From: bayko@hercules.cs.uregina.ca (J. Bayko)
Date: 12 Jun 92 18:49:18 GMT
Newsgroups: alt.folklore.computers,comp.arch
Subject: Great Microprocessors of the Past and Present

I had planned to submit an updated version of the Great Microprocessor list after I'd completed adding new processors, checked out little bits, and so on...

That was going to take a while...

But then I got to thinking of all the people stuck with the origional error-riddled version. All these poor people who'd kept it, or perhaps placed in in an FTP site, or even - using it for reference??!

In good conscience, I decided I can't leave that erratic old version lying out there, where people might think it's accurate. So here is an interim release. There will be more to it later - the 320xx will be included eventually, for example. But at least this has been debugged greatly.

Enjoy...

John Bayko.

—

Great Microprocessors of the Past and Present (V 3.2.1)

Section One: Before the Great Dark Cloud.

Part I: The Intel 4004 (1972)

The first single chip CPU was the Intel 4004, a 4-bit processor meant for a calculator. It processed data in 4 bits, but its instructions were 8 bits long. Internally, it featured four 12 bit(?) registers which acted as an internal evaluation stack. The Stack Pointer pointed to one of these registers, not a memory location (only CALL and RET instructions operated on the Stack Pointer). There were also sixteen 4-bit (or eight 8-bit) general purpose registers

The 4004 had 46 instructions. Intel created an 8-bit version of this, the 8008 (intended for a terminal controller).
[for additional information, see Appendix B]

Part II: The Intel 4040 and 8080

The 4040 was compatible with the 4004 instruction set - the 4040 had 60 instructions, which included the 46 4004 instructions. The 8080 was similar to the 4040, except being 8 bits wide.

The 8080 had a 16 bit address bus, and an 8 bit data bus. Internally it had seven 8 bit registers (six which could also be combined as three 16 bit registers), a 16 bit stack pointer (the stack was stored in memory, not in an internal register set), and 16 bit program counter. It also had several I/O ports - 256 of them, so I/O devices could be hooked up without taking away or interfering with the addressing space.

Part III: The Zilog Z-80 - End of the 8-bit line (July 1976)

The Z-80 was intended to be an improved 8080 (as was Intel's own 8085), and it was - vastly improved. It also used 8 bit data and 16 bit addressing, and could execute all of the 8080 op codes, but included 80 more, instructions that included 1, 4, 8 and 16 bit operations and even block move and block I/O instructions. The register set was doubled, with two banks of registers that could be switched between. This allowed fast operating system or interrupt context switches. The Z-80 also featured vectored interrupts.

Like many processors (including the 8085), the Z-80 featured many undocumented op codes. Chip area near the edge was used for added instructions, but fabrication made the failure of these 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 was actually the memory interface - the CPU generated it's own RAM refresh signals, which meant easier design and lower system cost. That and it's 8080 compatability, and CP/M, the first standard microprocessor operating system, made it the first choice of many systems.

Part IV: The 650x, Another Direction (1975-ish)

Shortly after the 8080, Motorola introduced the 6800. Some designers then started MOS Technologies, which introduced the 650x series, based on 6800 design (not a clone for legal reasons), and including the 6502 used in Commodores, Apples and Ataris. Steve Wozniak described it as the first chip you could get for less than a hundred dollars (a quarter of the 6800 price).

Unlike the 8080 and its kind, the 6502 had very few registers. It was an 8 bit processor, with 16 bit address bus. Inside was one 8 bit data register, and two 8 bit index registers and an 8 bit stack pointer (stack was preset from address 256 to 511). 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 with an 8-bit address that speeded operations (it didn't have to fetch a second byte for the address).

The 650x also had undocumented instructions.

As a side point, Apples, which were among the first microcomputers introduced, are still made, now using the 65816, which is compatable with the 6502, but has been expanded to 16 bits (including index and stack registers, and a 16-bit direct page register), and a 24-bit address bus. The Apple II line, which actually includes the Apple I, is the longest existing line of microcomputers.

Back when the 6502 was introduced, RAM was actually faster than CPUs, so it made sense to optimize for RAM access rather than increase the number of registers on a chip.

Part V: The 6809, extending the 650x

The 6800 from Motorola was essentially the same design as the 6502, but the latter left out one data register and added one index register, a minor change. But the 6809 was a major advance over both - at least relatively.

The 6809 had two 8 bit accumulators, rather than one in the 6502, and could combine them into a single 16 bit register. It also featured two index registers and two stack pointers, which allowed for some very advanced addressing modes. The 6809 was source compatible with the 6800, even though

the 6800 had 78 instructions and the 6809 only had around 59. Some instructions were replaced by more general ones which the assembler would translate, and some were even replaced by addressing modes.

Other features were one of the first multiplication instructions of the time, 16 bit arithmatic, and a special fast interrupt. But it was also highly optimized, gaining up to five times the speed of the 6800 series CPU. Like the 6800, it included the undocumented HCF (Halt Catch Fire) bus test instruction.

The 6800 lived on as well, becoming the 6801/3, which included ROM, some RAM, a serial I/O port, and other goodies on the chip. It was meant for embedded controllers, where the part count was to be minimized. The 6803 led to the 68HC11, and that was extended to 16 bits as the 68HC16. But the 6809 was a much faster and more flexible chip, particularly with the addition of the OS-9 operating system.

Of course, I'm a 6809 fan myself...

As a note, Hitachi produced a version called the 6309. Compatible with the 6809, it added 2 new 8-bit registers that could be added to form a second 16 bit register, and all four 8-bit registers could form a 32 bit register. It also featured division, and some 32 bit arithmatic, and was generally 30% faster in native mode. This information, suprisingly, was never published by Hitachi.

Part VI: Advanced Micro Devices Am2901, a few bits at a time

Bit slice processors were modular processors. Mostly, they consisted of an ALU of 1, 2, 4, or 8 bits, and control lines (including carry or overflow signals usually internal to the CPU). Two 4-bit ALUs could be arranged side by side, with control lines between them, to form an ALU of 8-bits, for example. A sequencer would execute a program to provide data and control signals.

The Am2901, from Advanced Micro Devices, was a popular 4-bit-slice processor. It featured sixteen 4-bit registers and a 4-bit ALU, and operation signals to allow carry/borrow or shift operations and such to operate across any number of other 2901s. An Am2910 address sequencer could provide control signals with the use of custom microcode in ROM.

The Am2903 featured hardware multiply support.

Section Two: Forgotten/Innovative Designs before the Great Dark Cloud

Part I: RCA 1802, wierdness at its best (1974)

The RCA 1802 was an odd beast, extremely simple and fabricated in CMOS, which allowed it to run at 6.4 MHz (very fast for 1974) or suspended with the clock stopped. It was an 8 bit processor, with 16 bit addressing, but the major features were it's extreme simplicity, and the flexibility of it's large register set. Simplicity was the primary design goal, and in that sense it was one of the first RISC chips.

It had sixteen 16-bit registers, which could be accessed as thirty-two 8 bit registers, and an accumulator D used for arithmatic and memory access - memory to D, then D to registers, and vice versa, using one 16-bit register as an address. This led to one person describing the 1802 as having 32 bytes of RAM and 65535 I/O ports. A 4-bit control register P selected any one general register as the program counter, while control registers X and N selected registers for I/O Index, and the operand for current instruction. All instructions were 8 bits - a 4-bit op code (total of 16 operations) and 4-bit

operand register stored in N.

There was no real conditional branching, no subroutine support, and no actual stack, but clever use of the register set allowed these to be implimented - for example, changing P to another register allowed jump to a subroutine. Similarly, on an interrupt P and X were saved, then R1 and R2 were selected for P and X until an RTI restored them.

A later version, the 1805, was enhanced, addind several Forth language primitives. Forth was commonly used in control applications.

Apart from the COSMAC microcomputer kit, the 1802 saw action in some video games from RCA and Radio Shack, and the chip is the heart of the Voyager, Viking and Galileo probes. One reason for this is that the 1802 was also fabricated mounted on saphire, which leads to radiation and static resistance, ideal for space operation.

Part II: Fairchild F8, Register windows

The F8 was an 8 bit processor. The processor itself didn't have an address bus - program and data memory access were contained in separate units, which reduced the number of pins, and the associated cost. It also featured 64 registers, accessed by the ISAR register in cells (windows) of eight, which meant external RAM wasn't always needed for small applications. In addition, the 2-chip processor didn't need support chips, unlike others which needed seven or more.

The use of the ISAR register allowed a subroutine to be entered without saving a bunch of registers, speeding execution - the ISAR would just be changed. Special purpose registers were stored in the second cell (regs 9-15), so the first window would be used for global variables.

The windowing concept was useful, but only the register pointed to by the ISAR could be accessed, limiting usefulness - to access another register the ISAR was incremented or decremented through the window.

Part III: SC/MP, early advanced multiprocessing (April 1976)

The National Semiconductor SC/MP, (nicknamed "Scamp") was a typical well-designed 8 bit processor intended for control applications. It featured 16 bit addressing, with 12 address lines and 4 lines borrowed from the data bus (it was common to borrow lines from the data bus for addressing). Internally, it included three index registers.

The unique feature was the ability to completely share a system bus with other processors. Most processors of the time assumed they were the only ones accessing memory or I/O devices. Multiple SC/MPs could be hooked up to the bus, as well as other intelligent devices, such as DMA controllers. A control line could be chained along the processors to allow cooperative processing.

This was very advanced for the time, compared to other CPUs. However, the SC/MP had no stack or subroutine instructions, though a stack could be emulated to some degree. It's intent was for embedded control, and these features were omitted for cost reasons. It was also bit serial internally to keep it cheap.

Part IV: F100-L, a self expanding design

The Ferranti F100-L was designed by a British company for the British Military. It was an 8 bit processor, with 16 bit addressing, but it could only

access 32K of memory (1 bit for indirection).

The unique feature of the F100-L was that it had a complete control bus available for a coprocessor that could be added on. Any instruction the F100-L couldn't decode was sent directly to the coprocessor for processing. Applications for coprocessors at the time were limited, but the idea is still used in modern processors. This coprocessor design was adapted by other processors, such as the National Semiconductor 320xx series, which included FPU, MMU, and other coprocessors that could just be added to the CPU's coprocessor bus in a chain. Other units not forseen could be added later.

The NS 320xx series was the predecessor of the Swordfish processor, described later.

Part V: The Western Digital 3-chip CPU (June 1976)

The Western Digital MCP-1600 was probably the most flexible processor available. It consisted of at least four separate chips, including the control circutry unit, the ALU, two or four ROM chips with microcode, and timing circuitry. It doesn't really count as a microprocessor, but neither do bit-slice processors (AMD 2901).

The ALU chip contained twenty six 8 bit registers and an 8 bit ALU, while the control unit supervised the moving of data, memory access, and other control functions. The ROM allowed the chip to function as either an 8 bit chip or 16 bit, with clever use of the 8 bit ALU. Even more, microcode allowed the addition of Floating Point routines, simplifying programming (and possibly producing a Floating Point Coprocessor).

Two standard microcode ROMS were available. This flexibility was one reason it was also used to impliment the LSI-11 processor as well as the WD Pascal Microengine.

Part VI: Intel 8048, Harvard bus

The MCS-48 was a microcontroller familly, not a single microprocessor, but did feature a separate data and program bus, known as Harvard Architecture. In theory, It allows simultaneous reading of data while the next instruction is fetched, allowing an increase in speed. In this case, its purpose was to allow different size data (RAM) and program (usually ROM) memory.

In other aspects as well, the MSC-48 series was small, featuring only a single 8 bit register, around 4K of program memory, and 256 bytes of data memory (organized in register pages, like the F8 processor). It did have stack and subroutine operations, but the 256 byte memory limit made that limited.

Part VII: Motorolla MC14500B ICU, one bit at a time

Probably the limit in small processors was the 1 bit 14500B from Motorola. It had a 4 bit instruction, and controlled a single signal line for application control. It had no address bus - that was an external unit that was added on. Another CPU could be used to feed control instructions to the 14500B in an application.

It had only 16 pins, less than a typical RAM chip, and ran at 1 MHz.

Section Three: The Great Dark Cloud Falls: IBM's Choice.

———————————————-

Part I: TMS 9900, first of the 16 bits (June 1976)

One of the first true 16 bit microprocessors was the TMS 9900, by Texas Instruments. It was designed as a single chip version of the TI 990 minicomputer series, much like the 12 bit Intersil 6100 was a single chip PDP-8E, used in the DECmate, DEC's competition for the IBM PC. Because of this, the TMS 9900 had a mature, well thought out design.

It had a 15 bit address space and two internal 16 bit registers. One unique feature, though, was that all its registers were actually kept in memory - this included stack pointers and the program counter. A single workspace register pointed to the register set in RAM, so when a subroutine was entered or an interrupt was processed, only the single workspace register had to be changed - unlike some CPUs which required dozens or more register saves before acknowleging a context switch.

This was feasable at the time because RAM was faster than the CPUs. A few modern designs, such as the INMOS Transputers, use this same design using caches or rotating buffers, for the same reason of improved context switches. Some chips of the time, such as the 650x series had a similar philosophy, using index registers, but the TMS 9900 went the farthest in this direction.

That wasn't the only positive feature of the chip. It had good interrupt handling features and very good instruction set. In typical comparisons with the Intel 8086, the TMS9900 had smaller and much faster programs. The only disadvantage was the small address space and need for fast RAM.

Despite the very poor support from Texas Instruments, the TMS 9900 had the potential at one point to surpass the 8086 in popularity.

Part II: Zilog Z-8000, another direct competitor.

The Z-8000 was introduced not long after the 8086, but had superior features. It was basically a 16 bit processor, but could address up to 23 bits in some versions by using segmentation. The segment simply held the upper 7 bits for the register. There was also an unsegmented version, but both could be extended further with an additional MMU that used paging and 64 segment registers.

Internally, the Z-8000 had sixteen 16 bit registers, but register size and use were exceedingly flexible. The Z-8000 registers could be used as sixteen 8 bit registers (only the first half were used like this), sixteen 16-bit registers, eight 32 bit registers, or four 64 bit registers, and included 32-bit multiply and divide. They were all general purpose registers - the stack pointer was typically register 15, with register 14 holding the segment (you could just access it as a 32 bit register for address calculations.

The Z-8000 featured two modes, one for the operating system and one for user programs. The user mode prevented the user from messing about with interrupt handling and other potentially dangerous stuff.

Finally, like the Z-80, the Z-8000 featured automatic RAM refresh circuitry. Unfortunately it was somewhat slow, but the features generally made up for that. Initial bugs also hindered its acceptance. There was a radiation resistant military version.

A later version, the Z-80000, was expanded to 32 bits internally.

Part III: Motorola 68000, a refined 16/32 bit CPU

The 68000 was actually a 32 bit architecture internally, but 16 bit externally for packaging reasons. It also included 24 bit addressing, without

the use of segment registers. That meant that a single directly accessed array or structure could be larger than 64K in size. Addresses were computed as 32 bit, but the top 8 bits were cut to reduce costs. No segments made programming the 68000 easier than competing processors.

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

The 68000 had sixteen registers, split into data and address registers. One address register was reserved for the Stack Pointer. Both types of registers could be used for any function except for direct addressing. Only address registers could be used as the source of an address, but data registers could provide the offset from an address.

Like the Z-8000, the 68000 featured a supervisor and user mode. There were two Stack Pointers, one for supervisor and one for user mode. The Z-8000 and 68000 were similar in capabilities, but the 68000 had 32 bit units internally, making it faster and eliminating forced segmentations. It was designed for expansion, including specifications for floating point and string operations (floating point later implimented in the 68040).

Part IV: Intel 8086, IBM's choice (1978)

The Intel 8086 was based on the design of the 8085, but with enhancements and as a 16 bit processor. It had a similar register set, and featured a 6 byte prefetch instruction queue for additional speed.

It featured four 16 bit general registers, which could be accessed as eight 8 bit registers or four 16 bit index registers, and four 16 bit segment registers. The data registers were often used implicitly by instructions, complicating register allocation for temporary values. There were also four index registers, which included the stack pointer.

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, segments overlapped, and 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. Most people consider this a brain damaged design.

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. A later version, the 80386, expanded the design to 32 bits, and 'fixed' the segmentation, but required extra modes (supressing the new features) for compatability.

So why did IBM chose the 8086 series 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 it's bubble memory designs. Apparently IBM was using 8086s in the IBM Displaywriter word processor.

Other factors were the 8-bit 8088 version, which could use existing 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.

Intel bubble memory was on the market for a while, but faded away as

better and cheaper memory technologies arrived.

Section Four: Unix and RISC, a New Hope

Part I: SPARC, an extreme windowed RISC

SPARC, or the Scalable 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 RISC processors had promised a major step forward in speed [See Appendix A], but existing manufacturers were slow to introduce a RISC type processor, so Sun went ahead and developed its own (based on Berkley'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 design) and Hewlitt-Packard Precision Architecture CPU. Most RISC CPUs are more conventional, but the SPARC is a good example of extreme RISC philosophies, originally even forgoing useful multi-cycle instructions like multiply and divide.

SPARC usually contains about 128 registers, compared to 16 for previous CISC designs. 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. This allows functions to be called in as little as 1 cycle. Registers are loaded or saved only at the top or bottom of the register stack. Like most RISC processors, global register zero is wired to zero to simplify instructions, and SPARC is pipelined for performance.

SPARC is 'scalable' mainly because the register stack can be expanded, to reduce loads and saves between functions, or scaled down to reduce interrupt or context switch time, when the entire 128 register set has to be written to memory. Function calls are usually much more frequent, so the large register set is usually a plus.

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. Most versions are 32 bits, but there are designs for 64 bit and superscalar versions. SPARC was submitted to the IEEE society to be considered for the P1754 microprocessor standard.

Part II: AMD 29000, a flexible register set (1986?)

The AMD 29000 is another RISC CPU descended from the Berkley RISC design. 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, while the caller's frame is stored in current frame's register, like in an ordinary stack. 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.

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.

Slomewhat more elegant than the SPARC, the AMD 29K can also claim to be 'the most popular RISC processor'.

Part III: Motorola 88000, a conservative RISC

A design that is typical of most current RISC processors is the Motorola 88000 (originally named the 78000). It is a 32 bit processor with Harvard architecture (separate data and instruction buses). Each bus has a separate cache, so simultaneous data and instruction access doesn't conflict. It is similar to the Hewlett Packard Precision Architecture in design, though slightly more elegant.

The chip contains thirty two 32 bit registers, and is organized with separate function units internally - an ALU and a floating point unit in the 88100 version. Other special function units, such as graphics, vector operations, and such can be added to the design to produce a custom design for customers. Additional ALU and FPU units can allow a superscalar operation of the CPU (as in the 88110 version, for example). The function units of the 88100 share the same register set, while the 88110, like most chips, has a separate set of thirty two 80-bit registers for the FPU.

The ALU typically executes in 1 cycle, but it or the FPU can take several clock cycles for an operation (ex. multiplication). For performance, the units are pipelined, so one instruction can be issued each cycle, with the result appearing several cycles later. To keep track of this latency, the 88000 has a scoreboard register which keeps track of registers, and ensures the result arrives before operation continues.

In the superscalar 88110, the result from one ALU can be fed directly into another in the next clock cycle (as opposed to saving to a register first), saving a clock cycle between instructions. Also, loads and saves are buffered so the processor doesn't have to wait, except when loading from a memory location still waiting for a save to complete. The 88110 version can also speculatively execute conditional branches in the pipeline. If the speculation is true, there is no branch delay in the pipeline. Otherwise, the operations are rolled back from a history buffer (at least 1 cycle penalty), and the other fork of the branch is taken.

Part IV: Acorn ARM, RISC for home use (1985)

ARM (Advanced RISC Machine) is often praised as one of the most elegant modern processors in existance. 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.

The original ARM2 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 compatability) and sixteen

registers. There is a multiple load/save instruction (many registers are shadowed in other modes). The ARM series consists of the ARM6 CPU core, which can be used as the basis for a custom CPU, the ARM60 base CPU, and the ARM600 which also includes 4K cache, MMU, write buffer, and coprocessor interface. It can be big- or little-endian.

A unique feature of ARM is that every instruction features a 4 bit condition code (including 'never execute'). This easily eliminates many branches and can speed execution. Every instruction has a bit to indicate if condition codes should be set - an instruction can set them, and several intervening ininstructions can execute before the codes are used. Addressing also features a very useful mode (base register indexed by index register shifted by a constant - ra + rb << k) found in few other processors.

The ARM hasn't had the development put into it that others have, so there aren't superscalar or superpipelined versions, and the clock rate is not breathtaking. However it wasn't meant to break speed records, and is a very elegant, fast and low cost CPU.

Section Five: Just Beyond Scalar

Part I: Intel 860, "Cray on a Chip"

The Intel 860 wasn't Intel's first RISC chip - that was the 960, but the 960 was slower, and marketed for embedded control applications. The 860 was an impressive chip when introduced, 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 at the time. It has lagged behind newer designs, though.

The 860 has several modes, from regular scaler mode to a superscaler mode that executes two instructions per cycle and a pipelined mode. It can use the entire 8K data cache as a vector register, in the same way that supercomputers like Crays do.

The 860 is a 64 bit processor essentially - though it normally uses thirty two 32 bit registers it has sixteen 64 bit floating point registers. It contains not only an integer ALU but a FPU, and even more unusual, a 3-D graphics unit that performs lines, Gouraud shading, Z-buffering for hidden line removal, and other operations, all in conjunction with the FPU. It has separate instruction and data busses, and can access 4 G of memory, with segments. It also includes a Memory Management Unit for virtual storage.

In a single cycle, the 860 can do an integer operation, and a special multiply and add floating point operation, for a total of three instructions. Actually getting the chip at top speed usually requires using assembly language - using standard compilers gives it a speed similar to other processors. Because of this, it's best used as a coprocessor, either for graphics, like the NeXTdimension board, or floating point acceleration, like add in units for workstations.

Another problem is the difficulty handling interrupts. It is extensively pipelined, having as many as four pipes going 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)

Part II: IBM RS/6000 POWER chip

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 Computing, but IBM calls it Reduced Instruction-Cycle Set Computing, and implimented a complex processor with more instructions than most CISC processors. They ended up with was a CPU that actually contains five or seven separate chips.

The chips are the branch unit, fixed point unit, floating point unit, and either two or four cache chips.

The branch unit is the heart of the CPU, and actually enables up to five instructions to be executed at once, though three is more common. It contains the condition code register, performs checks on this register, and also performs branches. It also dispatches instructions to the fixed or floating point units. For added speed, it contains its own loop register (for decrement and branch on zero). The condition code register has eight fields - two reserved for the fixed and floating point units, the other six settable separately. This allows a field to be set, then a branch can occur several instructions later. In addition, the branch unit can speculatively execute conditional branches, and then cancel the instructions if the branch is not taken.

The fixed point unit performs integer operations, as well as some complex string instructions and multiple load and store. The fixed unit contains thirty two 32 bit registers.

The floating point unit contains thirty two 64 bit registers and performs all typical floating point operations. In addition, like the Intel 860, the floating unit has a special multiply and add instruction. The floating unit performs this operation with up to 162 bits of precision before truncating it to 64 bits. Often this type of operation can produce over +/-100% error if too few bits are used. The registers are loaded and stored by the fixed point unit.

Unlike the Intel 860, the POWER CPU does not need a special mode - the branch unit actually reschedules instructions to operate faster. It also handles interrupts reasonably well, and is overall a very good design. It does, however, violate the RISC philosophy of fewer instructions at over a hundred, versus only about 34 for the ARM and 52 for the Motorola 88000 (including FPU instructions). A single chip POWER CPU (no separate branch unit) designed, to be manufactured by IBM and Motorola, qualifies it as a microprocessor.

Part III: National Semiconductor Swordfish

The Intel 860 is a superscaler chip, but is essentially a VLIW, or Very Long Instruction Word processor, which means that more than one instruction is contained in a single instruction word - in this case, two. The IBM POWER CPU reshedules instructions on the run, which gives it more flexibility, but it can only execute different types of instructions at the same time - one integer and one floating point, for example.

The Swordfish chip contains two separate integer units, allowing two integer instructions to execute at once, along with one floating point add/subtract and yet another DSP unit for multiplies, for a theoretical total of four instructions at once.

The CPU is a 32 bit processor, but has a 64 bit data bus for fetching multiple instructions at once. It features an instruction loader which functions like the branch unit in the IBM POWER CPU, but lacks the extended functions - it can’t branch concurrently. It also features a Digital Signal Processing (DSP) unit to perform multiplies and other DSP operations, and a

separate FPU. The DSP also performs single cycle integer multiplies, a task that usually takes around seven cycles for most integer ALUs.

It is RISC in the sense that it executes instructions in one cycle, but it doesn't use register windowing like the SPARC processor. It performs 20 MFLOPS at 50 MHz, which is good compared to other RISC chips, but slow compared to dedicated DSPs. Still, the FPU and integer units result in most processing tasks being faster - at 50 MHz the chip runs about 100 MIPS. The POWER CPU is about as fast, but operates at half the clock speed.

The Swordfish's neatest features, though, are it's hardware. It automatically can adjust it's data bus size, from 64 bits to 32, 16, or even 8 bits, making it very easy to design for. In addition, it can run from a 50MHz clock, or a 25 MHz clock in which case it multiplies the clock internally back to 50 MHz. This allows 25MHz parts to be used with it. It also features two DMA channels and a timer unit.

Part IV: DEC Alpha, Designed for the future (1992)

The DEC Alpha architecture is designed, according to DEC, for a operational life of 25 years. It doesn't contain particular innovations (although the PALcall operation is unusual), but is an elegant blend of features, selected to ensure no obvious limits to future performance - no special registers, etc. The 21064 is DEC's first Alpha chip.

It is a 64 bit chip 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. Oddly, Alpha does not provide a divide instruction.

One reason for Alpha is to replace DEC's two previous architectures - the VAX and MIPS CPUs. To do this, the chip provides both IEEE and VAX floating point operations. It also features a Privleged Architecture Library (PAL) calls, a set of programmable macros written in the Alpha instruction set, similar to the programmable microcode of the Western Digital MCP-1600 or the AMD Am2910 CPUs. It provides support of various operating systems - VMS, Unix or Microsoft NT.

Alpha was also designed for the future, including superscalar, multiprocessing, and high speed clock. Because of this, superscalar insturcions may be reordered and trap conditions are imprecise. Special instructions are available to control both occurrences when needed. SPARC also has a specification for instruction ordering.

Section Six: Wierd and Innovative Chips

Part I: T-9000, parallel computing (1990)

The INMOS T-9000 is the latest version of the Transputer architecture, a processor designed to be hooked up to other processors for high speed parallel processing. The previous versions were the 16 bit T-212 and 32 bit T-414 and T-800 processors (1985). The T-800 included a 64 bit floating point unit. They used some RISC principles before it became fashionable, but the most important feature was that each chip contained 4 serial ports to connect the chips in a network.

Although the previous chips were fast, the T-9000 is a much extended design. It starts with an architecture like the T-800. It contains only three

registers that are used as an evaluation stack - they are not general purpose. Instead, like the TMS 9900, it uses memory for registers, and points to the workspace using a workspace register and cache based on 32 word rotating buffers. This allows very fast context switching, less than a microsecond.

Unlike when the TMS 9900 was created, the T-9000 is far faster than memory access, so the CPU has several levels of very high speed caches and memory levels. It not only speeds process scheduling, is simplifies it enough that task switching is automated in hardware, unlike most processors. The Intel 432, described later, also attempted hardware process scheduling, but was unsuccessful at it. The main cache is 16 K, and is designed for 3 reads and 1 write simultaneously. The workspace cache allows 2 reads and 1 write simultaneously.

The T-9000 contains 4 main internal units, the CPU, the VCP, which performs the communications between chips, the PMI, which manages memory, and the Scheduler. There's also an instruction grouper which can schedule five instruction stages in the most efficient manner. The grouper can start an instruction in any stage (bypassing unneeded stages) and instructions don't need to pass through every stage, and can leave when finished, freeing the pipeline for other instructions. Instructions themselves are unique in that frequently used instructions are stored in only a byte, but other instructions are represented by multiple bytes.

The serial links allow communications at close to the speed of direct memory access, and the VCP unit allows this to take place without the CPU being aware of it. 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.

Like the Swordfish CPU, the T-9000 PMI (Programable Memory Interface) can adapt to a 64, 32, 16, or 8 bit bus.

Part II: 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 - the 80286 was envisioned as a step between the 8086 and the 432. 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, so it wasn't really a microprocessor.

The GDP was exclusively object oriented - normal linear memory access wasn't allowed. It was designed with the Ada programming language in mind. It had hardware support for data hiding, methods, inheritance, late binding, and access protection. Unfortunately this meant that every memory access was checked, which slowed execution (despite some caching). The stack oriented design meant the GDP had no local user data registers. It supported up to 2^24 segments, each limited to 64K in size, but the object oriented nature of the design meant that was not a real limitation. Instructions were bit encoded, ranging from 6 bits to 321 bits long (like the T-9000) and could be very complex.

The BIU defined the bus. BIUs were 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

pairity 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, up to five or ten times slower than a 16-bit 68000. Part of this was the lack of local data registers, or a significant cache. Part of this was the fault-tolerant BIU, which defined an asyncronous 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 often required multiple memory accesses for permission checks when data was accessed. Finally, the implimentation of the GDP on two chips instead of one produced a slower product.

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

Part III: Rekursiv, an object oriented processor

The Rekursiv processor is actually a processor board, not a microprocessor, but is neat. It was created by a manufacturing company called Linn, to control their manufacturing system. The owner was a believer in automation, and had automated the company as much as possible with Vaxes, but wasn't satisfied, so hired software experts to design a new system, which they called LINGO. It was completely object oriented, like smalltalk (and unlike C++, which allows some object concepts, but handles them in a conventional way), but too slow on the VAXes, so Linn commissioned a processor designed for the language.

This is not the only processor designed specifically for a language that is slow on other CPUs. Several specialized LISP processors, such as the Scheme-79 lisp processor, were created, but this chip is unique in its object oriented features. It also manages to support objects without the slowness of the Intel 432.

The Rekursiv processor features a writable instruction set, and is highly parallel. It uses 40 bits for objects, and 24 bit addressing, kind of. Memory can't be addressed directly, only through the object identifiers, which are 40 bit tags. The hardware handles all objects in memory and on disk, and swapping them to disk. It has no real program - all data and code/methods are embedded in the objects, and loaded when a message is sent to them. There is a page table which stores the object tags and maps them into memory.

There is a 64k area, arranges 16k X 128 bit words, for microcode, allowing an instruction set to be constructed on the fly. It can change for different objects.

The CPU hardware creates, loads, saves, destroys, and manipulates objects. The manipulation is accomplished with a standard AMD 29203 CPU, but the other parts are specially designed. It executes LINGO entirely fast enough, and is a perfect match between language and CPU, but it can execute more conventional languages, such as Smalltalk or C if needed - possible simultaneously, as separate complete objects.

John Bayko (Tau).

Appendix A:

==========

RISC and CISC definitions:
————————————-

RISC refers to a Reduced Instruction Set Computer. IBM pioneered many RISC ideas (and the acronym) in their 801 project. RISC ideas also come from the CDC 6600 computer and projects at Berkley (RISC I and II and SOAR) and Stanford University (the MIPS project). RISC designs call for each instruction to execute in a single cycle, which is done with pipelines, no microcode (to reduce chip complexity and increase speed). Operations are performed on registers only (with the only memory access being loading and storing). Finally, many RISC designs uses a large windowed register set to speed subroutine calls (see the entry on SPARC for a description).

But despite these specifications, RISC is more a philosophy than a set of design criteria, and almost everything is called RISC, even if it isn't. Pipelines are used in the 68040 and 80486 CISC processors to execute instructions in a single cycle, even though they use microcode, and experiments have shown that windowed registers can be added to CISC designs, speeding them up in a similar way. Basically, RISC asks whether an instruction is necessary, or whether it can be replaced by several simpler instructions without a major performance loss. Typically multiply and divide instructions are necessary, despite the fact they require multiple cycles to execute. The advantage is that a simpler chip can run at a higher clock speed.

CISC refers to a Complex Instruction Set Computer. There's not really a set of design features to characterize it like there is for RISC, but small register sets, large instruction sets, and use of microcode are common. The philosophy is that if a complex instruction can result in an overal increase in speed, it's good. The disadvantage is that it's harder to increase the clock speed of a complex chip.

Microcode was a way of simplifying processor design. Even though it resulted in instructions that are slower requiring multiple clock cycles, it was thought it would allow more complex instructions to better support high level languages, leading to better performance. However, most complex instructions are seldom used.

Processor Classifications:
————————————-

Arbitrarily assigned by me...

```
    ___CISC_____________________________________________________________RISC
    |                                        14500B
4-bit |                                    Am2903
    |                      4004
    |                    4040
8-bit |                                      1802
    |                      8008
    |                    8080      2650,SC/MP  F8
    |              F100-L       650x,6800
    |
    |       MCP1600 Z-80             6809
16-bit|
```

```
     |                8086    TMS9900
     |           Z8000          65816
     |
     |               68000
32-bit|                                      SPARC ARM
     | 432                                R2000
     |         Z80000     68040
     |              —<T9000>-Sword–HP-PA      88100
     | Rekurs     RS/6000          -fish       88110
64-bit|         ——-<860>——
     |                                      Alpha
     |
```

Appendix B:
==========

Appearing in IEEE Computer 1972:
————————————-

NEW
PRODUCTS

FEATURE PRODUCT

COMPUTER ON A CHIP

Intel has introduced an integrated CPU complete with a 4-bit parallel adder, sixteen 4-bit registers, an accumulator and a push-down stack on one chip. It's one of a family of four new ICs which comprise the MCS-4 micro computer system–the first system to bring the power and flexibility of a dedicated general-purpose computer at low cost in as few as two dual in-line packages.

MSC-4 systems provide complete computing and control functions for test systems, data terminals, billing machines, measuring systems, numeric control systems and process control systems.

The heart of any MSC-4 system is a Type 4004 CPU, which incudes a set of 45 instructions. Adding one or more Type 4001 ROMs for program storage and data tables gives a fully functioning micro-programmed computer. Add Type 4002 RAMs for read-write memory and Type 4003 registers to expand the output ports.

Using no curcuitry other than ICs from this family of four, a system with 4096 8-bit bytes of ROM storage and 5120 bits of RAM storage can be created. For rapid turn-around or only a few systems, Intel's erasable and re-programmable ROM, Type 1701, may be substituted for the Type 4001 mask-programmed ROM.

MCS-4 systems interface easily with switches, keyboards, displays, teletypewriters, printers, readers, A-D converters and other popular peripherals. For further information, circle the reader service card 87 or call Intel

at (408) 246-7501.
Circle 87 on Reader Service Card

COMPUTER/JANUARY/FEBRUARY 1971/71

Appearing in IEEE Computer 1975:
———————————-

The age of the affordable computer.

MITS announces the dawning of the Altair 8800 Computer. A lot of brain power at a price that's bound to create love and understanding. To say nothing of excitement.

The Altair 8800 uses a parallel, 8-bit processor (the Intel 8080) with a 16-bit address. It has 78 basic machine instrcutions with variences ofer 200 instructions. It can directly address up to 65K bytes of memory and it is fast. Very fast. The Altair 8800's basic instruction sysle time is 2 microseconds.

Combine this speed and power with Altair's flexibility (it can directly address 256 input and 256 output devices) and you have a computer that's competitive with most mini's on the market today.

The basic Altair 8800 Computer includes the CPU, front panel control board, front panel lights and switches, power supply (enough to power any additional cards), and expander board (with room for 3 extra cards) all enclosed in a hansome, aluminum case. Up to 16 cards can be added inside the main case.

Options now available include 4K dynamic memory cards, 1K static memory cards, parallel I/O cards, three seial I/O cards (TTL, R232, and TTY), octal to binary computer terminal, 32 character alpha-numeric display terminal, ASCII keyboard, audio tape interface, 4 channel storage scope (for testing), and expander cards.

Options under development include a floppy disc system, CRT terminal, line printer, floating point processor, vectored interrupt (8 levels), PROM programmer, direct memory access controller and much more.

PRICE
Altair 8800 Computer: $439.00* kit
$621.00* assembled

prices and specifications subject to change without notice

For more information or our free Altair Systems Catalogue phone or write: MITS, 6328 Linn N.E., Albuquerque, N.M. 87108, 505/265-7553.

*In quantities of 1 (one). Substantial OEM discounts available.

[Picture of computer, with switches and lights]

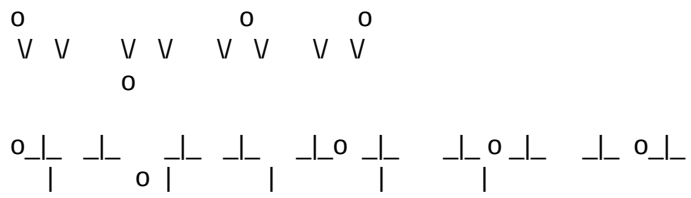
Appendix C:
==========

Bubble Memories:
————————

Certain materials (ie. gadolinium gallium garnet) are magnetizable easily in only one direction. A film of these materials can be created so that it's magnetizable in an up-down direction. The magnetic fields tend to stick together, so you get a pattern that is kind of like air bubbles in water squished between glass, half with the north pole facing up, half with the south, floating inside the film. When a vertical magnetic field is imposed on this, the areas in opposite alignment to this field shrink to circles, or 'bubbles'.

A bubble can be formed by reversing the field in a small spot, and can be destroyed by increasing the field.

The bubbles are anchored to tiny magnetic posts arranged in lines. Usually a 'V V V' shape or a 'T T T' shape. Another magnetic field is applied across the chip, which is picked up by the posts and hods the bubble. The field is rotated 90 degrees, and the bubble is attracted to another part of the post. After four rotations, a bubble gets moved to the next post:



I hope that diagram makes sense.

These bubbles move in long thin loops arranged in rows. At the end of the row, the bits to be read are copied to another loop that shift to read and write units that create or destroy bubbles. Access time for a particular bit depends on where it is, so it's not consistant.

One of the limitations with bubble memories, why they were superceded, was the slow access. A large bubble memory would require large loops, so accessing a bit could require cycling through a huge number of other bits first. The speed of propagation is limited by how fast magnetic fields could be switched back and forth, a limit of about 1 MHz. On the plus side, they are non-volatile, but eeproms, flash memories, and ferroelectric technologies are also non-volatile and and are faster.

Ferroelectric and Ferromagnetic (core) Memories:
————————————————————

Ferroelectric materials are analogous to ferromagnetic materials, though neither actually need to contain any iron. Ferromagnetic materials, used in core memories, will retain a magnetic field that's been applied to it.

Core memories consist of ferromagnetic rings strung together on tiny wires. The wires will induce magnetic fields in the rings, which can later be

read back. Usually reading this memory will erase it, so once a bit is read, it is written back. This type of memory is expensive because it has to be constructed physically, but is very fast and non-volitile. Unfortunately it's also large and heavy, compared to other technologies.

Legend reports that a Swedish jet prototype (the Viggen I believe) once crashed, but the flight recorders weren't fast enough to record the cause of the crash. The flight computers used core memory, though, so they were hooked up and read out, and the still contained the data microseconds before the crash occurred, allowing the cause to be determined.

Ferroelectric materials retain an electric field rather than a magnetic field. like core memories, they are fast and non-volatile, but bits have to be rewritten when read. Unlike core memories, ferroelectric memories can be fabricated on silicon chips in high density and at low cost.