded away as better and cheaper memory technologies arrived.

Intel Corporation:
    http://www.intel.com/
Intel Product Info:
    http://www.intel.com/intel/product/index.htm
AMD, Inc.:
    http://www.amd.com/
AMD PC Processors Plus:
    http://www.amd.com/products/cpg/cpg.html
Cyrix:
    http://www.cyrix.com/
Centaur:
    http://www.centtech.com/
Hewlett-Packard:
    http://hpcc920.external.hp.com/
HP Microprocessors:
    http://hpcc920.external.hp.com/computing/framed/technology/micropro/
Interview with Bill Gates:
    http://innovate.si.edu/history/gates/gatestoc.htm

SPARC, an extreme windowed RISC (1987)

SPARC, or the Scalable (originally Sun) Processor ARChitecture was designed by Sun Microsystems for their own use. Sun was a maker of workstations, and used standard 68000-based CPUs and a standard operating system, Unix. Research versions of load-store processors had promised a major step forward in speed [See Appendix A], but existing manufacturers were slow to introduce a RISC processor, so Sun went ahead and developed its own (based on Berkeley's design). In keeping with their open philosophy, they licensed it to other companies, rather than manufacture it themselves.

SPARC was not the first RISC processor. The AMD 29000 (see below) came before it, as did the MIPS R2000 (based on Stanford's experimental design) and Hewlett-Packard PA-RISC CPU, among others.

The SPARC design was radical at the time, even omitting multiple cycle multiply and divide instructions (added in later versions), using single-cycle "step" instructions instead, while most RISC CPUs were more conventional.

SPARC usually contains about 128 or 144 registers, (memory-data designs typically had 16 or less). At each time 32 registers are available - 8 are global, the rest are allocated in a 'window' from a stack of registers. The window is moved 16 registers down the stack during a function call, so that the upper and lower 8 registers are shared between functions, to pass and return values, and 8 are local. The window is moved up on return, so registers are loaded or saved only at the top or bottom of the register stack. This allows functions to be called in as little as 1 cycle. Like most RISC processors, global register zero is wired to zero to simplify instructions, and SPARC is pipelined for performance (a new instruction can start execution before a previous one has finished), but not as deeply as others - like the MIPS CPUs, it has branch delay slots. Also like previous processors, a dedicated CCR holds comparison results.

SPARC is 'scalable' mainly because the register stack can be expanded (up to 512, or 32 windows), to reduce loads and saves between functions, or scaled down to reduce interrupt or context switch time, when the entire register set has to be saved. Function calls are usually much more frequent than interrupts, so the large register set is usually a plus, but compilers now can usually produce code which uses a fixed register set as efficiently as a windowed register set across function calls.

SPARC is not a chip, but a specification, and so there are various designs of it. It has undergone revisions, and now has multiply and divide instructions. Original versions were 32 bits, but 64 bit and superscalar versions were designed and implemented (beginning with the Texas Instruments SuperSparc in late 1992), but performance lagged behind other load-store and even Intel 80x86 processors until the UltraSPARC (late 1995) from Texas Instruments and Sun, and superscalar HAL/Fuji SPARC64 multichip CPU.

The UltraSPARC is a 64-bit superscalar processor series which can issue up to four instructions at once (but not out of order) to any of nine units: two integer units, two of the five floating point/graphics units, the branch and load/store unit. The UltraSparc also added a block move instruction which bypasses the caches (2-way 16K instr, 16K direct mapped data), to avoid disrupting it, and specialized pixel operations (VIS - the Visual Instruction Set) which can operate in parallel on 8, 16, or 32-bit integer values packed in a 64-bit floating point register (for example, four 8 X 16 -> 16 bit multiplications in a 64 bit word, a sort of simple SIMD/vector operation. More extensive than the Intel MMX instructions, or earlier HP PA-RISC MAX and Motorola 88110 graphics extensions, VIS also includes some 3D to 2D conversion, edge processing and pixes distance (for MPEG, pattern-matching support).

The HAL/Fuji SPARC64 can issue up to four in order instructions simultaneously to four buffers, then to four integer, two floating point, two load/store, and the branch unit, and may complete out of order (an instruction completes when it finishes without error, is committed when all instructions ahead of it have completed, and is retired when its resources are freed - these are 'invisible' stages in the SPARC64 pipeline). A combination of register renaming, a branch history table, and processor state storage (like in the Motorola 88K) allow for speculative execution while maintaining precise exceptions/interrupts (renamed integer, floating, and CC registers - trap levels are also renamed and can be entered speculatively).

Sun Microsystems:
http://www.sun.com/
Sun Microprocessor Products:
http://www.sun.com/microelectronics/products/microproc.html
HAL Computer Systems, Inc.:
http://www.hal.com/

AMD 29000, a flexible register set (1987)

The AMD 29000 is another load-store CPU descended from the Berkeley RISC design (and the IBM 801 project), as a modern successor to the earlier 2900 bitslice series (beginning around 1981). Like the SPARC design that was introduced shortly later, the 29000 has a large set of registers split into local and global sets. But though it was introduced before the SPARC, it has a more elegant method of register management.

The 29000 has 64 global registers, in comparison to the SPARC's eight. In addition, the 29000 allows variable sized windows allocated from the 128 register stack cache. The current window or stack frame is indicated by a stack pointer (a modern version of the ISAR register in the Fairchild F8 CPU), a pointer to the caller's frame is stored in the current frame, like in an ordinary stack (directly supporting stack languages like C, a CISC-like philosophy). Spills and fills occur only at the ends of the cache, and registers are saved/loaded from the memory stack. This allows variable window sizes, from 1 to 128 registers. This flexibility, plus the large set of global registers, makes register allocation easier than in SPARC (optimised stack operations also make it ideal for a stack-oriented interpreted languages such as PostScript, making it popular as a laser printer controller).

There is no special condition code register - any general register is used instead, allowing several condition codes to be retained, though this sometimes makes code more complex. An instruction prefetch buffer (using burst mode) ensures a steady instruction stream. Branches to another stream can cause a delay, so the first four new instructions are cached - next time a cached branch (up to sixteen) is taken, the cache supplies instructions during the initial memory access delay.

Registers aren't saved during interrupts, allowing the interrupt routine to determine whether the overhead is worthwhile. In addition, a form of register access control is provided. All registers can be protected, in blocks of 4, from access. These features make the 29000 useful for embedded applications, which is where most of these processors are used, allowing it at one point to claim the title of 'the most popular RISC processor'. The 29000 also includes an MMU and support for the 29027 FPU. The superscalar 29050 version in 1990 integrated a redesigned FPU (4 instructions could be dispatched to execute out of order and speculatively).

In late 1995 Advanced Micro Devices dropped development of the 29K in favour of its more profitable clones of Intel 80x86 processors, although much of the development of the superscalar core for a new AMD 29000 (including FPU designs from the 29050) was shared with the 'K5' (1995) Pentium compatible processor (the 'K5' translates 80x86 instructions to RISC-like instructions, and dispatches up to five at once to two integer units, one FPU, a branch and a load/store unit).

AMD, Inc.:
    http://www.amd.com/
AMD 29K Family:
    http://www.amd.com/products/lpd/29k/29kindex.html

MIPS R2000, the other approach. (June 1986)

The R2000 design came from the Stanford MIPS project, which stood for Microprocessor without Interlocked Pipeline Stages [See Appendix A], and was arguably the first commercial RISC processor (other candidates are the ARM and IBM ROMP used in the IBM PC/RT workstation, which was designed around 1981 but delayed until 1986). It was intended to simplify processor design by eliminating hardware interlocks between the five pipeline stages. This means that only single execution cycle instructions can access the thirty two 32 bit general registers, so that the compiler can schedule them to avoid conflicts. This also means that LOAD/STORE and branch instructions have a 1 cycle delay to account for. However, because of the importance of multiply and divide instructions, a special HI/LO pair of multiply/divide registers exist which do have hardware interlocks, since these take several cycles to execute and produce scheduling difficulties.

Like the AMD 29000 and DEC Alpha, the R2000 has no condition code register considering it a potential bottleneck. The PC is user readable. The CPU includes an MMU unit that can also control a cache, and the CPU was one of the first which could operate as a big or little endian processor. An FPU, the R2010, is also specified for the processor.

Newer versions included the R3000 (1988), with improved cache control, and the R4000 (1991) (expanded to 64 bits and is superpipelined (twice as many pipeline stages do less work at each stage, allowing a higher clock rate and twice as many instructions in the pipeline at once, at the expense of increased latency when the pipeline can't be filled, such as during a branch, (and requiring interlocks added between stages for compatibility, making the original "I" in the "MIPS" acronym meaningless))). The R4400 and above integrated the FPU with on-chip caches. The R4600 and later versions abandoned superpipelines.

The superscalar R8000 (1994) was optimised for floating point operation, issuing two integer or load/store operations (from four integer and two load/store units) and two floating point operations simultaneously (FP instructions sent to the independent R8010 floating point coprocessor (with its own set of thirty-two 64-bit registers and load/store queues)).

The R10000 and R12000 versions (early 1996 and May 1997) added multiple FPU units, as well as almost every advanced modern CPU feature, including separate 2-way I/D caches (32K each) plus on-chip secondary controller (and high speed 8-way split transaction bus (up to 8 transactions can be issued before the first completes)), superscalar execution (load four, dispatch five instructions (may be out of order) to any of two integer, two floating point, and one load/store units), dynamic register renaming (integer and floating point rename registers (thirty two in the R10K, fourty eight in the R12K)), and an instruction cache where instructions are partially decoded when loaded into the cache, simplifying the processor decode (and register rename/issue) stage. This technique was first implemented in the AT&T CRISP/Hobbit CPU, described later. Branch prediction and target caches are also included.

The 2-way (int/float) superscalar R5000 (January, 1996) was added to fill the gap between R4600 and R10000, without any fancy features (out of order or branch prediction buffers). For embedded applications, MIPS and LSI Logic added a compact 16 bit instruction set which can be mixed with the 32 bit set (same as the ARM Thumb 16 bit extension), implemented in a CPU called TinyRISC (October 1996), as well as MIPS V and MDMX (MIPS Digital Multimedia Extensions, announced October 1996)). MIPS V adds parallel floating point (two 32 bit fields in 64 bit registers) operations (compared to similar HP MAX integer extensions), MDMX adds integer 8 or 16 bit subwords in 64 bit FPU registers and a 24 and 48 bit subwords in a 192 bit accumulator for multimedia instructions (a MAC instruction on an 8-bit value can produce a 24-bit result, hence the large accumulator). Vector-scalar operations (ex: multiply all subwords in a register by subword 3 from another register) are also supported. These extensive instructions are partly derived from Cray vector instructions (Cray is owned by SGI, the parent company of MIPS). Future versions are expected to add Java virtual machine support.

Silicon Graphics MIPS Group:
    http://www.sgi.com/MIPS/

Hewlett-Packard PA-RISC, a conservative RISC (Oct 1986)

A design typical of many load-store processors, the PA-RISC (Precision Architecture, originally code-named Spectrum) was designed to replace older processors in HP-3000 MPE minicomputers, and Motorola 680x0 processors in the HP-9000 HP/UX Unix minicomputers and workstations. It has an unusually large instruction set for a RISC processor (including a conditional (predicated) skip instruction similar to those in the ARM processor), partly because initial design took place before RISC philosophy was popular, and partly because careful analysis showed that performance benefited from the instructions chosen - in fact, version 1.1 added new multiple operation instructions combined from frequent instruction sequences, and HP was among the first to add multimedia instructions (the MAX-1 and MAX-2 instructions, similar to Sun VIS or Intel MMX). Despite this, it’s a simple design - the entire original CPU had only 115,000 transistors, less than twice the much older 68000.

Much of the RISC philosophy was independently invented at HP from lessons learned from FOCUS (pre 1984), HP’s (and the world’s) first fully 32 bit microprocessor. It was a huge (at the time) 450,000 transistor chip with a stack based instruction set, described as "essentially a gigantic microcode ROM with a simple 32 bit data path bolted to its side". Performance wasn’t spectacular, but it was used in a pre-Unix workstation from HP.

It’s almost the cannonical load-store design, similar except in details to most other mainstream load-store processors like the Fairchild/Intergraph Clipper (1986), and the Motorola 88K in particular. It has a 5 stage pipeline, which (unlike early MIPS (R2000) processors) had hardware interlocks from the beginning for instructions which take more than one cycle, as well as result forwarding (a result can be used by a previous instruction without waiting for it to be stored in a register first).

It is a load/story architecture, originally with a single instruction/data bus, later expanded to a Harvard architecture (separate instruction and data buses). It has thirty-two 32-bit integer registers (GR0 wired to constant 0, GR31 used as a link register for procedure calls), with seven ’shadow registers’ which preserve the contents of a subset of the GR set during fast interrupts (also like ARM), and thirty-two 64-bit floating point registers (also as sixty-four 32-bit and sixteen 128-bit), in an FPU (which could execute a floating point instruction simultaneously, from the Apollo-designed Prism architecture (1988?) after Hewlett-Packard acquired the company). Later versions (the PA-RISC 7200 in 1994) added a second integer unit (still dispatching only two instructions at a time to any of the three units). Addressing originally was 48 bits, and expanded to 64 bits, using a segmented addressing scheme.

The PA-RISC 7200 also included a tightly integrated cache and MMU, a high speed 64-bit ’Runway’ bus, and a fast but complex fully associative 2KB on-chip assist cache, between the simpler direct-mapped data cache and main memory, which reduces thrashing (repeatedly loading the same cache line) when two memory addresses are aliased (mapped to the same cache line). Instructions are predecoded into a separate instruction cache (like the AT&T CRISP/Hobbit).

The PA-RISC 8000 (April 1996), intended to compete with the R10000, UltraSparc, and others) expands the registers and architecture to 64 bits (eliminating the need for segments), and adds aggressive superscalar design - up to 5 instructions out of order, using fifty six rename registers, to ten units (five pairs of: ALU, shift/merge, FPU mult/add, divide/sqrt, load/store). The CPU is split in two, with load/store (high latency) instructions dispatched from a separate queue from operations (except for branch or read/modify/write instructions, which are copied to both queues). It also has a deep pipeline and speculative execution of branches (many of the same features as the R10000, in a very elegant implementation).

The PA-RISC 8500 breaks with HP tradition (in a big way) and adds on-chip cache - 1.5Mb L1 cache.

Although typically sporting fewer of the advanced (and promised) features of competing CPUs designs, a simple elegant design and effective instruction set has kept PA-RISC performance among the best in its class (of those actually available at the same time) since its introduction.

HP pioneered the addition of multimedia instructions with the MAX-1 (Multimedia Acceleration eXtension) extensions in the PA-7100LC (pre-1994) and 64-bit (version 2.0) MAX-2 extensions in the PA-8000, which allowed vector operations on two or four 16-bit subwords in 32-bit or 64-bit integer registers (this only required circuitry to slice the integer ALU (similar to bit-slice processors, such as the AMD 2901), adding only 0.1 percent to the PA-8000 CPU area - using the FPU registers like Sun’s VIS and Intels MMX do would have required duplicating ALU functions. 8 and 32-bit support, multiplication, and complex instructions were also left out in favour of powerful ’mix’ and ’permute’ packing/unpacking operations).

In the future Hewlett-Packard plans to pursue a "post-VLIW" (Very Long Instruction Word) design in conjunction with Intel code named "Merced" or IA-64, possibly expanding on the idea of MAX or MMX operations. Some of the newer CPUs which execute Intel 80x86 instructions (The AMD ’K5’ and NexGen Nx586, for example) treat 80x86 instructions as VLIW instructions, decoding them into RISC-like instructions and executing several concurrently. However, it would more likely follow the VLIW design of the TI 320C6x DSP.

Hewlett-Packard:

http://hpcc920.external.hp.com/
HP Microprocessors:
http://hpcc920.external.hp.com/computing/framed/technology/micropro/

Acorn ARM, RISC for the masses (1986)

ARM (Advanced RISC Machine, originally Acorn RISC Machine) is often praised as one of the most elegant modern processors in existence. It was meant to be "MIPs for the masses", and designed as part of a family of chips (ARM - CPU, MEMC - MMU and DRAM/ROM controller, VIDC - video and DAC, IOC - I/O, timing, interrupts, etc), for the Archimedes home computer (multitasking OS, windows, etc). It's made by VLSI Technologies Inc, and based partly on the Berkeley experimental load-store design. It is simple, has a short 3-stage pipeline, and it can operate in big- or little-endian mode.

The original ARM (ARM1, 2 and 3) was a 32 bit CPU, but used 26 bit addressing. The newer ARM6xx spec is completely 32 bits. It has user, supervisor, and various interrupt modes (including 26 bit modes for ARM2 compatibility). The ARM architecture has sixteen registers (including user visible PC as R15) with a multiple load/save instruction, though many registers are shadowed in interrupt modes (2 in supervisor and IRQ, 7 in FIRQ) so need not be saved, for fast response. The instruction set is reminiscent of the 6502, used in Acorns earlier computers.

A feature introduced by the ARM is that every instruction is predicated, using a 4 bit condition code (including 'never execute', not officially recommended), an idea later used in some HP PA-RISC instructions and the TI 320C6x DSP. Another bit indicates whether the instruction should set condition codes, so intervening instructions don't change them. This easily eliminates many branches and can speed execution. Another unique and useful feature is a barrel shifter which operates on the second operand of most ALU operations, allowing shifts to be combined with most operations (and index registers for addressing), effectively combining two or more instructions into one.

These features make ARM code both dense (unlike most load-store processors) and efficient, despite the relatively low clock rate and short pipeline - it is roughly equivalent to a much more complex 80486 in speed. And like the Motorola Coldfire, ARM has developed a low cost 16-bit version called Thumb, which recodes a subset of ARM CPU instructions into 16 bits (decoded to native 32-bit ARM instructions without penalty - similar to the CISC decoders in the newest 80x86 compatible and 68060 processors, except they decode native instructions into a newer one, while Thumb does the reverse). Thumb programs can be 30-40% smaller than already dense ARM programs. Native ARM code can be mixed with Thumb code when the full instruction set is needed.

The ARM series consists of the ARM6 CPU core (35,000 transistors, which can be used as the basis for a custom CPU) the ARM60 base CPU, and the ARM600 which also includes 4K 64-way set-associative cache, MMU, write buffer, and coprocessor interface (for FPU). A newer version, the ARM7 series (Dec 1994), increases performance by optimising the multiplier, and adding DSP extensions including 32 bit and 64 bit multiply and multiply/accumulate instructions (operand data paths lead from registers through the multiplier, then the shifter (one operand), and then to the integer ALU for up to three independent operations). It also doubles cache size to 8K, includes embedded In Circuit Emulator (ICE) support, and raises the clock rate significantly.

A full DSP coprocessor (codenamed Piccolo, expected second half 1997) adds an independent set of sixteen 32-bit registers (also accessable as thirty two 16 bit registers), four which can be used as 48 bit registers, and a complete DSP instruction set (including four level zero-overhead loop operations), using a load-store model similar to the ARM itself. The coprocessor has its own program counter, interacting with the CPU which performs data load/store through input/output buffers connected to the coprocessor bus (similar but more intelligent than the address unit in a typical DSP (such as the Motorola 56K) supporting the data unit). The coprocessor shares the main ARM bus, but uses a separate instruction buffer to reduce conflict. Two 16 bit values packed in 32 bit registers can be computed in parallel, similar to the HP PA-RISC MAX-1 multimedia instructions.

The ARM CPU was chosen for the Apple Newton handheld system because of its speed, combined with the low power consumption, low cost and customizable design (the ARM610 version used by Apple includes a custom MMU supporting object oriented protection and access to memory for the Newton's NewtOS). DEC has also licensed the architecture, and has developed the SA-110 (StrongARM) (February 1996), running a 5-stage pipeline at 100 to 233MHz (using only 1 watt of power), with 5-port register file, faster multiplier, single cycle shift-add, and Harvard architecture (16K each 32-way I/D caches). To fill the gap between ARM7 and DEC StrongARM, ARM also developed the ARM8/800 which includes many StrongARM features, and the ARM9 with Harvard busses, write buffers, and flexible memory protection mapping.

An experimental asynchronous version of the ARM6 (operates without an external or internal clock signal) called AMULET has been produced by Steve Furber's research group at Manchester university. The first version (AMULET1, early 1993) is about 70% the speed of a 20MHz ARM6 on average (using the same fabrication process), but simple operations (multiplication is a big win at up to 3 times the speed) are faster (since they don't need to wait for a clock signal to complete). AMULET2e (October 1996, 93K transistor AMULET2 core plus four 1K fully associative cache blocks) is 30% faster (40 MIPS, 1/2 the performance of a 75MHz ARM810 using same fabrication), uses less power, and includes features such as branch prediction. AMULET3 is expected to be a commercial product in 1999.

Advanced RISC Machines:
    http://www.arm.com/
Digital Electronics Corporation:
    http://www.digital.com/
Newton, Inc.:
    http://www.newton.apple.com/

Intel 960, Intel quietly gets it right (1987 or 1988?)

Largely obscured by the marketing hype surrounding the Intel 80860, the 80960 was actually an overall better processor, and replaced the AMD 29K series as "the world's most popular embedded RISC" until 1996. The 960 was aimed for the high end embedded market (it included multiprocessor and debugging support, and strong interrupt/fault handling, but lacked MMU support), while the 860 was intended to be a general purpose processor (the name 80860 echoing the popular 8086).

Although the first implementation was not superscalar, the 960 was designed to allow dispatching of instructions to multiple (undefined, but generally including at least one integer) execution units, which could include internal registers (such as the four 80 bit registers in the floating point unit (32, 64, and 80 bit IEEE operations)) - the 960 CA version (1989) was superscalar. There are sixteen 32 bit global registers which can be shared by all excution units and sixteen register "caches" - similar to the SPARC register windows, but not overlapping (originally four banks). It's a load/store Harvard architecture (32-bit flat addressing), but has some complex microcoded instructions (such as CALL/RET). There are also thirty-two 32 bit special function registers.

It's a very clean embedded architecture, not designed for high level applications, but very effective and scalable - something that can't be said for all Intel's processor designs.

Intel Corporation:
 http://www.intel.com/
Intel i960 Processor Info:
 http://developer.intel.com/design/i960/INDEX.HTM

Intel 860, "Cray on a Chip" (late 1988?)

The Intel 80860 was an impressive chip, able at top speed to perform close to 66 MFLOPS at 33 MHz in real applications, compared to a more typical 5 or 10 MFLOPS for other CPUs of the time. Much of this was marketing hype, and it never become popular, lagging behind most newer CPUs and Digital Signal Processors in performance.

The 860 has several modes, from regular scaler mode to a superscalar mode that executes two instructions per cycle and a user visible pipeline mode (instructions using the result register of a multi-cycle op would take the current value instead of stalling and waiting for the result). It can use the 8K data cache in a limited way as a small vector register (like those in supercomputers). The unusual cache uses virtual addresses, instead of physical, so the cache has to be flushed any time the page tables changes, even if the data is unchanged. Instruction and data busses are separate, with 4 G of memory, using segments. It also includes a Memory Management Unit for virtual storage.

The 860 has thirty two 32 bit registers and thirty two 32 bit (or sixteen 64 bit) floating point registers. It was one of the first microprocessors to contains not only an FPU as well as an integer ALU, and also included a 3-D graphics unit (attached to the FPU) that supports lines drawing, Gouraud shading, Z-buffering for hidden line removal, and operations in conjunction with the FPU. It was also the first able to do an integer operation, and a (unique at the time) multiply and add floating point instruction, for the equivalent of three instructions, at the same time.

However actually getting the chip at top speed usually requires using assembly language - using standard compilers gives it a speed closer to other processors. Because of this, it was used as a coprocessor, either for graphics, or floating point acceleration, like add in parallel units for workstations. Another problem with using the Intel 860 as a general purpose CPU is the difficulty handling interrupts. It is extensively pipelined, having as many as four pipes operating at once, and when an interrupt occurs, the pipes can spill and lose data unless complex code is used to clean up. Delays range from 62 cycles (best case) to 50 microseconds (almost 2000 cycles).

IBM RS/6000 POWER chips (1990)

When IBM decided to become a real part of the workstation market (after its unsuccessful PC/RT based on the ROMP processor), it decided to produce a new innovative CPU, based partly on the 801 project that pioneered RISC theory. RISC normally stands for Reduced Instruction Set Computer, but IBM calls it Reduced Instruction Set Cycles, and implemented a relatively complex processor with more high level instructions than most memory-data processors. They ended up with was a CPU (POWER1) that initially contained five or seven separate chips - the branch unit, fixed point unit, floating point unit, and either two or four cache chips (separate data and instruction cache). Some PowerPC versions (co-developed by IBM, Apple, and Motorola as a successor to both the Motorola 68000 and Intel 80x86) have a unified on-chip cache (32K in the 601), newer versions have split I/D caches. PowerPC versions also have a simplified instruction set, emulating the older instructions if necessary.

The branch unit is the heart of the CPU, and enables multiple instructions (up to four in the original POWER1, more commonly two or three) to be executed at once. It contains the condition code register, a loop register (can decrement and branch on zero with no penalty - a feature usually only found on DSPs like the TMS320C30), and performs branches. The condition code register has eight fields - in POWER1 two were reserved for the fixed and floating point units, the other six could be set separately (or combined from several instructions), and can be checked several instructions later. It also dispatches multiple instructions (out of order if possible) to available execution units (each unit has a buffer allowing instructions to be dispatched to a unit still executing a complex instruction).

The branch unit can speculatively take branches (using a prediction bit in the POWER1 and PowerPC 601 (1993), and using dynamic prediction and a Branch History Table in the PowerPC 604 (mid 1995) and newer versions), dispatching instructions and then canceling them if the branch is not taken (3 cycle maximum penalty) while buffering the other instruction path to reduce latency. The branch unit also manages procedure calls and returns on a program counter stack, allowing effective zero-cycle calls when overlapped with other instructions. Finally, it handles interrupts (except floating point exceptions) without software intervention.

The integer unit(s) perform integer operations, as well as some complex string instructions in the POWER1 and 2 and 64-bit PowerPC-AS (AS/400, mid 1996), and loads and stores in the POWER1 and PowerPC 601 (newer versions added a separate concurrent load/store unit) - including 'update' forms of load/stores (similar to post-increment/decrement memory-data addressing modes). Most versions contain thirty two 32 bit registers (the POWER1/2 design included a special MQ register for extended precision integer multiply/divides (like the MIPS HI/LO registers), but it was removed in PowerPC CPUs after the 601 as a potential bottleneck), while the PowerPC 620 (delivered late, April 1996, then withdrawn for redesign) and AS registers are 64 bits (with appropriate new instructions). R0 is sometimes treated as a constant 0, like other load-store processors, but can also contain a value for other operations. The high end PowerPC-AS, intended for the AS/400 minicomputer series, also has decimal arithmetic and string instructions (but only for 64-bit instructions), and an interface for a matrix coprocessor (for possible future RS/6000 workstations). All integer units can forward results needed by subsequent instructions before the write stage occurs, and some versions (PowerPC 603 and later) include extra registers which are renamed for speculative or out of order instruction execution to prevent write conflicts, and make it easier to discard the results of a canceled instruction. A reorder buffer in the branch/dispatch unit tracks renamed integer and floating point registers.

The floating point unit contains thirty two 64 bit registers and performs all typical floating point operations (single or double precision in PowerPC, double only in POWER1 and POWER2), including DSP-like multiply/accumulate instructions and array multiply and add. The registers are loaded and stored by the fixed point unit in POWER1 and PowerPC 601, by the Load/Store unit in others (the multichip POWER2 has two dedicated floating point load/store units). The FPU also includes rename registers. Like some other CPUs, floating point traps are imprecise because of pipelining - normally, a trap bit is set on a floating point exception, and software can test for the condition to generate a trap - or ignore it if its a safe operation. For debugging, a slower precise trap mode is included.

Data buses range from 32 bits for early and low end versions to 256 bits (plus ECC bits) for the high bandwidth POWER2 multichip CPU (late 1993 - 8 chips, 23 million transistors including 256K cache, later combined into one chip, and then into 8 CPUs+cache on one chip) which issues up to six instructions and four simultaneous loads or stores (the POWER2 was later superceded by the lower cost PowerPC based POWER3). The PowerPC 601 used the Motorola 88000 microprocessor bus, more recent versions use PowerPC specific buses, some with a 128 bit 'backside' bus (620 and later versions) used to access a L2 cache.

Very high clock rate (500MHz) BiCMOS 704 (based on a simplified 604 with only one integer, one FPU, one load/store unit) was being developed (expected early 1997) by Exponential Technology, expanding on the type of technology Intel found necessary to keep its Pentium and Pentium Pro CPUs competitive, but advances in CMOS and a slower initial product (410MHz) sharply reduced the clock speed advantages, cancelling the project (faster, lower power, fully CMOS Pentium and Pentium Pro CPUs have replaced earlier BiCMOS versions). Embedded versions have also been introduced by IBM (40x series) and Motorola (8xx and 50x series - ironically, Motorola now has a MPC860 CPU (on-chip communications support) with a viable future, in contrast to Intel's i860).

In direct response to Intel's MMX instructions, VMX (Video and Multimedia eXtension) are planned to be added to the fourth (G4) and possibly some third (G3) generation PowerPC CPUs. This consists of extending FPU registers to 128 bits, and adding SPARC-type VIS vector/multimedia operations (8, 16, or 32 bit subwords).

Overall the IBM POWER/PowerPC CPU is very powerful, reminiscent of mainframe designs, which almost qualifies it as "Weird and Innovative", and violates the RISC philosophy of simplicity and fewer instructions (at over a hundred (including identical pairs where one implicitly sets the CC registers and the other doesn't), versus only about 34 for the ARM and 52 for the Motorola 88000 (including FPU instructions)). The high complexity is very effective, but also limits the clock rate of the designs - an interesting tradeoff considering that a highly parallel 71.5 MHz POWER2 is faster than a 200MHz DEC Alpha 21064.

IBM:

    http://www.ibm.com/

IBM PowerPC Microprocessor:

http://www.chips.ibm.com/products/ppc/
Motorola:
http://www.mot.com/
Motorola Microprocessors:
http://www.mot.com/SPS/MMTG/mp.html
Apple Computer:
http://www.apple.com/

DEC Alpha, Designed for the future (1992)

The DEC Alpha architecture is designed, according to DEC, for a operational life of 25 years. Its main innovation is PALcalls (or writable instruction set extension), but it is an elegant blend of features, selected to ensure no obvious limits to future performance - no special registers, etc. The first Alpha chip is the 21064.

Alpha is a 64 bit architecture (32 bit instructions) that doesn't support 8- or 16-bit operations, but allows conversions, so no functionality is lost (Most processors of this generation are similar, but have instructions with implicit conversions). Alpha 32-bit operations differ from 64 bit only in overflow detection. Alpha does not provide a divide instruction due to difficulty in pipelining it. It's very much like the MIPS R2000, including use of general registers to hold condition codes. However, Alpha has an interlocked pipeline, so no special multiply/divide registers are needed, and Alpha is meant to avoid the significant growth in complexity which the R2000 family experienced as it evolved into the R8000 and R10000.

One of Alpha's roles is to replace DEC's two prior architectures - the MIPS-based workstations and VAX minicomputers (Alpha evolved from a VAX replacment project codenamed PRISM, not to be confused with the Apollo Prism acquired by Hewlett Packard). To do this, the chip provides both IEEE and VAX 32 and 64 bit floating point operations, and features Privileged Architecture Library (PAL) calls, a set of programmable (non-interruptable) macros written in the Alpha instruction set, similar to the programmable microcode of the Western Digital MCP-1600 or the AMD Am2910 CPUs, to simplify conversion from other instruction sets using a binary translator, as well as providing flexible support for a variety of operating systems.

Alpha was also designed for the future for a 1000-fold eventual increase in performance (10 X by clock rate, 10 X by superscalar execution, and 10 X by multiprocessing) Because of this, superscalar instructions may be reordered, and trap conditions are imprecise (like in the 88100). Special instructions (memory and trap barriers) are available to syncronise both occurrences when needed (different from the POWER use of a trap condition bit which is explicitly by software, but similar in effect. SPARC also has a specification for similar barrier instructions). And there are no branch delay slots like in the R2000, since they produce scheduling problems in superscalar execution, and compatibility problems with extended pipelines. Instead speculative execution (branch instructions include hint bits) and a branch cache are used.

The 21064 was introduced with one integer, one floating point, and one load/store unit. The 21164 (Early 1995) added one integer/load/store unit with byte vector (multimedia-type) instructions (replacing the load/store unit) and one floating point unit, and increased clock speed from 200 MHz to 300 MHz (still roughly twice that of competing CPUs), and introduced the idea of a level 2 cache on chip (8K each inst/data level 1, 96K combined level 2). The 21264 (early 1997) expanded this to four integer units (two add/logic/shift/branch (one also with multiply, one with multimedia) and two add/logic/load/store), two different floating point units (one for add/div/square root and for multiply), with the ability to load four, dispatch six, and retire eight instructions per cycle (and for the first time including 40 integer and 40 floating point rename registers and out of order execution), at up to 500MHz.

Multimedia extensions introduced with the 21264 are simple, but include VIS-type motion estimation (MPEG).

DEC's Alpha is in many ways the antithesis of IBM's POWER design, which gains performance from complexity, and the expense of a large transistor count, while the Alpha concentrates on the original RISC idea of simplicity and a higher clock rate - though that also has its drawback, in terms of very high power consumption.

Digital Electronics Corporation:
http://www.digital.com/

Intel 432, Extraordinary complexity (1980)

The Intel iAPX 432 was a complex, object oriented 32-bit processor that included high level operating system support in hardware, such as process scheduling and interprocess messaging. It was intended to be the main Intel microprocessor (some said the 80286 was envisioned as a step between the 8086 and the 432, others claim the 8086 was to be the bridge to the 432, rushed through design when the 432 was late and resulting in its many design problems). The 432 actually included four chips. The GDP (processor) and IP (I/O controller) were introduced in 1980, and the BIU (Bus Interface Unit) and MCU (Memory Control Unit) were introduced in 1983 (but not widely). The GDP complexity was split into 2 chips (decode/sequencer and execution units, like the Western Digital MCP-1600), so it wasn't really a microprocessor.

The GDP was exclusively object oriented - normal linear memory access wasn't allowed, and there was hardware support for data hiding, methods, inheritance, late binding, and access protection, and it was promoted as being ideal for the Ada programming language. To enforce this, permission and type checks for every memory access (via a 2 stage segmentation) slowed execution (despite cached segment tables). It supported up to 2^24 segments, each limited to 64K in size (within a 2^32 address space), but the object oriented nature of the design meant that was not a real limitation. The stack oriented design meant the GDP had no user data registers. Instructions were bit encoded (and bit-aligned in memory), ranging from 6 bits to 321 bits long (the T-9000 has variable length byte encoded/aligned instructions) and could be very complex.

The BIU defined the bus, designed for multiprocessor support allowing up to 63 modules (BIU or MCU) on a bus and up to 8 independent buses (allowing memory interleaving to speed access). The MCU did automatic parity checking and ECC error correcting. The total system was designed to be fault tolerant to a large degree, and each of these parts contributes to that reliability.

Despite these advanced features, the 432 didn't catch on. The main reason was that it was slow, sometimes up to five or ten times slower than a 68000 or Intel's own 80286. Part of this was the lack of local (user) data registers, or a data cache. Part of this was the fault-tolerant BIU, which defined an (asynchronous protocol) clocked bus that resulted in 25% to 40% of the access time being used by wait states. The instructions weren't aligned on bytes or words, and took longer to decode. In addition, the protections imposed on the objects slowed data access. Finally, the implementation of the GDP on two chips instead of one produced a slower product. However, the fact that this complex design was produced and bug free is impressive.

Its high level architecture was similar to the Transputer systems, but it was implemented in a way that was much slower than other processors, while the T-414 wasn't just innovative, but much faster than other processors of the time.

The Intel i960 is sometimes considered a successor of the 432 (also called "RISC applied to the 432"), and does have similar hardware support for context switching. This path came about indirectly through the 960 MC designed for the BiiN machine, which was still very complex (it included many i432 object-oriented ideas, including a tagged memory system).

T-9000, parallel computing (1994)

The INMOS T-9000 is the latest version of the Transputer architecture, a processor designed to be hooked up to other processors for parallel processing. The previous versions were the 16 bit T-212 and 32 bit T-414 and T-800 (which included a 64 bit FPU) processors (1983 and 1985). The instruction set is minimised, like a RISC design, but is based on a stack/accumulator design (similar in idea to the PDP-8), and designed around the OCCAM language. The most important feature is that each chip contains 4 serial links to connect the chips in a network.

While the transputers were originally faster than their contemporaries, recent load-store designs have surpassed them. The T-9000 was an attempt to regain the lead. It starts with the architecture of the T-800 which contains only three 32 bit integer and three 64 bit floating point registers which are used as an evaluation stack - they are not general purpose. Instead, like the TMS 9900, it uses memory, addressed relative to the workspace register (the 9900 workspace contained only sixteen registers, the Transputer workspace can be any length, though access slows down with every 4 bits used for offset from the workspace register - sixteen bytes can be accessed with just one instruction, 256 needs two, and so on). This allows very fast context switching, less than a microsecond, speeding and simplifying process scheduling enough that it is automated in hardware (supporting two priority levels and event handling (link messages and interrupts)). The Intel 432 also attempted some hardware process scheduling, but was unsuccessful.

Unlike the TMS 9900, the T-9000 is far faster than memory, so the CPU has several levels of high speed caches and memory types. The main cache is 16K, and is designed for 3 reads and 1 write simultaneously. The workspace cache is based on 32 word rotating buffers, allows 2 reads and 1 write simultaneously.

Instructions are in bytes, consisting of 4 bit op code and 4 bit data (usually a 16 byte offset into the workspace), but prefix instructions can load extra data for an instruction which follows, 4 bits at a time. Less frequent instructions can be encoded with 2 (such as process start, message I/O) or more bytes (CRC calculations, floating point operations, 2D block copies and scheduler queue management). The stack architecture makes instructions very compact, but executing one instruction byte per clock can be slow for multibyte instructions, so the T-9000 has a grouper which gathers instruction bytes (up to eight) into a single CISC-type instruction then sent into the 5 stage pipeline (fetching four per cycle, grouping up to 8 if slow earlier instructions allow it to catch up). For example, two concurrent memory loads (simple or indexed), a stack/ALU operation and a store (a[i] = b[2] + c[3]) can be grouped.

The T-9000 contains 4 main internal units, the CPU, the VCP (handling the individual links of the previous chips, which needed software for communication), the PMI, which manages memory, and the Scheduler.

This processor is ideal for a model of parallel processing known as systolic arrays (a pipeline is a simple example). Even larger networks can be created with the C104 crossbar switch, which can connect 32 transputers or other C104 switches into a network hundreds of thousands of processors large. The C104 acts like a instant switch, not a network node, so the message is passed through, not stored. Communication can be at close to the speed of direct memory access.

Like the many CPUs, the Transputers can adapt to a 64, 32, 16, or 8 bit bus. They can also feed off a 5 MHz clock, generating their own internal clock (up to 50MHz for the T-9000) from this signal, and contain internal RAM, making them good for high performance embedded applications.

Unfortunately excessive delays in the T-9000 design (partly because of the stack based design) left it uncompetitive with other CPUs (roughly 36 MIPS at 50 MHz). The T-4xx and T-8xx architecture still exist in the SGS-Thomson ST20 microcore family.

As a note, the T-800 FPU is probably the first large scale commercial device to be proven correct through formal design methods.

SGS-Thomson Microelectronics:
    http://www.st.com/
SGS-Thomson Products Contents:
    http://www.st.com/stonline/books/index.htm

Sun picoJava - not another language-specific processor! (October 1997)

Sun first introduced Java as a combination of language, integrated classes, and a run time system called the Java Virtual Machine (JVM). To support Java, Sun Microelectronics designed picoJava and microJava hardware to execute Java bytecode programs faster than a virtual machine or recompiled code.

The picoJava I (early 1997) is a stack oriented CPU core like the JVM, with a 64 entry stack cache (similar to the Patriot Scientific ShBoom PSC1000), but there are interesting differences between it and FORTH-style stack CPUs. Java only uses a single stack (like many languages such as C, which the AT&T Hobbit and AMD 29K were designed to support) and the picoJava CPU enhances performance with a 'dribbler' unit which constantly updates a complete copy of the stack cache in memory, without affecting other CPU operations (similar to a write-back cache), so stack frames can be added without waiting for a stack frame to be stored. Some Java instructions are complex, so the CPU has microcoded instructions, and a 4 stage pipeline (fetch, decode, execute/cache, stack writeback). Finally, picoJava groups (or 'folds') load and stack operations together, executing both at once (treating the top of stack as an accumulator) (this is a much simpler version of instruction grouping tried in the Transputer T-9000), This usually eliminates 60% of stack operation inefficiency. Seldom used instructions aren't implemented, but are emulated using trap handlers.

The picoJava II (October 1997) core is used in the first actual CPU from Sun, the microJava701. It extends the pipeline to 6 stages, and can four up to four instructions into one operation. It also adds a FPU and separate 16Kb I/D caches.

Sun Microsystems:
    http://www.sun.com/
Sun picoJava Core Architecture:
    http://www.sun.com/microelectronics/whitepapers/wpr-0014-01/

IBM System 360/370/390: The Mainframe(1964)

The IBM System/360 is a sort of geologic feature in the computer world, and isn't at all a microprocessor, but was certainly influential (and enough people asked for it to be included in this list). It was designed to be an "all around" (as in, 360 degrees) system usable for any computing task, and as a result created many of the standards for the computing industry, such as 8-bit bytes and byte addressable memory, 32-bit words, segmented and paged memory (see the DEC VAX, except that in the S/360 the PSW includes the program counter (24 bits in the S/360, 31 bits in the S/370 XA (eXtended Architecture, pre 1983) and later versions). The S/370 (pre 1977) also includes sixteen control registers used by the operating system.

A two stage pipeline was first introduced in the IBM 3033 (1977). Instructions are fetched from the cache into three 32 bit buffers. The Instruction Pre-Processing Function (IPPF) then decodes them, generates operand addresses and stores them in operand address registers, and places source operands in operand buffers. Decoded instructions were placed into a 4 entry queue until the execution unit was ready.

In some models (such as 360/91) when a conditional branch occurs, the most likely next instruction is loaded into the IPPF buffer, but the previous next instruction is not discarded, so either can be executed without penalty. Two speculative branches can be buffered this way. Some had a "loop mode" like the Motorola 68010.

Addressing was originally 24 bit, but was extended to 31 bits (the high bit indicated whether to use 24 or

32 bits) with the XA architecture (This caused problems with software which stored type information in the unused 8 bits of a 32 bit word. The same thing happened when the Motorola 68000 was expanded from 24 to 32 bit addressing). The S/360 used completely position independent (register+offset and register+index) addressing modes. Virtual memory was added in the S/370, and used a segment and paging method - the first 8 bits of an address indicated an entry in a segment table which is added to the next 4 or 8 bits to get the page table index which contains the upper (12 or 20) bits of the physical memory address, and the rest of the address provides the lower 12 bits (the Intel 80386 uses a similar method, while the Motorola 68030 uses fixed length logical/physical pages instead of variable length segments).

Like the DEC VAX, the S/370 has been implemented as a microprocessor. The Micro/370 discarded all but 102 instructions (some supervisor instructions differed), with a coprocessor providing support for 60 others, while the rest are emulated (as in the MicroVAX). The Micro/370 had a 68000 compatible bus, but was otherwise completely unique (some legends claim it was a 68000 with modified microcode plus a modified 8087 as the coprocessor, others say IBM started with the 68000 design and completely replaced most of the core, keeping the bus interface, ALU, and other reusable parts, which is more likely).

More recently, with increased microprocessor complexity, a complete S/390 superscalar microprocessor with 64K L1 cache (at up to 350MHz, a higher clock rate than the 200MHz Intel's Pentium Pro available at the time) has been designed.

IBM:
    http://www.ibm.com/
IBM System/390:
    http://www.s390.ibm.com/

VAX: The Penultimate CISC (1978)

The VAX architecture wasn't designed as a microprocessor, though single chip versions were implemented (around 1984). However, it and its predecessor, the PDP-11, helped inspire design of the Motorola 68000, Zilog Z8000, and particularly the National Semiconductor 32xxx series CPUs. It was considered the most advanced CISC design, and the closest so far to the ultimate CISC goal. This is one reason that the VAX 11/780 is used as the speed benchmark for 1 MIPS (Million Instructions Per Second), though actual execution was apparently closer to 0.5 MIPS.

The VAX was a 32 bit architecture, with a 32 bit address range (split into 1G sections for process space, process specific system space, system space, and unused/reserved for future use). Each process has its own 1G process and 1G process system address space, with memory allocated in pages.

It features sixteen user visible 32 bit registers. Registers 12 to 15 are special - AP (Argument Pointer), FP (Frame Pointer), SP and PC (user, supervisor, executive, and kernal modes have separate SPs in R14, like the 68000 user and supervisor modes). All these registers can be used for data, addressing and indexing. A 64 bit PSL (Program Status Longword) keeps track of interrupt levels, program status, condition codes, and access mode (kernal (hardware management), executive (files/records), supervisor (interpreters), user (programs/data)).

The VAX 11 featured an 8 byte instruction prefetch buffer, like the 8086, while the VAX 8600 has a full 6 stage pipeline. Instructions mimic high level language constructs, and provide dense code. For example, the CALL instruction, which not only handles the argument list itself, but enforces a standard procedure call for all compilers. However, the complex instructions aren't always the fastest way of doing things. For example, the INDEX instruction was 45% to 60% faster when by replaced by simpler VAX instructions. This was one inspiration for the RISC philosophy.

Further inspiration came from the MicroVAX (VAX 78032) implementation, since in order to reduce the architecture to a single (integer) chip, only 175 of the 304 instructions (and 6 of 14 native data types) were implemented (through microcode), while the rest were emulated - this subset included 98% of instructions in a typical program. The optional FPU implemented 70 instructions and 3 VAX data types, which was another 1.7% of VAX instructions. All remaining VAX instructions were only used 0.2% of the time, and this allowed MicroVAX designs to eventually exceed the speed of full VAX implementations, before being replaced by the Alpha architecture.

RISC Refined: Berkeley RISC, Stanford MIPS

Some time after the 801, around 1981, projects at Berkeley (RISC I and II) and Stanford University (MIPS) further developed these concepts. The term RISC came from Berkeley's project, which was the basis for the fast Pyramid minicomputers and SPARC processor. Because of this, features are similar, including a windowed register file (10 global and 22 windowed, vs 8 and 24 for SPARC) with R0 wired to 0. Branches are delayed, and like ARM, all instructions have a bit to specify if condition codes should be set, and execute in a 3 stage pipeline. In addition, next and current PC are visible to the user, and last PC is visible in supervisor mode.

The Berkeley project also produced an instruction cache with some innovative features, such as instruction line prefetch that identified jump instructions, frequently used instructions compacted in memory and expanded upon cache load, multiple cache chips support, and bits to map out defective cache lines.

The Stanford MIPS project was the basis for the MIPS R2000, and like the case with Berkeley project, there are close similarities. MIPS stood for Microprocessor without Interlocked Pipeline Stages, using the compiler to eliminate register conflicts. Like the R2000, the MIPS had no condition code register, and a special HI/LO multiply and divide register pair.

Unlike the R2000, the MIPS had only 16 registers, and two delay slots for LOAD/STORE and branch instructions. The PC and last three PC values were tracked for exception handling. In addition, instructions were 'packed' (like the Berkeley RISC), in that many instructions specified two operations that were dispatched in consecutive cycles (not decoded by the cache). In this way, it was a 2 operation VLIW, but executed sequentially. User assembly language was translated to 'packed' format by the assembler.

Being experimental, there was no support for floating point operations.