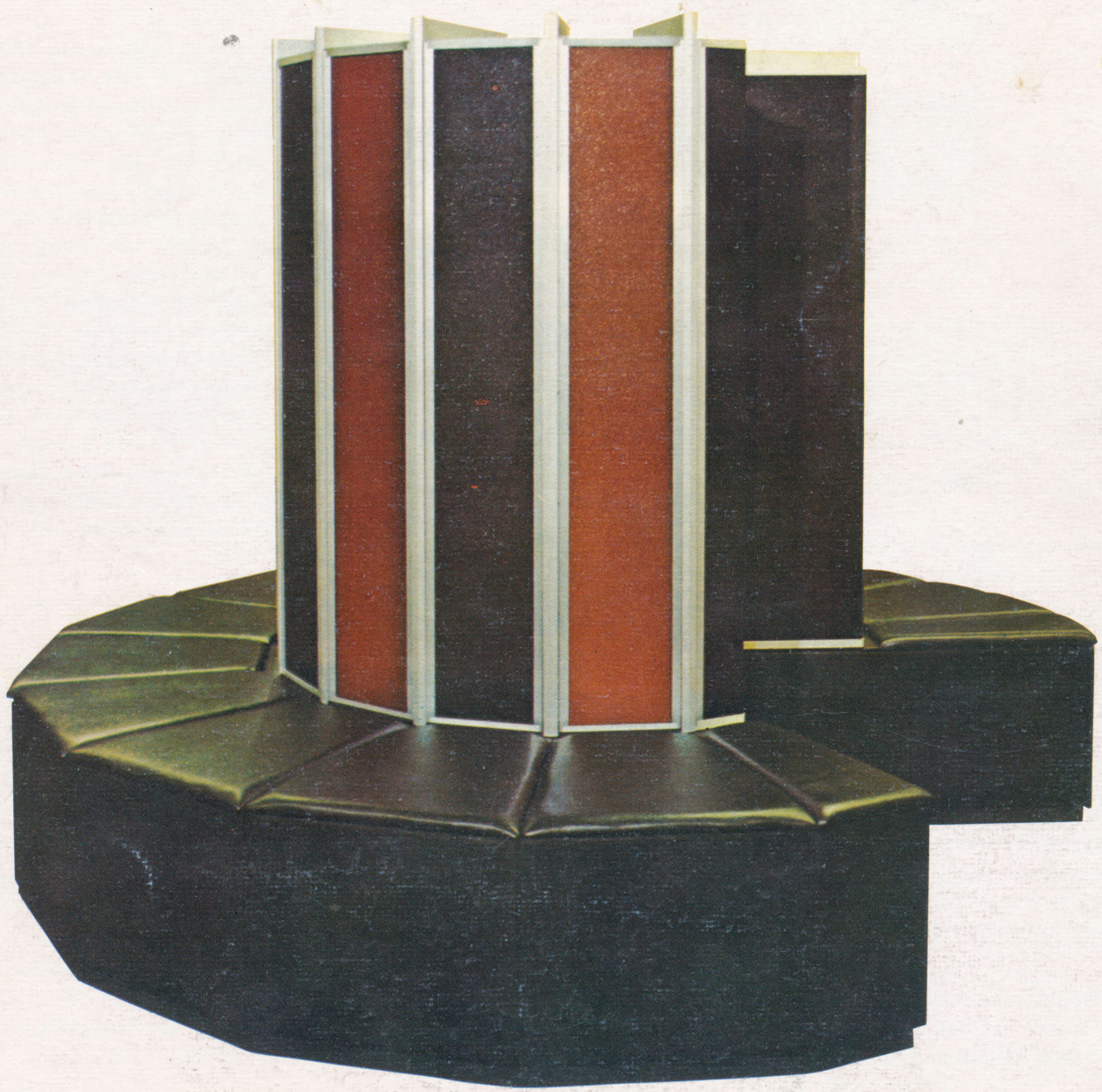
CRAY RESEARCH, INC.

THE CRAY-1 COMPUTER SYSTEM



THE CRAY-1® COMPUTER SYSTEM

The Cray Research, Inc. CRAY-1 Computer System is a large-scale, general-purpose digital computer featuring vector as well as scalar processing, a 12.5 nanosecond clock period, and a 50 nanosecond memory cycle time. The CRAY-1 is capable of executing over 80 million floating point operations per second. Even higher rates are possible with programs that take advantage of the vector features of the computer.

The CRAY-1 is an effective host processor for local computer networks and time-sharing networks. Cray Research provides hardware and software interfaces that link the CRAY-1 with other manufacturer's computer systems. This approach allows users to add the CRAY-1 to existing computing facilities.

The CRAY-1 is particularly adapted to the needs of the scientific community and is especially useful in solving problems requiring the analysis and prediction of the behavior of physical phenomena through computer simulation. The fields of weather forecasting, aircraft design, nuclear research, geophysical research, and seismic analysis involve this process. For example, the movements of global air masses for weather forecasting, air flows over wing and airframe surfaces for aircraft design, and the movements of particles for nuclear research, all lend themselves to such simulations. In each scientific field, the equations are known but the solutions require extensive computations involving large quantities of data. The quality of a solution depends heavily on the number of data points that can be considered and the number of computations that can be performed. The CRAY-1 provides substantial increases with respect to both the number of data points and computations so that researchers can apply the CRAY-1 to problems not feasibly solvable in the past.

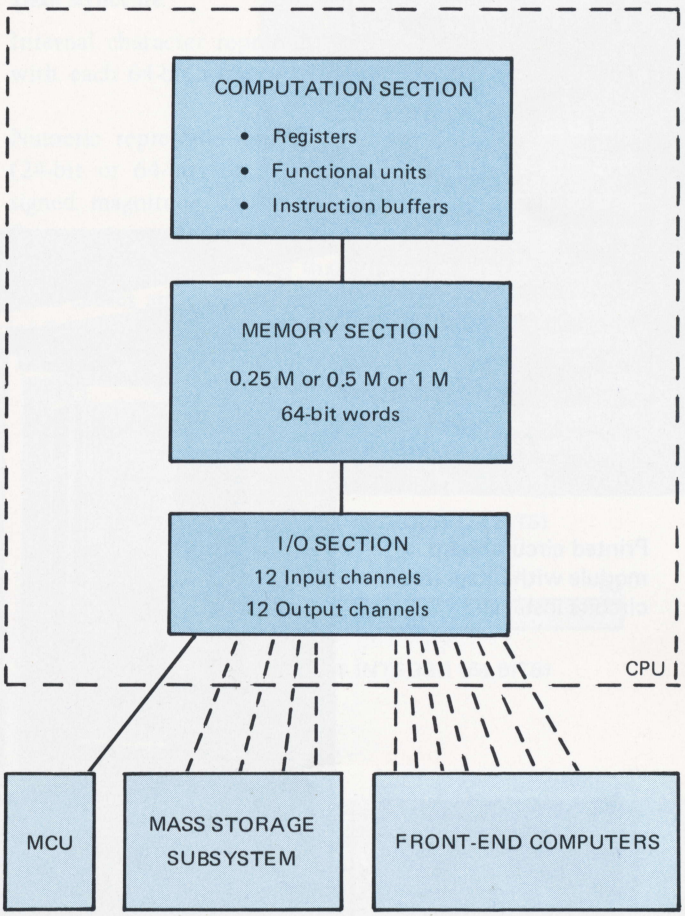
CONFIGURATION

The basic configuration of the CRAY-1 consists of the central processor unit (CPU), power and cooling equipment, one or more minicomputer consoles, and a mass storage (disk) subsystem. The CPU holds the computation, memory, and I/O sections of the computer. A minicomputer serves either as a maintenance control unit or a job entry station.

INPUT/OUTPUT

Input/output is via twenty-four I/O channels, twelve of which are input and twelve output. Any number of channels may be active at a given time. For a 16-bit channel, transfer rates of 160 million bits per second can be achieved. Higher rates are possible but in practice, the maximum transfer rate is limited by the speed of peripheral devices.

Publication Number 2240008 B



BASIC COMPUTER SYSTEM

Input/output and the CPU share memory access via a single port. I/O priority is sufficient to ensure maintaining the required transfer rates for the peripheral devices.

MEMORY

The CRAY-1 memory is constructed of 1024-bit LSI chips. Up to 1,048,576 (generally referred to as one million) 72-bit words are arranged in 16 banks. Up to 524,288 words can be arranged in 8 banks. A word consists of 64 data bits and 8 check bits. The bank cycle time, that is, the time required to remove or insert an element of data in memory, is 50 nanoseconds. This short cycle time provides an extremely efficient random-access memory. There is no inherent memory degradation for 16-bank memories with less than one million words of memory.

A single-error-correction double error detection (SECDED) network assures that data written into memory can be returned to the computation section with consistent precision. There are eight check bits per memory word.

FACTS AND FIGURES

CPU	
Instruction size	16 or 32 bits
Repertoire size	128 instruction codes
Clock period	12.5 nsec
Instruction stack/buffers	64 words (4096 bits)
Functional units	twelve: 3 integer add 1 integer multiply 2 shift 2 logical 1 floating add 1 floating multiply 1 reciprocal approx. 1 population count
Programmable registers	8x64 64-bit 73 64-bit 72 24-bit 1 7-bit
Max. vector result rate	12.5 nsec / unit
FLOATING POINT COMPUTATION RATES (results per second)	
Addition	80×10^6 / sec
Multiplication	80×10^6 / sec
Division	25×10^6 / sec
MEMORY	
Technology	bipolar semiconductor
Word length	72 bits (64 data, 8 SECDED)
Address space	4M words
Data path width (bits)	64 (1 word)
Cycle time	50 nsec.
Size	262,144 words or 524,288 words or 1,048,576 words
Organization / interleave	16 banks (8 banks optional)
Maximum band width	80×10^6 words / sec (5.1×10^9 bits / sec)
Error checking	SECDED
PHYSICAL CHARACTERISTICS / ELECTRONIC TECHNOLOGY	
Size of CPU cabinet	9 ft diameter base 4.5 ft diameter center 6.5 ft height
Weight of mainframe	5.25 tons
Cooling	Freon
Plug-in modules	1662
Module types	113
PC boards	5 layer
Circuitry (equivalent no. of transistors)	2.5M
Logic	ECL, 1 nsec.
High-density logic	SSI

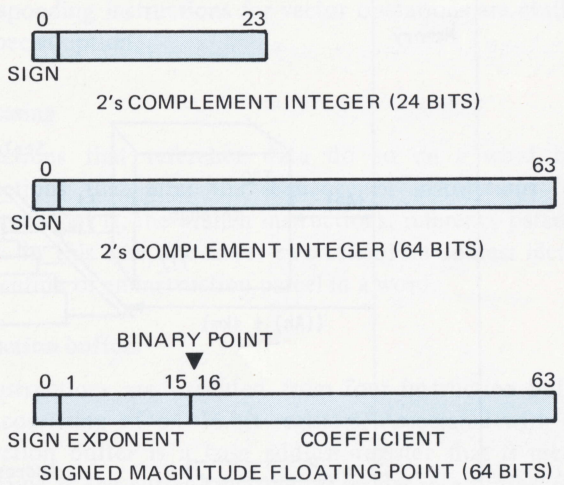
COMPUTATION SECTION

The computation section as illustrated on page 4 is composed of instruction buffers, registers, and functional units which operate together to execute sequences of instructions.

Data structure

Internal character representation in the CRAY-1 is in ASCII with each 64-bit word able to accommodate eight characters.

Numeric representation is either in two's complement form (24-bit or 64-bit) or in 64-bit floating point form using a signed magnitude binary coefficient and a biased exponent. Exponent overflow and underflow is caused if the exponent is greater than 57777_8 or less than 20000_8. Either of these conditions causes an interrupt except where the interrupt has been inhibited.



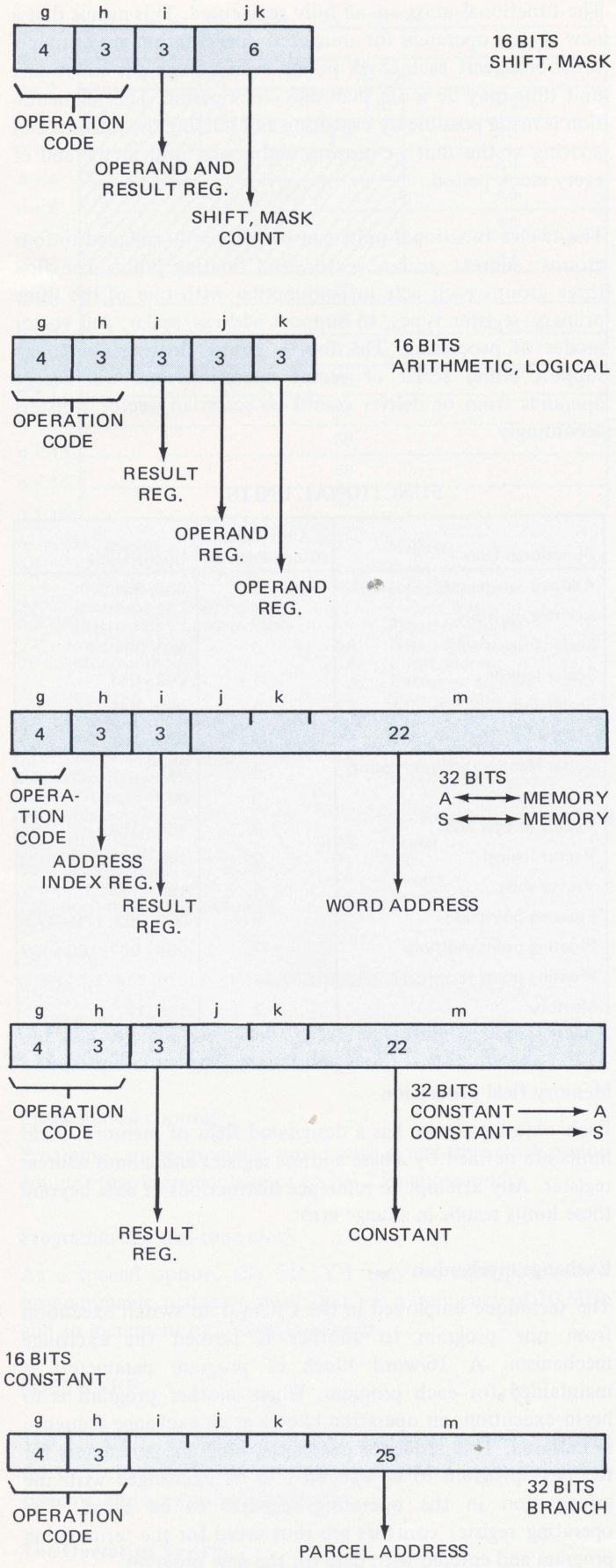
DATA FORMATS

Instruction set

The CRAY-1 executes 128 operation codes as either 16-bit (one parcel) or 32-bit (two-parcel) instructions. Operation codes provide for both scalar and vector processing.

In general, an instruction that references registers occupies one parcel; an instruction that references memory occupies two parcels. All of the arithmetic and logical instructions reference registers.

Floating point instructions provide for addition, subtraction, multiplication, and reciprocal approximation. The reciprocal approximation instruction allows for the computation of a floating point divide operation using a multiple instruction sequence.



Integer or fixed point operations are provided for as follows: integer addition, integer subtraction, and integer multiplication. An integer multiply operation produces a 24-bit result; additions and subtractions produce either 24-bit or 64-bit results. No integer divide instruction is provided. The operation can be accomplished through a software algorithm using floating point hardware.

The instruction set includes Boolean operations for OR, AND, and exclusive OR and for a mask-controlled merge operation. Shift operations allow the manipulation of 64- or 128-bit operands to produce a 64-bit result. Similar 64-bit arithmetic capability is provided for both scalar and vector processing. Full indexing capability allows the programmer to index throughout memory in either scalar or vector modes of processing. This allows matrix operations in vector mode to be performed on rows, on columns, or on the diagonal.

Instructions for scalar population and leading zero counts return bit counts based on S register contents to an A register. Corresponding instructions for vector operations are available as a special option.

Addressing

Instructions that reference data do so on a word basis. Instructions that alter the sequence of instructions being executed, that is, the branch instructions, reference parcels of words. In this case, the lower two bits of an address identify the location of an instruction parcel in a word.

Instruction buffers

All instructions are executed from four instruction buffers, each consisting of 64 16-bit registers. Associated with each instruction buffer is a base address register that is used to determine if the current instruction resides in a buffer. Since the four instruction buffers are large, substantial program segments can reside in them. Forward and backward branching within the buffers is possible and the program segments may be noncontiguous. When the current instruction does not reside in a buffer, one of the instruction buffers is filled from memory. Four memory words are transferred per clock period. The buffer that is filled is the one least recently filled, that is, the buffers are filled in rotation. To allow the current instruction to issue as soon as possible, the memory word containing the current instruction is among the first four transferred. A parcel counter register (P) points to the next parcel to exit from the buffers. Prior to issue, instruction parcels may be held in the next instruction parcel (NIP), lower instruction parcel (LIP) and current instruction parcel (CIP) registers.

Operating registers

The CRAY-1 has five sets of registers, three primary and two intermediate. Primary registers can be accessed directly by functional units. Intermediate registers are not accessible by functional units but act as buffers between primary registers and memory.

The figure on page 4 represents the CRAY-1 registers and functional units. The 64 address and 64 scalar intermediate registers can be filled by block transfers from memory. Their purpose is to reduce memory references made by the scalar and address registers.

The eight address registers are each 24 bits and can be used to count loops, provide shift counts, and act as index registers in addition to their main use for memory references.

The eight 64-bit scalar registers in addition to contributing operands and receiving results for scalar operations can provide one operand for vector operations.

Each of the eight vector (V) registers is actually a set of 64 64-bit registers, called elements. The number of vector operations to be performed (that is, the vector length) determines how many of the elements of a register are used to supply operands in a vector set or receive results of the vector operation. The hardware accommodates vectors with lengths up to 64; longer vectors are handled by the software dividing the vector into 64-element segments and a remainder.

Associated with the vector registers are a 7-bit vector length register and a 64-bit vector mask register. The vector length register, as its name implies, determines the number of operations performed by a vector instruction. Each bit of the vector mask register corresponds to an element of a V register. The mask is used with vector merge and test instructions to allow operations to be performed on individual vector elements.

Supporting registers

In addition to the operating registers, the CPU contains a variety of auxiliary and control registers. For example, there is a channel address (CA) register and a channel limit register (CL) for each I/O channel.

Functional units

Instructions other than simple transmits or control operations are performed by hardware organizations known as functional units. Each of the twelve units in the CRAY-1 executes an algorithm or a portion of the instruction set. Units are independent. A number of functional units can be in operation at one time.

A functional unit receives operands from registers and delivers the result to a register when the function has been performed. The units operate essentially in three-address mode with source and destination addressing limited to register designators.

All functional units perform their algorithms in a fixed amount of time. No delays are possible once the operands have been delivered to the unit. The amount of time required from delivery of the operands to the unit to the completion of the calculation is termed the "functional unit time" and is measured in 12.5 nsec clock periods.

The functional units are all fully segmented. This means that a new set of operands for unrelated computation may enter a functional unit each clock period even though the functional unit time may be more than one clock period. This segmentation is made possible by capturing and holding the information arriving at the unit or moving within the unit at the end of every clock period.

The twelve functional units can be arbitrarily assigned to four groups: address, scalar, vector, and floating point. The first three groups each acts in conjunction with one of the three primary register types, to support address, scalar, and vector modes of processing. The fourth group, floating point, can support either scalar or vector operations and will accept operands from or deliver results to scalar or vector registers accordingly.

FUNCTIONAL UNITS

Functional Unit	Unit Time (Clock Periods)	Instructions
Address integer add	2	030, 031
Address multiply	6	032
Scalar integer add	3	060, 061
Scalar logical	1	042 - 051
Scalar shift	2	052 - 055
	3	056, 057
Scalar leading zero/pop count	4	026
	3	027
Vector integer add	3	154 - 157
Vector logical	2	140 - 147, 175
Vector shift	4	150 - 153
Floating point add	6	062, 063, 170 - 173
Floating point multiply	7	064 - 067, 160 - 167
Floating point reciprocal	14	070, 174
Memory	7	176, 177

Memory field protection

Each object program has a designated field of memory. Field limits are defined by a base address register and a limit address register. Any attempt to reference instructions or data beyond these limits results in a range error.

Exchange mechanism

The technique employed in the CRAY-1 to switch execution from one program to another is termed the exchange mechanism. A 16-word block of program parameters is maintained for each program. When another program is to begin execution, an operation known as an exchange sequence is initiated. This sequence causes the program parameters for the next program to be executed to be exchanged with the information in the operating registers to be saved. The operating register contents are thus saved for the terminating program and entered with data for the new program.

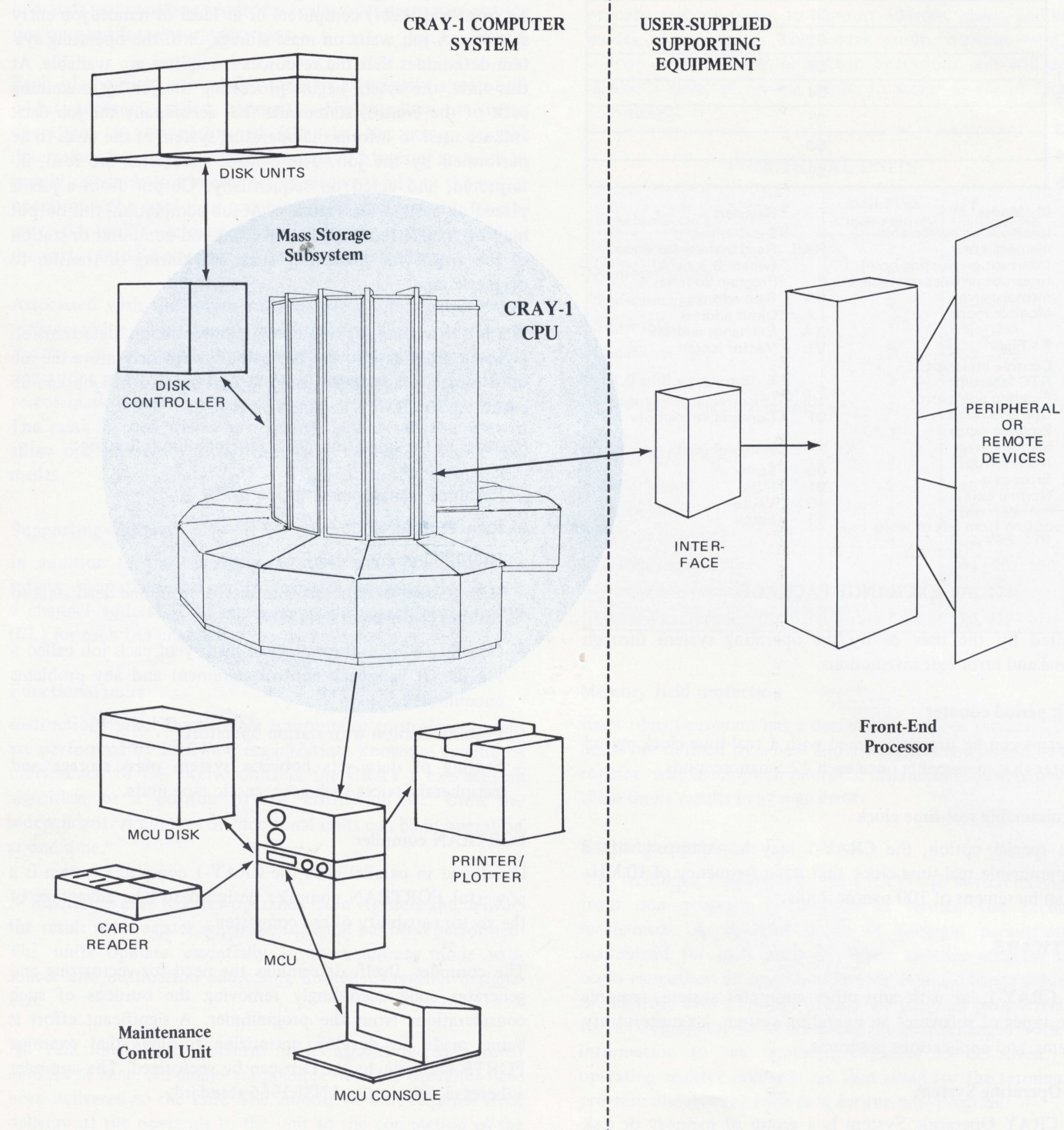
Exchange sequences may be initiated automatically upon occurrence of an interrupt condition or may be voluntarily

means of expressing symbolically all hardware functions of the CPU. Augmenting the instruction repertoire is a set of versatile pseudo instructions that provide users with options for generating macro instructions, organizing programs, and so on.

CAL enables the user to tailor programs to the architecture of the CRAY-1. The operating system as well as most other software provided by Cray Research, Inc. is coded in CAL assembly language.

Source program maintenance

An UPDATE program provides a means of maintaining source language programs as card images on mass storage or on magnetic tape. The user converts the source language program to a data set called a library in which each of the original cards is assigned a number. Later, when the user wishes to modify the card, it is referenced by its number and is accordingly deleted, replaced, or used for an insertion point of additional



DATA FLOW THROUGH SYSTEM

CRAY-1 INSTRUCTION SET

CRAY-1	CAL		DESCRIPTION
000xxx	ERR		Error exit
†000ijk	ERR	exp	Error exit
0010jk	CA,Aj	Ak	Set the channel (Aj) current address to (Ak) and begin the I/O sequence
0011jk	CL,Aj	Ak	Set the channel (Aj) limit address to (Ak)
0012jx	CI,Aj		Clear channel (Aj) interrupt flag
0013jx	XA	Aj	Enter XA register with (Aj)
0014jx	RT	Sj	Enter real-time clock register with (Sj)
0020xk	VL	Ak	Transmit (Ak) to VL register
†0020x0	VL	1	Transmit 1 to VL register
0021xx	EFI		Enable interrupt on floating point error
0022xx	DFI		Disable interrupt on floating point error
003xjx	VM	Sj	Transmit (Sj) to VM register
†003x0x	VM	0	Clear VM register
004xxx	EX		Normal exit
†004ijk	EX	exp	Normal exit
005xjk	J	Bjk	Jump to (Bjk)
006ijkm	J	exp	Jump to exp
007ijkm	R	exp	Return jump to exp; set B00 to P
010ijkm	JAZ	exp	Branch to exp if (A0) = 0
011ijkm	JAN	exp	Branch to exp if (A0) ≠ 0
012ijkm	JAP	exp	Branch to exp if (A0) positive
013ijkm	JAM	exp	Branch to exp if (A0) negative
014ijkm	JSZ	exp	Branch to exp if (S0) = 0
015ijkm	JSN	exp	Branch to exp if (S0) ≠ 0
016ijkm	JSP	exp	Branch to exp if (S0) positive
017ijkm	JSM	exp	Branch to exp if (S0) negative
020ijkm	Ai	exp	Transmit exp = jkm to Ai
021ijkm	Ai	exp	Transmit exp = 1's complement of jkm to Ai
022ijk	Ai	exp	Transmit exp = jk to Ai
023ijx	Ai	Sj	Transmit (Sj) to Ai
024ijk	Ai	Bjk	Transmit (Bjk) to Ai
025ijk	Bjk	Ai	Transmit (Ai) to Bjk
026ijx	Ai	PSj	Population count of (Sj) to Ai
027ijx	Ai	ZSj	Leading zero count of (Sj) to Ai
030ijk	Ai	Aj+Ak	Integer sum of (Aj) and (Ak) to Ai
†030i0k	Ai	Ak	Transmit (Ak) to Ai
†030ij0	Ai	Aj+1	Integer sum of (Aj) and 1 to Ai
031ijk	Ai	Aj-Ak	Integer difference of (Aj) less (Ak) to Ai
†031i00	Ai	-1	Transmit -1 to Ai
031i0k	Ai	-Ak	Transmit the negative of (Ak) to Ai
031ij0	Ai	Aj-1	Integer difference of (Aj) less 1 to Ai
032ijk	Ai	Aj*Ak	Integer product of (Aj) and (Ak) to Ai
033i0x	Ai	CI	Channel number to Ai (j=0)
033ij0	Ai	CA,Aj	Address of channel (Aj) to Ai (j≠0; k=0)
033ij1	Ai	CE,Aj	Error flag of channel (Aj) to Ai (j≠0; k=1)
034ijk	Bjk,Ai	,A0	Read (Ai) words to B register jk from (A0)
†034ijk	Bjk,Ai	0,A0	Read (Ai) words to B register jk from (A0)

CRAY-1	CAL		DESCRIPTION
062ijk	Si	Sj+FSk	Floating sum of (Sj) and (Sk) to Si
†062i0k	Si	+FSk	Normalize (Sk) to Si
063ijk	Si	Sj-FSk	Floating difference of (Sj) and (Sk) to Si
†063i0k	Si	-FSk	Transmit normalized negative of (Sk) to Si
064ijk	Si	Sj*FSk	Floating product of (Sj) and (Sk) to Si
065ijk	Si	Sj*HSk	Half precision rounded floating product of (Sj) and (Sk) to Si
066ijk	Si	Sj*RSk	Full precision rounded floating product of (Sj) and (Sk) to Si
067ijk	Si	Sj*ISk	2 - Floating product of (Sj) and (Sk) to Si
070ijx	Si	/HSj	Floating reciprocal approximation of (Sj) to Si
071i0k	Si	Ak	Transmit (Ak) to Si with no sign extension
071i1k	Si	+Ak	Transmit (Ak) to Si with sign extension
071i2k	Si	+FAk	Transmit (Ak) to Si as unnormalized floating point number
071i3x	Si	0.6	Transmit constant 0.75*2**48 to Si
071i4x	Si	0.4	Transmit constant 0.5 to Si
071i5x	Si	1.	Transmit constant 1.0 to Si
071i6x	Si	2.	Transmit constant 2.0 to Si
071i7x	Si	4.	Transmit constant 4.0 to Si
072ixx	Si	RT	Transmit (RTC) to Si
073ixx	Si	VM	Transmit (VM) to Si
074ijk	Si	Tjk	Transmit (Tjk) to Si
075ijk	Tjk	Si	Transmit (Si) to Tjk
076ijk	Si	Vj,Ak	Transmit (Vj, element (Ak)) to Si
077ijk	Vi,Ak	Sj	Transmit (Sj) to Vi element (Ak)
†077i0k	Vi,Ak	0	Clear Vi element (Ak)
10hijkm	Ai	exp,Ah	Read from ((Ah) + exp) to Ai (A0=0)
†100ijkm	Ai	exp,0	Read from (exp) to Ai
†100ijkm	Ai	exp,	Read from (exp) to Ai
†10hi000	Ai	,Ah	Read from (Ah) to Ai
11hijkm	exp,Ah	Ai	Store (Ai) to (Ah) + exp (A0=0)
†110ijkm	exp,0	Ai	Store (Ai) to exp
†110ijkm	exp,	Ai	Store (Ai) to exp
†11hi000	,Ah	Ai	Store (Ai) to (Ah)
12hijkm	Si	exp,Ah	Read from ((Ah) + exp) to Si (A0=0)
†120ijkm	Si	exp,0	Read from exp to Si
†120ijkm	Si	exp,	Read from exp to Si
†12hi000	Si	,Ah	Read from (Ah) to Si
13hijkm	exp,Ah	Si	Store (Si) to (Ah) + exp (A0=0)
†130ijkm	exp,0	Si	Store (Si) to exp
†130ijkm	exp,	Si	Store (Si) to exp
†13hi000	,Ah	Si	Store (Si) to (Ah)
140ijk	Vi	Sj&Vk	Logical products of (Sj) and (Vk) to Vi
†140i00	Vi	0	Clear Vi
141ijk	Vi	Vj&Vk	Logical products of (Vj) and (Vk) to Vi
142ijk	Vi	Sj!Vk	Logical sums of (Sj) and (Vk) to Vi
†142i0k	Vi	Vk	Transmit (Vk) to Vi

†035ijk	0,A0	Bjk,Ai	Store (Ai) words at B register jk to (A0)
036ijk	Tjk,Ai	,A0	Read (Ai) words to T register jk from (A0)
†036ijk	Tjk,Ai	0,A0	Read (Ai) words to T register jk from (A0)
037ijk	,A0	Tjk,Ai	Store (Ai) words at T register jk to (A0)
†037ijk	0,A0	Tjk,Ai	Store (Ai) words at T register jk to (A0)
040ijkm	Si	exp	Transmit jkm to Si
041ijkm	Si	exp	Transmit exp = 1's complement of jkm to Si
042ijk	Si	<exp	Form 1's mask exp = 64-jk bits in Si from the right
	Si	#>exp	
†042i77	Si	1	Enter 1 into Si
†042i00	Si	-1	Enter -1 into Si
043ijk	Si	>exp	Form 1's mask exp = jk bits in Si from the left
	Si	#<exp	
†043i00	Si	0	Clear Si
044ijk	Si	Sj&Sk	Logical product of (Sj) and (Sk) to Si
†044ij0	Si	Sj&SB	Sign bit of (Sj) to Si
†044ij0	Si	SB&Sj	Sign bit of (Sj) to Si (j≠0)
†045ijk	Si	#Sk&Sj	Logical product of (Sj) and 1's complement of (Sk) to Si
†045ij0	Si	#SB&Sj	(Sj) with sign bit cleared to Si
046ijk	Si	Sj\Sk	Logical difference of (Sj) and (Sk) to Si
†046ij0	Si	Sj\SB	Toggle sign bit of Sj, then enter into Si
†046ij0	Si	SB\Sj	Toggle sign bit of Sj, then enter into Si (j≠0)
047ijk	Si	#Sj\Sk	Logical equivalence of (Sk) and (Sj) to Si
†047i0k	Si	#Sk	Transmit 1's complement of (Sk) to Si
†047ij0	Si	#Sj\SB	Logical equivalence of (Sj) and sign bit to Si
†047ij0	Si	#SB\Sj	Logical equivalence of (Sj) and sign bit to Si (j≠0)
†047i00	Si	#SB	Enter 1's complement of sign bit into Si
050ijk	Si	Sj!Si&Sk	Logical product of (Si) and (Sk) complement ORed with logical product of (Sj) and (Sk) to Si
†050ij0	Si	Sj!Si&SB	Scalar merge of (Si) and sign bit of (Sj) to Si
051ijk	Si	Sj!Sk	Logical sum of (Sj) and (Sk) to Si
†051i0k	Si	Sk	Transmit (Sk) to Si
†051ij0	Si	Sj!SB	Logical sum of (Sj) and sign bit to Si
†051ij0	Si	SB!Sj	Logical sum of (Sj) and sign bit to Si (j≠0)
†051i00	Si	SB	Enter sign bit into Si
052ijk	S0	Si<exp	Shift (Si) left exp = jk places to S0
053ijk	S0	Si>exp	Shift (Si) right exp = 64-jk places to S0
054ijk	Si	Si<exp	Shift (Si) left exp = jk places
055ijk	Si	Si>exp	Shift (Si) right exp = 64-jk places
056ijk	Si	Si,Sj<Ak	Shift (Si and Sj) left (Ak) places to Si
†056ij0	Si	Si,Sj<1	Shift (Si and Sj) left one place to Si
†056i0k	Si	Si<Ak	Shift (Si) left (Ak) places to Si
057ijk	Si	Sj,Si Ak	Shift (Sj and Si) right (Ak) places to Si
†057ij0	Si	Sj,Si>1	Shift (Sj and Si) right one place to Si
†057i0k	Si	Si>Ak	Shift (Si) right (Ak) places to Si
060ijk	Si	Sj+Sk	Integer sum of (Sj) and (Sk) to Si
061ijk	Si	Sj-Sk	Integer difference of (Sj) and (Sk) to Si
†061i0k	Si	-Sk	Transmit negative of (Sk) to Si

† Special CAL syntax format

144ijk	Vi	Sj\Vk	Logical differences of (Sj) and (Vk) to Vi
145ijk	Vi	Vj\Vk	Logical differences of (Vj) and (Vk) to Vi
146ijk	Vi	Sj!Vk&VM	Transmit (Sj) if VM bit = 1; (Vk) if VM bit = 0 to Vi
†146i0k	Vi	#VM&Vk	Vector merge of (Vk) and 0 to Vi
147ijk	Vi	Vj!Vk&VM	Transmit (Vj) if VM bit = 1; (Vk) if VM bit = 0 to Vi
150ijk	Vi	Vj<Ak	Shift (Vj) left (Ak) places to Vi
†150ij0	Vi	Vj<1	Shift (Vj) left one place to Vi
151ijk	Vi	Vj>Ak	Shift (Vj) right (Ak) places to Vi
†151ij0	Vi	Vj>1	Shift (Vj) right one place to Vi
152ijk	Vi	Vj,Vj<Ak	Double shift (Vj) left (Ak) places to Vi
†152ij0	Vi	Vj,Vj<1	Double shift (Vj) left one place to Vi
153ijk	Vi	Vj,Vj>Ak	Double shift (Vj) right (Ak) places to Vi
153ij0	Vi	Vj,Vj>1	Double shift (Vj) right one place to Vi
154ijk	Vi	Sj+Vk	Integer sums of (Sj) and (Vk) to Vi
155ijk	Vi	Vj+Vk	Integer sums of (Vj) and (Vk) to Vi
156ijk	Vi	Sj-Vk	Integer differences of (Sj) and (Vk) to Vi
†156i0k	Vi	-Vk	Transmit negative of (Vk) to Vi
157ijk	Vi	Vj-Vk	Integer differences of (Vj) and (Vk) to Vi
160ijk	Vi	Sj*FVk	Floating products of (Sj) and (Vk) to Vi
161ijk	Vi	Vj*FVk	Floating products of (Vj) and (Vk) to Vi
162ijk	Vi	Sj*HVk	Half precision rounded floating products of (Sj) and (Vk) to Vi
163ijk	Vi	Vj*HVk	Half precision rounded floating products of (Vj) and (Vk) to Vi
164ijk	Vi	Sj*RVk	Rounded floating products of (Sj) and (Vk) to Vi
165ijk	Vi	Vj*RVk	Rounded floating products of (Vj) and (Vk) to Vi
166ijk	Vi	Sj*IVk	2 - floating products of (Sj) and (Vk) to Vi
167ijk	Vi	Vj*IVk	2 - floating products of (Vj) and (Vk) to Vi
170ijk	Vi	Sj+FVk	Floating sums of (Sj) and (Vk) to Vi
†170i0k	Vi	+FVk	Normalize (Vk) to Vi
171ijk	Vi	Vj+FVk	Floating sums of (Vj) and (Vk) to Vi
172ijk	Vi	Sj-FVk	Floating differences of (Sj) and (Vk) to Vi
†172i0k	Vi	-FVk	Transmit normalized negatives of (Vk) to Vi
173ijk	Vi	Vj-FVk	Floating differences of (Vj) and (Vk) to Vi
174ijx	Vi	/HVj	Floating reciprocal approximations of (Vj) to Vi
175xj0	VM	Vj,Z	VM=1 where (Vj) = 0
175xj1	VM	Vj,N	VM=1 where (Vj) ≠ 0
175xj2	VM	Vj,P	VM=1 where (Vj) positive
175xj3	VM	Vj,M	VM=1 where (Vj) negative
176ixk	Vi	,A0,Ak	Read (VL) words to Vi from (A0) incremented by (Ak)
†176ix0	Vi	,A0,1	Read (VL) words to Vi from (A0) incremented by 1
177xjk	,A0,Ak	Vj	Store (VL) words from Vj to (A0) incremented by (Ak)
†177xj0	,A0,1	Vj	Store (VL) words from Vj to (A0) incremented by 1