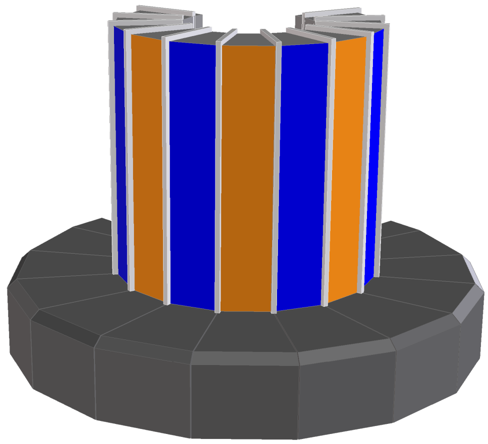

The Cray-1 Supercomputer

By Andie Hioki

San Jose State University
Engr. 120 Computer Organization & Architecture
Prof. Richard Sinn
Spring, 2002 Writing Project

OUTLINE

THE CRAY-1 SUPERCOMPUTER

Introduction

In the mid-to-late 1970's, the Cray-1 was the fastest computer in the world, with a clock speed of 12.5 ns (80 MHz), computational rates of 138 million floating-point operations per second (MFLOPS) during sustained periods, and 250 MFLOPS in short bursts. Up until that time, there has been no other computer like it. The Cray-1 had spawned a new class of computers called the "supercomputer," a computer highly optimized for computational speed, and is typically used for its mega number-crunching capabilities. Considered to be the first supercomputer, Cray Research's Cray-1 was unveiled in 1976 by Seymour R. Cray, its inventor and chief design architect.

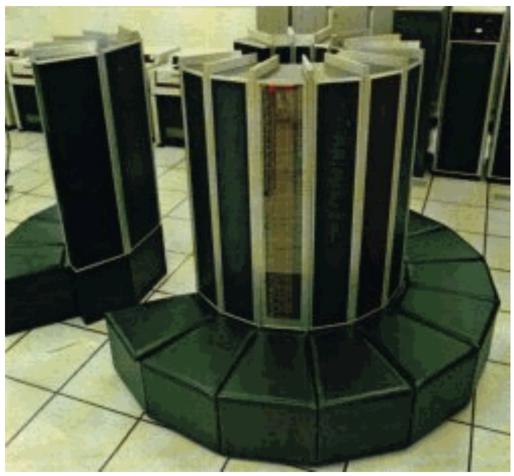

Figure A. The Cray-1.
(Photo from ESP Educational Support Package:
Cray Historical Archives.)

The installation of the first Cray-1 became a competition between the Lawrence Livermore National Laboratory and Los Alamos National Laboratory. The Energy & Research Development Administration decided to approve funding for only one of the requests submitted by each of the labs. Fierce competition led the labs to prevent each other from purchasing the first Cray-1. Finally Seymour Cray loosened the gridlock by offering to give Los Alamos the machine for six months, at the end of which they could decide to keep it, lease it, or give it back. Unable to refuse the offer, the first Cray-1 was installed at Los Alamos National Laboratory.

The first paying customer of the Cray-1 was the National Center for Atmospheric Research (NCAR) in Boulder, Colorado. They learned of it from the programmers at Los Alamos who disclosed that the Cray-1 was five times faster than their best computer, the Control Data Corp. 7600 (in vector mode), incidentally also designed by Cray. They paid $8.86 million to use it for weather forecasting. Besides weather forecasting, the type of problems that the Cray-1 was suited to included fluid flow simulation, aircraft design, nuclear research, seismic analysis (especially those involved in oil exploration), and other computationally intense areas.

Features of the Cray-1 are detailed below. Among them are its architecture, CPU characteristics, I/O, and its applications software.

Back to OUTLINE.

Architecture: Physical Dimensions

The physical dimensions of the Cray-1 are shown in Figure B. It was laid out in a 270° arc with 12 wedge-like columns. With a base of 103-1/2", columns 77" high, and a weight of 10,500 lbs. (with max. memory option), it was considered "small" at the time. This "small" size was considered a plus because distances which signals must travel were reduced, thus helping to attain a clock speed of 12.5 ns. A cylindrical shape was chosen to keep wire lengths relatively short. No wire was more than four feet long.

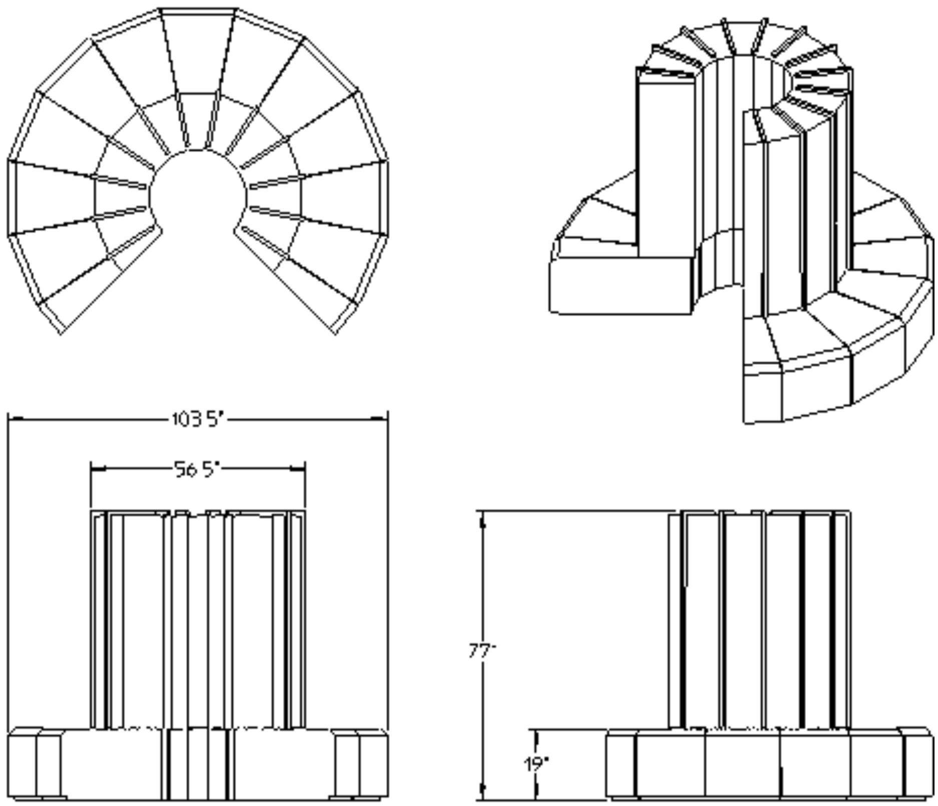

Figure B. Physical dimensions of the Cray-1.
(Courtesy of R. Sean Murphy, Agilent Technologies, San Jose, CA)

Back to OUTLINE.

Architecture: Physical Organization of the Cray-1

Figure C shows the overall system structure of the Cray-1. At the core of the Cray-1 were twelve functional units (pipeline processors), described below, that implemented a comprehensive set of scalar and vector instructions. These units, or processors, communicated directly with a huge set of high-speed registers giving the Cray-1 a register-to-register architecture. This was in contrast with a memory-to-memory architecture of some of the Cray-1's predecessors, such as the STAR-100.

Its computation section included:

- Scalar and vector processing
- 12.5 ns clock period
- 80 million instructions per second (MIPS)
- 64-bit word size
- Integer and floating-point arithmetic
- 12 functional units (pipelined processors)
- Address (*A*) registers: eight 24-bit *A* registers
- Scalar (*S*) registers: eight 64-bit S registers
- Intermediate address (*B*) registers: sixty-four 24-bit *B* registers
- Vector (*V*) registers: sixty-four 64-bit V registers
- Vector Length (*VL*) and Vector Mask (*VM*) registers
- One 64-bit real-time clock register
- Instruction buffers: four buffers, 64 16-bit instruction parcels
- 128 basic instructions
- Prioritized interrupt control

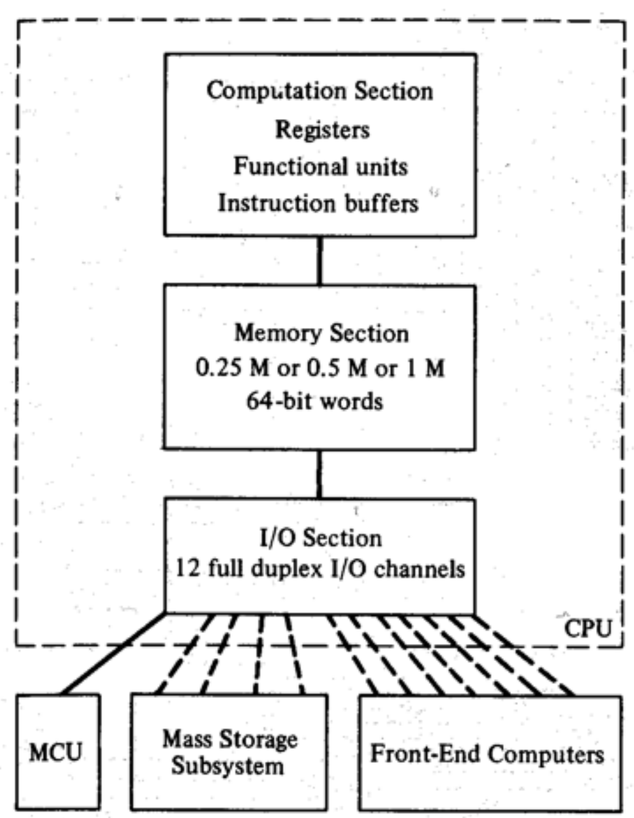

Figure C. Overall system structure of the Cray-1.
(From Gorsline.)

Its memory section included:

- 256K, 512K, or 1M 64-bit words (8 MB)
- 16 independent banks of 65K words each
- 4 clock period (50 ns) bank cycle time
- 1 word per clock period transfer rate for *B*,*T*, and *V* registers
- 1 word per 2 clock period transfer rate for *A* and *S* registers
- 4 words per clock period transfer rate to instruction buffers

  The I/O section included:

- 24 I/O channels, divided into four 6-channel groups
- Each channel group has either 6 input or 6 output channels
- Each channel group is served every 4 clock cycles by memory
- Channel priority within each channel group
- 16 data bits, 3 control bits per channel, 4 parity bits
- Max. channel rate of one 64-bit word every 100 ns
- Max. data streaming rate of 500K 64-bit words per second
- Channel error detection

Back to OUTLINE.

Architecture: The Modules

The Cray-1 module was a 5-layer printed circuit board. The two outer layers were used as signal layers; three inner layers consisted of –5.2V and -2.0V power planes, and one ground plane. Up to 288 IC's (Integrated Circuit) could go on each board, 144 per side, and up to 72 modules could go inside one of twenty-four 28-inch high chassis. All IC's were 16-pin hermetically sealed IC's which were provided by Fairchild and Motorola. In all, there were 1,662 modules and 113 module types.

The Cray-1 used three ECL (Emitter-Coupled Logic) chip types. The register chips were 16 x 4 bit bipolar, with a 6-ns cycle time; the memory chips were 1024 x 1 bit bipolar, with a 50-ns cycle time; and the bipolar logic chips were simple 5- and 4-wide NAND gates with sub-nanosecond propagation times.

One problem that was dealt with in the Cray-1 circuit design was standing waves. Standing waves could be induced in the ground plane when a signal ran around a board. Standing waves created peaks and troughs in the signal level (which was undesirable) and also created reflections, which could cause the signal to be diminished, depending on how bad it was.

The first part of the solution was to make all the signal paths the same length, which was done by adding foil runs and additional IC's. All paths were locked down to a total length that had been tested to deliver the maximum signal strength at the destination. In this way, the lengths of the signal paths did not do away with the peaks, but rather utilized them to deliver the maximum signal. About 10 to 20 percent of all IC's were there just to pad out signal lines.

The other solution was to terminate both sides of each gate. Terminating means that the power supplies saw no dynamic components. This was done by using resistive components on both sides of each gate. Hence, reflections caused by standing waves got absorbed. This was the main reason simple gates were used. Complex gates would have made it impossible to terminate both sides of every gate.

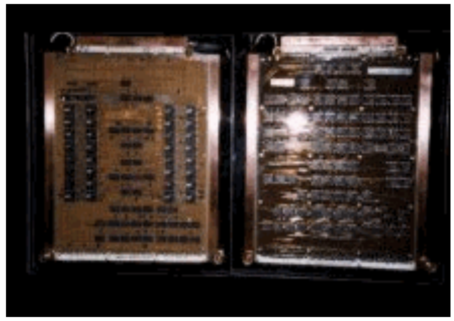

Figure D. A Cray-1 module.
(Photo from ESP Educational Support Package:
Cray Historical Archives.)

Back to OUTLINE.

Architecture: Cooling

Because such high component densities were on each module, an enormous amount of heat was generated. For the maximum memory option of 1024K, 115 KW was dissipated. Since power supplies were usually only 50 percent efficient, half of the remaining 57.5 KW was used for the memory and half for the processor and switch. With the portion going to memory scaled linearly, this meant that for the basic memory option of 256K, the power consumed/dissipated was approximately 71.87 KW.

To successfully carry away so much heat, a new cooling technology had to be developed. This technology was Freon-based but used available metal conductors in a new way. In each chassis, vertical aluminum/stainless steel bars lined each column wall. Freon was passed through a stainless steel tube within an aluminum casing. Modules were laid horizontally and stacked in tower formation, about six feet high. (See Figure E.) Flowing Freon ran from top to bottom of each tower, contacting each module along the way, thereby taking the heat away. The cooling system that was developed had a 40-ton cooling capacity. Along with power supply units, the cooling equipment was enclosed under a series of upholstered seats (see Fig. A), forming the base.

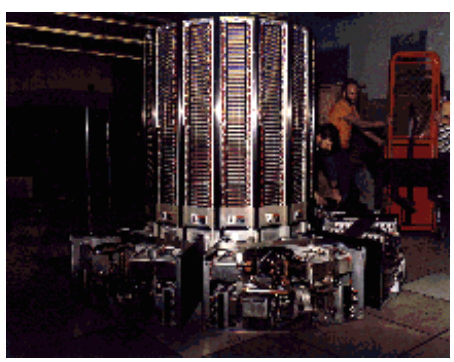

Figure E. An exposed Cray-1.
(Photo from ESP Educational Support Package:
Cray Historical Archives.)

A problem related to the cooling system was addressed by Seymour Cray in a speech he had given to prospective Cray-1 users in 1975. In it he indicated a problem with the cold bar, namely that if there was a crack in the stainless steel tubing at the bond between the tubing and the elbow, that Freon would leak through the porous aluminum casting. Freon was not so much the problem as the little bit of oil that was mixed with it, if it dripped onto the modules. This problem was later fixed by changing the casting and/or the welding technology.

Back to OUTLINE.

Maintenance

The Cray-1 required extensive maintenance. Initially, MTBF was on the order of 50 hours. MTBF is Mean Time Between Failures, and in this case, it was the average time the Cray-1 worked without any failures. Two hours of everyday was typically set aside for preventive maintenance. Additional time was devoted to maintenance during the weekends. Even with such extensive maintenance, the Cray-1 (when a 75 MHz clock rate was run) still yielded more results than other computers running full time.

Back to OUTLINE.

CPU Characteristics: Functional Units

The Cray-1 had 12 functional units (pipeline processors). A key feature of a pipeline register/processor was that it allowed an instruction to be fetched in parallel with the data operation—i.e. while one instruction was being executed, the next instruction was being fetched. What's the advantage? Speed. CPU dead time was minimized.

The 12 units were divided into four groups: address, scalar, vector, and floating point. Table 1 lists the 12 functional units, along with which registers they used and how many time units, in clock periods, was required. Notice that there was no divide unit. A method of reciprocal approximation followed by a multiply was used instead, allowing a more efficient pipeline implementation.

Table 1. The 12 Functional Units

Functional Unit	Register Usage	Unit Time (in clock periods)
Address Address add unit Address multiply unit	 A A	 2 6
Scalar Units Scalar add unit Scalar shift unit Scalar logical unit Population/leading 0 count	 S S S S	 3 2 or 3 1 3
Vector Vector add unit	 V	 3

Vector shift unit	V	4
Vector logical unit	V	2
Floating-point (F-p)		
F-p add unit	S, V	6
F-p multiply unit	S, V	7
Reciprocal approximation unit	S, V	14

Back to OUTLINE.

Figure F shows the Cray-1's computation section. Register paths are shown in detail. The figure also shows the instruction buffers with the associated registers: NIP, for next instruction parcel; CIP, for current instruction parcel; and LIP, for last instruction parcel. Operating and supporting registers are described below.

CPU Characteristics: Operating Registers

There were five types of registers that the Cray-1 employed. Three were primary registers: A, S, and V; two were intermediate or temporary storage registers: B, and T.

A Registers: There were eight 24-bit A registers. Their primary functions were as address registers for memory references, index registers, and also provide values for shift counts, loop control, and channel I/O operations. In address applications, A registers were used to index base addresses for scalar memory references. They also provided a base address and an index address for vector memory references. A registers were designated A0 through A7.

B Registers: There were sixty-four 24-bit B registers. They were used as auxiliary or temporary storage for the A registers. Only one clock period was required for the transfer of an operand between an A and a B register. B registers typically contained addresses and counters that were referenced over a longer period of time than would have been allowed in the A registers. $B00_8$ through $B77_8$ designated the B registers. (Base 10 equivalent designation would be B00 – B63.)

S Registers: Eight 64-bit registers made up this register set. They were the main data handling registers for scalar operations and served as both source and destination registers for scalar arithmetic and logical instructions. Scalar values used in vector operations were held in S registers. Operations performed on S register data were logical, shift, fixed-point, and floating-point operations. S0 through S7 designated the S registers.

T Registers: Like the B registers, there were sixty-four 24-bit T registers which were used as auxiliary or temporary storage for the S registers. Also, similar to the B registers, only one clock period was needed for the transfer of an operand between an S register and a T register. T registers normally contained operand values that were referenced over a longer period of time than normally allowed in the S registers. T registers allowed intermediate results to be held in intermediate access storage rather than use memory. One word per clock period was the transfer rate of a block of data in the T registers to or from memory. T registers were designated $T00_8$ through $T77_8$ (or T00 through T63 in base 10).

V Registers: There were eight 64-element V registers, which provided operands to and received results from the functional units at a rate of one clock period. Each of the 64 elements contained a 64-bit quantity. When data were grouped in a successive fashion, the register could have been considered to contain a vector. A vector was simply a row and column of values, as in a matrix or a table. When each vector element was processed in the same way, computational efficiency was achieved. V Registers were designated V0 through V7.

Back to OUTLINE.

CPU Characteristics: Supporting Registers

Besides the operating registers, the CPU also had supporting registers that supported control of program execution. Supporting registers were VL (Vector Length), VM (Vector Mask), P (Program counter), BA (Base Address), LA (Limit Address), XA (eXchange Address), F (Flag), and M (Mode) registers.

VL Register: The VL (Vector Length) register contained the number of vector register elements to be processed.

VM Register: The VM (Vector Mask) register was a 64-bit register that controlled the designation of vector elements in vector merge and test instructions, with each bit corresponding to a vector register element. By testing each V register element for a specific condition, in the vector test instruction, the VM content was defined.

P Register: The P (Program counter) register was a 24-bit register, whose upper 22 bits specified a memory address and the lower 2 bits specified the parcel number. As each instruction parcel in a non-branching sequence was executed, the parcel address was incremented by one and replaced whenever branching occurred in a program.

BA Register: The BA (Base Address) register was an 18-bit register that contained the upper 18 bits of a 22-bit memory address. The lower 4 bits of this address were zeros. A process called "exchange sequence" stored the upper 18 bits of the lowest memory address to be referenced during program execution into the BA register just before initial or continued program execution. As the program ran, the address portion of each instruction referencing memory had its content added to the BA register. The sum became the absolute address used for memory reference. Memory addresses lower than what was in the BA register were then ensured of not being accessed. Therefore, programs must have had all instructions that made memory references use relative addresses to make those references. This process supported memory protection. The production of an absolute address did not affect the contents of the BA register, instruction buffer, or memory.

LA Register: The LA (Limit Address) register was an 18-bit register that contained the upper 18 bits of a 22-bit memory address. The lower 4-bits, just as with the BA register, contained zeros. The "exchange sequence" process stored into the LA register the upper 18 bits of the absolute address one higher than allowed to be referenced by the program just before initial or continued program execution. When the program started, each instruction referencing a memory location had the absolute address for that reference. (See BA Register above.) The absolute address was then compared against the LA register content. If the absolute address was greater than or equal to the limit address register content, an out-of-range error was flagged and the program ended. This process also supported the memory protection operation.

XA Register: This eXchange Address register held the upper 8 bits of a 12-bit memory address, whose lower 4 bits were zeros. The XA register could reference only every 16^{th} memory address, from 0000 through 4080, because only 12 bits were used with the lower 4 bits being zeros. Each of these memory addresses designated the first word of a 16-word set. So 256 sets, 16 memory words each, could be specified. The XA register held the first memory address of a 16-word set or exchange package just before initial or continued program execution. Certain operating and support registers' contents were contained in the exchange package as needed for operations following an interrupt.

F Register: The F (Flag) register was a 9-bit register that had flags that indicated interrupt conditions whenever they were set, which then caused the initiation of an exchange sequence. There were nine interrupt conditions: normal exit, error exit, I/O interrupt, uncorrected memory error, program range error, operand range error, floating-point overflow, real-time clock interrupt, and console interrupt.

M Register: The M (Mode) register was a 3-bit register that had part of an exchange package for a presently active program. During an exchange sequence, the three bits were selectively set. The three mode flags were floating-point error mode flag, which when set enabled interrupts on floating-point errors; uncorrectable memory error mode flag, which when set enabled interrupts on uncorrectable memory parity errors; and monitor mode flag, which when set inhibited all interrupts other than parity errors.

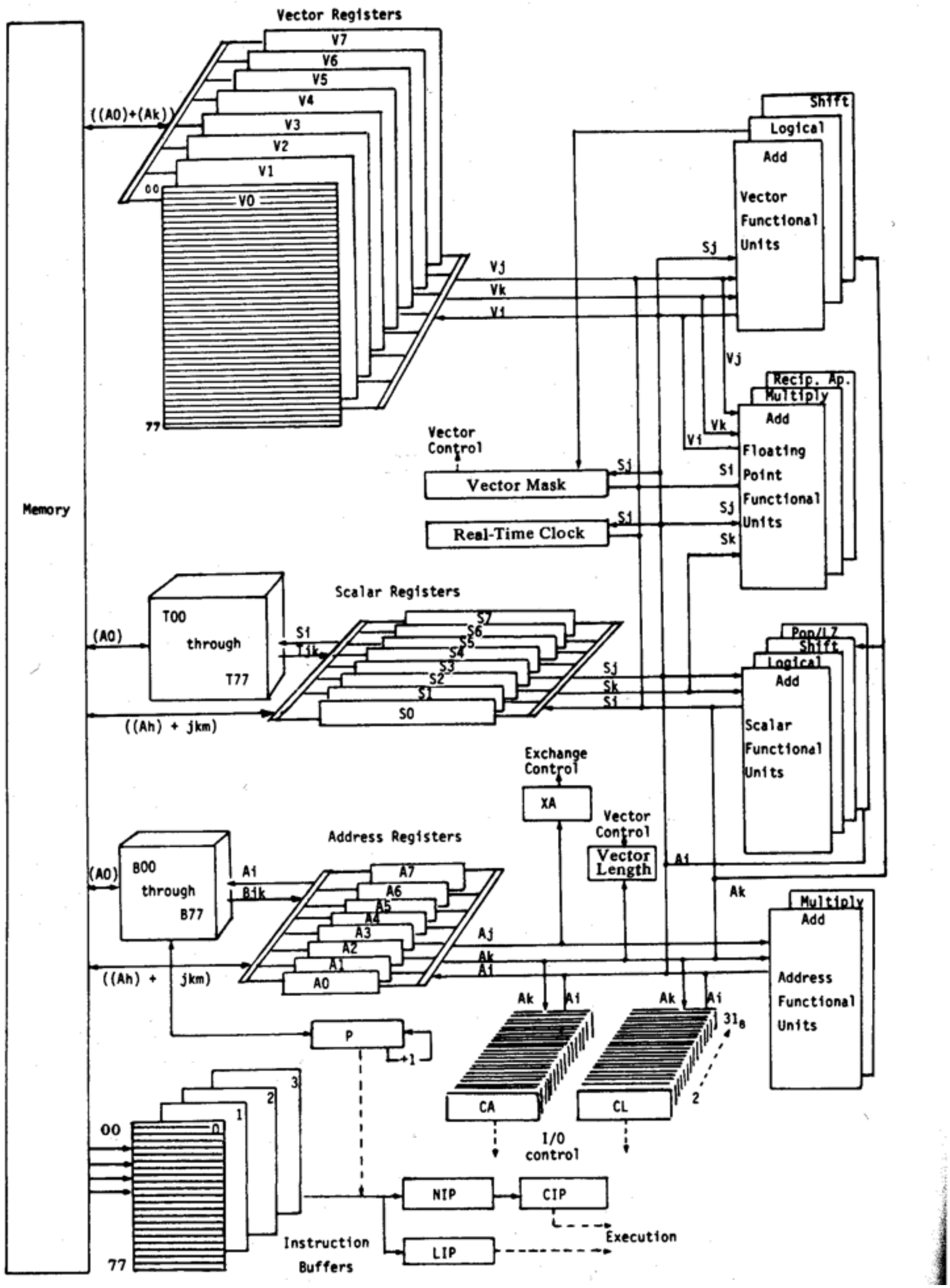
Vector Registers
V7
V6
V5
V4
V3
V2
V1
V0
00
77
((A0)+(Ak))
Memory
Vj
Vk
Vi
Shift
Logical
Add
Vector Functional Units
Sj
Recip. Ap.
Multiply
Add
Floating Point Functional Units
Si
Sk
Vector Control
Vector Mask
Real-Time Clock
T00 through T77
(A0)
Si
Tjk
Scalar Registers
S7
S6
S5
S4
S3
S2
S1
S0
((Ah) + jkm)
Pop/LZ
Scalar Functional Units
Exchange Control
XA
Vector Control
Vector Length
Address Registers
B00 through B77
Ai
Bjk
A7
A6
A5
A4
A3
A2
A1
A0
Aj
Ak
Multiply
Address Functional Units
P
+1
CA
CL
31_8
2
I/O control
NIP
CIP
LIP
Execution
Instruction Buffers
0
1
2
3

Figure F. The Cray-1 computation section. (From Gorsline.)

Back to OUTLINE.

CPU Characteristics: Integer & Floating-Point Arithmetic

Integer arithmetic was done in 24-bit or 64-bit 2's complement. Floating-point numbers were represented in signed magnitude. The format was a packed, signed binary fraction-- represented by a 49-bit signed magnitude value--and a 15-bit biased binary integer exponent. The unbiased exponent range was approximately 10^{-2500} to 10^{+2500}. An exponent outside this range would have caused an underflow or overflow error, if floating-point interrupts were enabled.

Back to OUTLINE.

CPU Characteristics: Vector Operations, Chaining

To make efficient use of the Cray-1's pipelined feature, programs had to be structured to take advantage of the available vector processing capabilities. The vectorization process could be done either manually by the programmer or automatically by a vectorizing compiler. The Fortran programming language was typically used. Vectorization involved organizing a Fortran program so that during compilation, the Fortran statements could be mapped directly into long vector instructions.

For example, take the following Fortran statements:

DO 10 I = 1, N, 3
Z(I) = X(I) * Y(I)

The above says that a scalar multiplication is to be done with index I = 1 to N, in increments of 3. This corresponds to a vector multiplication Z¬ X ´ Y. If an appropriate vector multiply is in the instruction repertoire (or library), then the scalar multiply Z(I) = X(I) * Y(I) can be replaced by a single vector multiply instruction. This is almost like overloading the multiply operator in the C++ language so that it knows that a vector multiply is to be used rather than the conventional scalar multiply. Do-loops like the one above, were one major means of specifying vector instructions in Fortran.

Parallel operations on vectors allowed two or more results per clock period to be generated. Vector operations used either two vector registers as sources of operands or one scalar register and one vector register. Vectors could have a maximum of 64 elements. If this was exceeded, they were processed in 64-element segments. A graphical representation of vector operations is shown in Figure G.

Chaining was another technique used by the Cray-1. Chaining occurred when results from one functional unit were directly and immediately fed into another functional unit (and so on). Intermediate results did not have to be stored in memory and could be used before the operation that created them finished. Figure H shows a graphical representation of chaining.

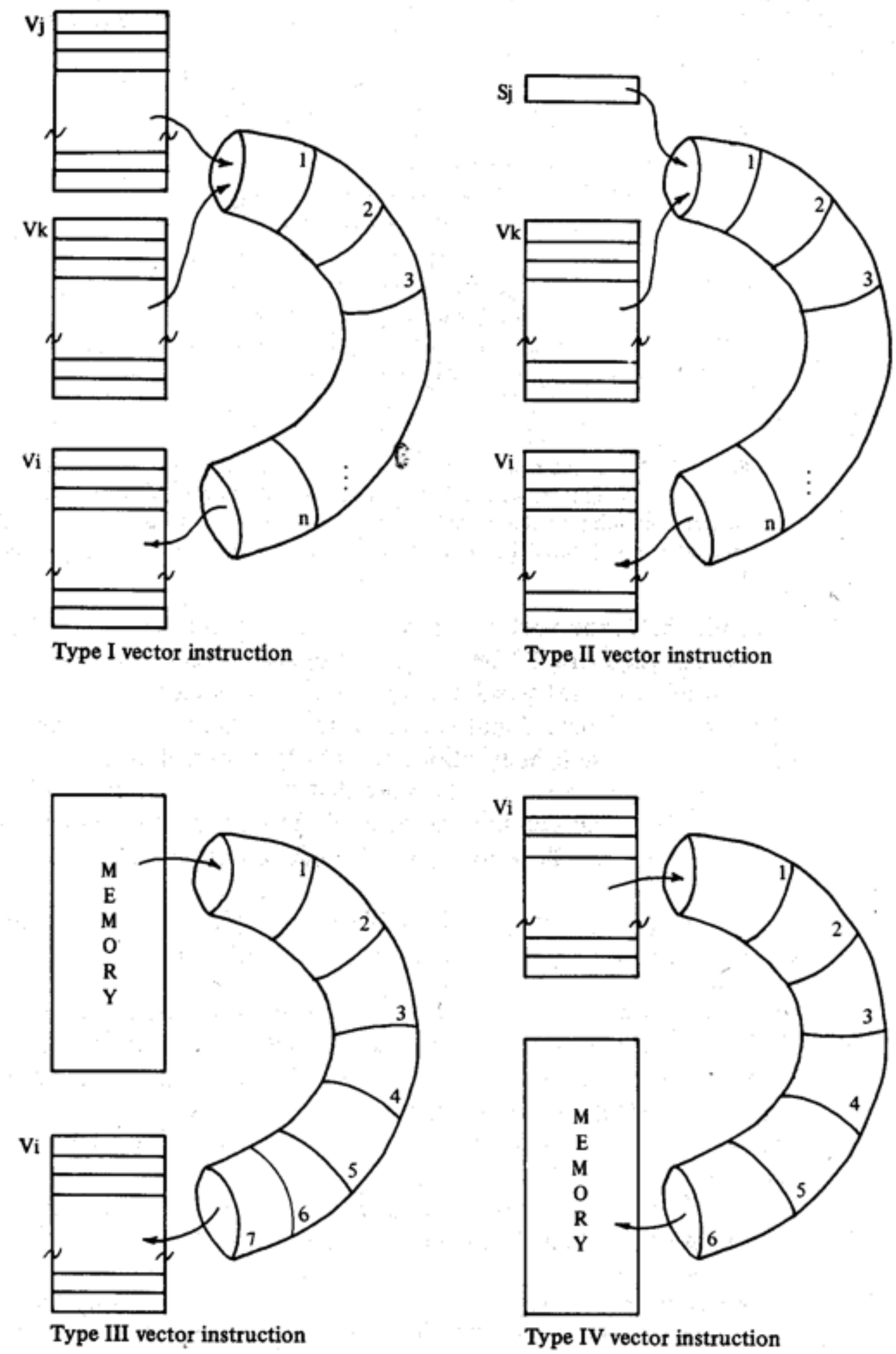

Figure G. A graphical representation of the vector operations in the Cray-1. A type I vector operation referred to a register-to-register operation, with a 16-bit instruction; both operands were vectors. Type II was the same as type I but with scalar and vector operands. Types III and IV referred to a load and store operation, requiring a 32-bit instruction. (From Gorsline.)

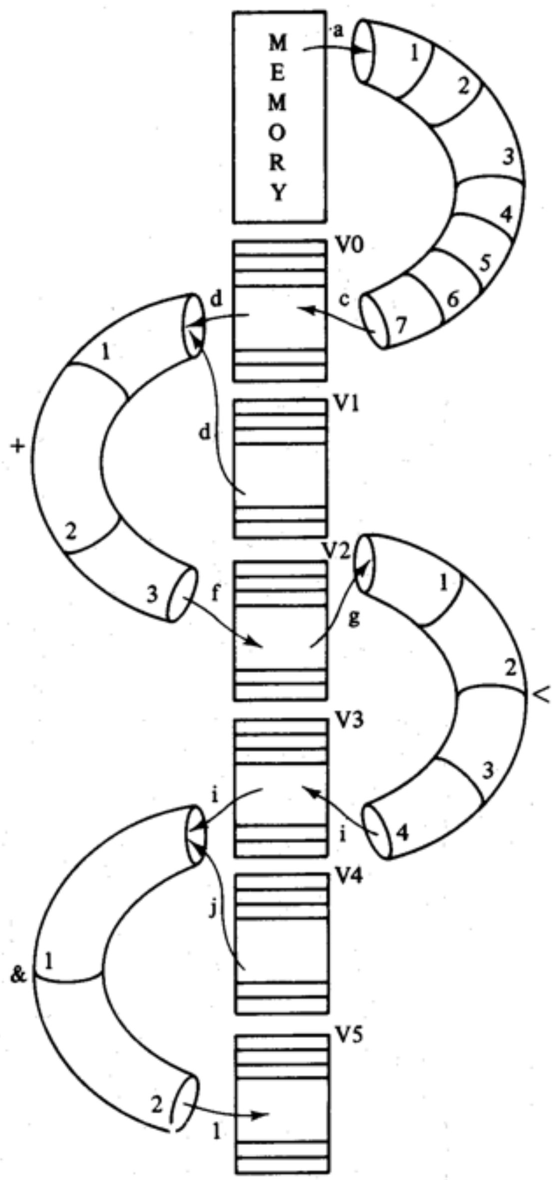

Figure H. A graphical representation of chaining. A LOAD instruction was followed by a type I arithmetic instruction, then by a MOVE instruction and finally by another type I arithmetic instruction. See also Figure G. (From Gorsline.)

Back to OUTLINE.

CPU Characteristics: Interrupts

When an interrupt condition was detected, instruction issue was killed by the hardware. However, all memory bank activity was allowed to finish as were any vector instructions that were still running. An exchange sequence was then activated. During an exchange sequence, the cause of an interrupt was analyzed, and all interrupts were eventually processed.

Back to OUTLINE.

CPU Characteristics: Memory

The Cray-1's memory section was organized in 16 banks, with 72 modules per bank, Each module was a 1-bit contribution to a 64-bit word. The remaining modules were used for an 8-bit check byte, which was required for SECDED, Single-bit Error Correction, Double-bit Error Detection. Other memory features are listed in the section titled Architecture: Physical Organization of the Cray-1.

Back to OUTLINE.

I/O

A front-end computer was required with the Cray-1, as it was not designed to be a stand-alone computer. A Data General S/200 minicomputer was used as the console processor, which allowed operator interfacing, maintenance, and execution monitoring. A General Eclipse minicomputer was also used, and was replaced in 1978 by Cray Research's own minicomputer with an "A" processor, a 16-bit and 80 million instructions per second (MIPS) machine. The Cray-1 could be attached to almost any mini- or maxicomputer for front-end control and I/O. A "native-mode" I/O to mass storage was also available on the Cray-1.

Back to OUTLINE.

Software

By today's standards, the Cray-1 had a very limited set of application software. However, it was sufficient for the tasks it was designed to tackle. Its software set included the Cray Operating System (COS), the Cray Fortran compiler (CFT), and the Cray Assembler Language (CAL).

The Cray Operating System was a batch operating system able to support up to 63 jobs. It was designed to be the recipient of job requests and data files from front-end computers.

The CFT was an optimizing Fortran compiler designed to compile ANSI 66 Fortran IV. Although most of the functions in the Cray library were vectorizable, the Cray Fortran compiler had its limitations. Loops that contained branches such as GO TO's, IF's, or CALL statements were not vectorized. However, loops could contain function references if the function had a vector version. Later versions of the CFT did away with these limitations.

The Cray Assembler Language was a macro assembler and an overlay loader. It had a utilities set that included a text editor and some debugging features.

Later machines also ran the UNICOS, Cray's version of UNIX. The Cray-2 was the first to use it, and was in use through the T90 family.

Back to OUTLINE.

Conclusion

Although the Cray-1 has long been surpassed by desktop computers that are now commonplace in businesses and in our homes, it remains as one of the most innovative and impressive systems ever invented. In all, 61 units were shipped, with the last unit shipped in 1984.

Back to OUTLINE.

BIBLIOGRAPHY

Baer, Jean-Loup. Computer Systems Architecture. Potomac, Maryland: Computer Science Press, 1980.

Distinguished Lecture Series, Vol. II video. . "Seymour Cray. What's All This About Gallium Arsenide?" Sponsors: Cray Research, Inc., Lawrence Livermore National Laboratory, Los Alamos National Laboratory. Orlando, Florida: Supercomputing '88 Conference, 15 Nov. 1988.

Gorsline, G.W. Computer Organization: Hardware/Software. Englewood Cliffs, NJ: Prentice Hall, 1986.

Hayes, John P. Computer Architecture and Organization. 2nd ed. New York: McGraw-Hill, 1988.

Huppenthal, Jon. SRC Computers. Personal correspondence: May, 2002. [hupp@srccomp.com].

Murphy, R. Sean. "Physical dimensions of the Cray-1." [sean_murphy@agilent.com]. 2002.

Murray, Charles J. The Supermen: The Story of Seymour Cray and the Technical Wizards behind the Supercomputer. New York: John Wiley & Sons, 1997.

Ramsey, Dr. Dennis J. ESP Educational Support Package: Cray Historical Archives. [http:www.tmeg.com/esp/c_cray/cray.htm]. 1998.

Siewiorek, Daniel P., C. Gordon Bell, Allen Newell. Computer Structures: Principles and Examples. New York: McGraw-Hill, 1982.

Wikepedia. [http://www.wikepedia.com]. 2002.

Back to OUTLINE.