2 CONVEX Architecture

2.1 System Architecture Overview

The CONVEX C-1 system is a high performance system based on a proprietary bus-oriented architecture. The central processing unit (CPU) portion of the system utilizes a CRAY-like architecture. This architecture employs vector accumulators and multiple arithmetic units functioning at 100 nanoseconds for 64-bit operands and 50 nanoseconds for 32-bit operands to achieve peak processing rates exceeding 60 million operations per second(MOPS).

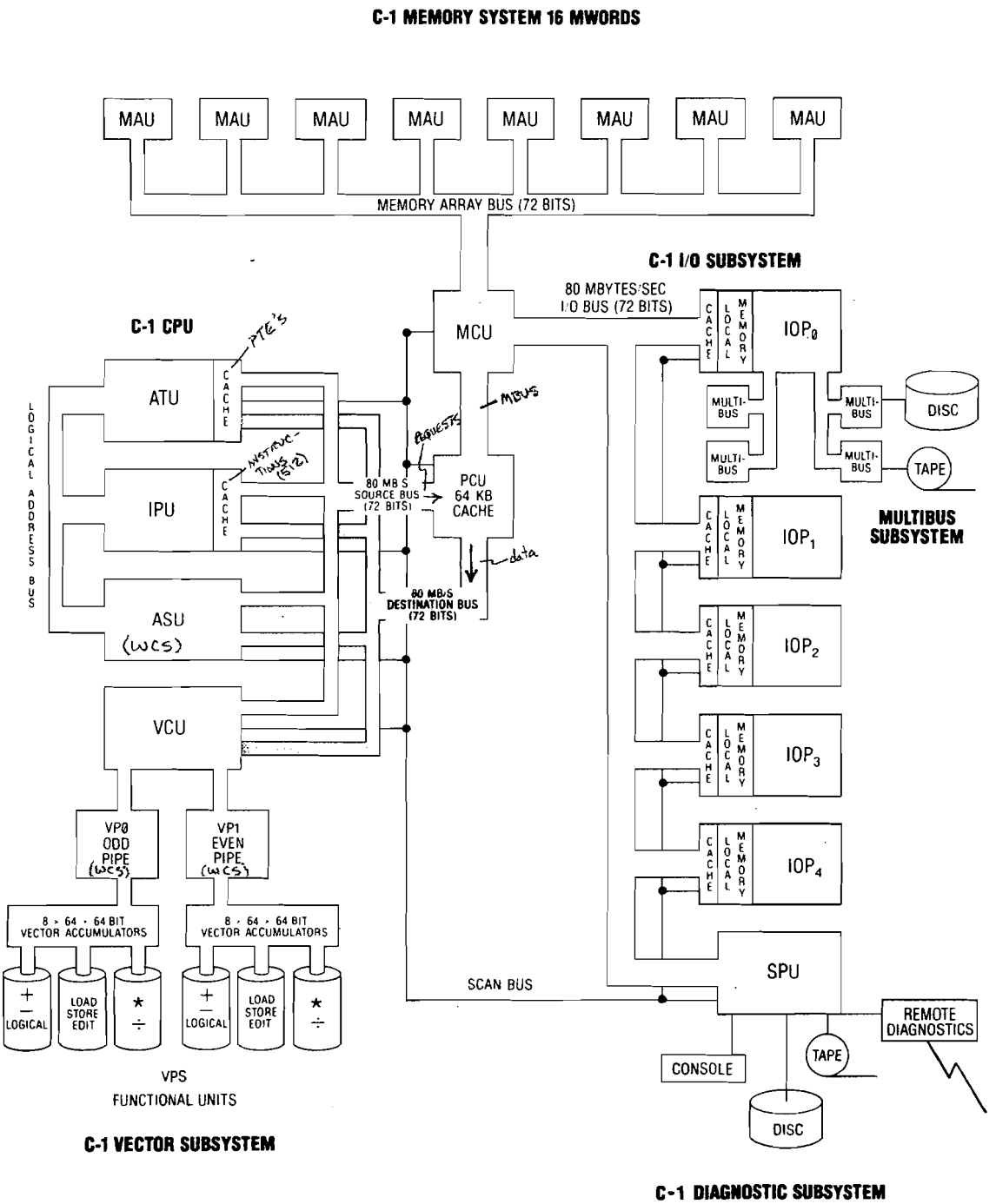
The C-1 central processor(sometimes referred to as Job Processor - JP) consists of multiple asynchronous units, operating concurrently in a pipelined fashion. These units are interconnected through multiple high-speed buses. The asynchronous nature of these units permits up to 8 operations to be executing at the same time. The inclusion of a highly intelligent I/O system permits the vast majority of I/O operations to be concurrently executed and controlled independent of the central processor. The off-loading of these functions makes the central processor much more productive.

The CONVEX C-1 is a tightly coupled asymmetric multiprocessor that comprises five functional subsystems:

1. Central Processing Unit Subsystem
2. Memory Subsystem
3. Input/Output Subsystem
4. Service Processor Subsystem
5. Electromechanical Subsystem

Refer to Figure 2-1. The service processor, input/output processor, and central processing unit subsystems make this a multiprocessor system. It is an asymmetric system because each processor performs a unique set of tasks. The system is tightly coupled because communication between the processors is through shared memory. UNIX is the main operating system. There are two smaller specialized operating systems called SPU UNIX and SPOKE that coexist with UNIX. Portions of UNIX are shared by all processors, and other specialized portions of UNIX are used only by specific processors.

Figure 2–1: CONVEX C-1 Functional Subsystems



2.2 Central Processing Unit Subsystem

The CPU subsystem performs the computational functions. These include arithmetic and logical operations on scalar and vector data.

The central processing unit (CPU) subsystem comprises five parts:

1. Physical Cache Unit (PCU)
2. Address Translation Unit (ATU)
3. Instruction Processor Unit (IPU)
4. Address and Scalar Unit (ASU)
5. Vector Processor System(VPS)

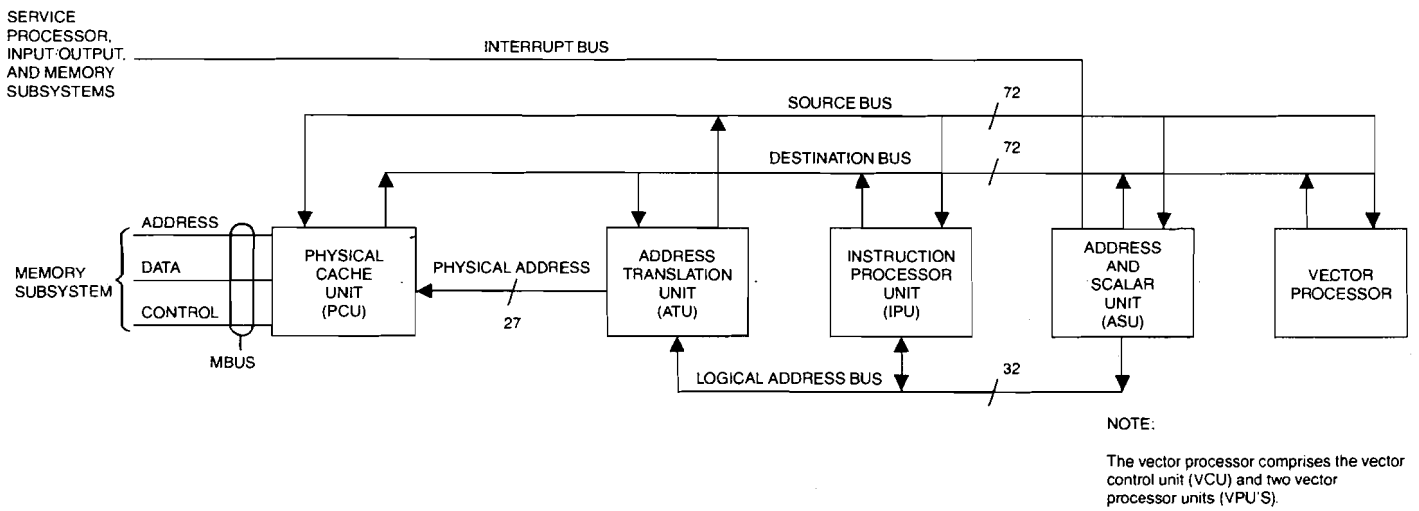
    a. Vector Control Unit (VCU)
    b. Vector Processor Unit (VPU) (two identical boards)

Refer to Figure 2-2. The source bus carries operands from the ATU to the PCU, IPU, ASU, VCU, and VPU's. The destination bus carries operands from the PCU, IPU, ASU, VCU, and VPU's to the ATU. Logical addresses move between the ATU, IPU, and ASU over the logical address (LA) bus. Physical addresses are sent from the ATU to the PCU through the back plane and sent from the PCU to the Memory Control Unit(MCU) through the MBUS in connectors on the foreplane.

2.2.1 Physical Cache Unit

The PCU interfaces the CPU to the MCU. The MCU deals with memory only in terms of 64-bit longwords, longword aligned. The CPU sees memory in terms of two independent 32-bit data paths, byte aligned. That is, the CPU may load or store two 32-bit words on every clock cycle, with both words coming from unrelated memory addresses, aligned on arbitrary byte boundaries. The PCU control logic maps each such request into the most efficient pattern of longword accesses and cycles the MCU to achieve them.

Figure 2–2: Central Processing Unit Subsystem



2.2.1.1 CPU/Memory Interface

The CPU/memory interface provides a two-level pipeline and a seven-level pipeline. Requests for data that are in the p-cache use the two-level pipeline. This is necessary because typical PCU operations require two clock cycles. The request is decoded and the p-cache is accessed in the first clock cycle, and data transfer occurs in the second clock cycle. The two-level pipeline permits the decoding of one request and the data transfer of the preceding request to be performed concurrently. Requests for data that are not in the p-cache use the seven-level pipeline. The seven-level pipeline can accept a request for data from main memory on every clock cycle.

The PCU receives store data and parity from the source bus, and it drives data and parity from the p-cache, main memory, or both, onto the destination bus.

The MCU and PCU are connected by the MBUS. Data and parity are bidirectional signals in the MBUS. The MBUS also contains address and control signals.

2.2.1.2 P-cache

The PCU also manages a pipeline of addresses and data that enable longword memory access to begin on every clock cycle. This depends on the addresses being aligned such that full interleave is possible. Because this is not always the case, and to provide lower latency to scalar operations, the PCU also contains the P-cache, a 64 Kbyte cache memory accessed with physical addresses. This cache provides dual 32-bit data ports, allowing two arbitrary cache locations to be read or written on each clock cycle. A write-back algorithm is used, so that data written to the cache is not written into main memory until the block is overwritten by a subsequent access or until an I/O operation attempts to access the old data in main memory.

2.2.1.3 P-cache Bypass

There are three cases encountered when loading or storing vectors from memory. The vectors can be contiguous in memory, can have constant stride, or can involve random memory accesses (gather/scatter). According to a study done at one of the national laboratories involved in scientific processing, typical programs contained code using contiguous elements 78% of the time, constant stride 20%, and random access 2%. Therefore a technique to enhance performance for the contiguous case would be very beneficial. This technique is the cache bypass.

The p-cache bypass prevents instruction fetches and sequential vectors from overwriting operands in the p-cache. The address translation unit (ATU) can designate an operand request as bypass mode. In this case, the operand will be transferred to or from main memory unless main memory has not been updated by the p-cache. In this case, the operand is transferred to or from the p-cache.

Physical memory accessing is fully pipelined. For contiguous memory locations the memory interleaving is efficiently utilized, thus the total time for loading or storing contiguous vectors via cache bypass is practically the same as the time for loading or storing using the p-cache. The benefit of this approach is that data already residing in the p-cache does not need to be flushed. Thus the purpose of the p-cache in storing multiple non-contiguous quantities is not compromised by the loading/storing of long contiguous vectors.

Those operations involving constant stride and random access often cannot take full advantage of memory interleaving and thus would not benefit from cache bypass. For the first constant stride or random access of particular operands, a cache load is required, but subsequent accesses to those operands are to the p-cache which operates at the maximum bandwidth of the memory system.

The result of this design is that the hardware automatically chooses the most efficient method for loading or storing vectors, totally transparent to the user.

2.2.1.4 Referenced Bits and Modified Bits

The referenced and modified bits indicate activity for each memory page that can be physically implemented in the CONVEX C-1. A table of referenced bits indicates which physical pages of memory have been accessed by the CPU subsystem. A referenced bit is set when the CPU subsystem accesses (reads or writes) the corresponding physical page of memory. A table of modified bits indicates which physical pages of memory have been written into by the CPU subsystem. A modified bit is set when the CPU subsystem writes into the corresponding physical page of memory. These tables are used by the operating

system to keep track of the least recently used pages of physical memory. Both tables must be initialized (set to all zeros) by the CPU subsystem when power is applied to the system.

2.2.2 Address Translation Unit

The Address Translation Unit(ATU) contains the circuitry required to translate logical addresses generated by the Address & Scalar Unit (ASU) or the Instruction Preprocessing Unit (IPU) into the physical addresses required by the Physical Cache Unit (PCU), thus the name Address Translation Unit. However, the address translation function is only one of many functions performed by the ATU. The primary ATU functions:

- Contain and control the primary data path switch for the CONVEX system.
- Provide arbitration and source/destination selection for all system busses.
- Align the data being transferred as required by the destination.
- Stages data as required to deliver complete operands to the destination.
- Maintain a logical cache of aligned data to accelerate memory accesses.
- Perform the majority of the memory address computations for the CONVEX system.
- Provide arbitration and control for the logical address bus.
- Perform indirect addressing for data accesses.
- Generate secondary logical addresses for non-aligned scalar accesses.
- Generate secondary logical addresses for all vector accesses.
- Perform logical to physical address translation for all memory accesses.
- Maintain a cache of page table entries to accelerate address translation.
- Perform all protection checks for data accesses to generate system faults.
- Contain control and queuing for up to nine concurrent memory accesses.

2.2.2.1 Data alignment

Operands in registers are aligned to the register. Operands in memory may begin on any byte boundary. The ATU provides the means to go from one alignment to the other during the data transfer to or from memory. Alignment is performed as data is moved from the destination bus to the source bus, and the ATU provides arbitration among the users of these busses.

2.2.3 Instruction Processor Unit

The Instruction Preprocessing Unit (IPU) is a major speed factor in the CONVEX C-1. All instructions are preprocessed and pipelined by the IPU in a manner such that the Address and Scalar Unit or the Vector Units received the decoded instructions and displacement fields each clock cycle.

The Instruction Preprocessing Unit (IPU) performs the following major functions:

- Dispatches instructions to the VCU and ASU with the proper fields and an entry point address for the ASU or VCU microcontroller.
- Cracks(decodes) instruction opcodes.

- Maintains the system Program Counter.
- Maintains an instruction cache.

The IPU saves machine time by executing some instructions such as branch or jump instead of dispatching them to the ASU or VCU.

2.2.3.1 Instruction Decode and Dispatch

The IPU is started by execution of a branch by the ASU. Subsequently, it fetches, decodes, and presents each instruction for dispatch. It always attempts to have ready the next instruction to be executed by either the ASU or VCU, or both. It will execute unconditional branches without intervention of the ASU. When a conditional branch is decoded, it will wait until the ASU signals whether to branch or not. Owing to a unique pre-decoding scheme, conditional branches are usually executed in a single clock cycle.

2.2.3.2 Instruction Pre-fetching and Cacheing

In order to minimize the amount of time the processor is waiting on instruction fetches, a cache is provided which contains up to 512 instructions. The IPU attempts to fetch instructions from this cache, which may be accessed without interfering with memory data operations. In addition, a pre-fetcher looks ahead of the current point of execution and pre-loads the cache if it is not already loaded.

2.2.4 Address and Scalar Unit

The Address and Scalar Unit (ASU) is a 64-bit microprogrammed processor within the CONVEX C-1 Computer. The ASU performs the following functions within the C-1 computer.

- Executes all address register manipulation instructions. The Operands for multiplication and division are sent to the VPU, which does the arithmetic operation.
- Executes all scalar register manipulation instructions. The Operands for multiplication, division, population counts, floating point adds and subtracts, and portions of float to fix and fix to float conversions are sent to the VPU for arithmetic operations.
- Generates operand address for address, scalar, and vector loads and stores.
- Executes all program control and priviledged instructions.
- Generates addresses and control the JP during context load and stores.
- Maintains the interval timer and elapsed time counter.
- Maintains the Program Status Word (PSW).

2.2.5 Vector Processor

The vector processor consists of one vector control unit (VCU) and two vector processor units (VPU's). The VCU provides all timing and control. The VPU's provide the vector accumulators and data paths, but are totally under control of the VCU. Refer to Figure 2-3.

2.2.5.1 Vector Control Unit

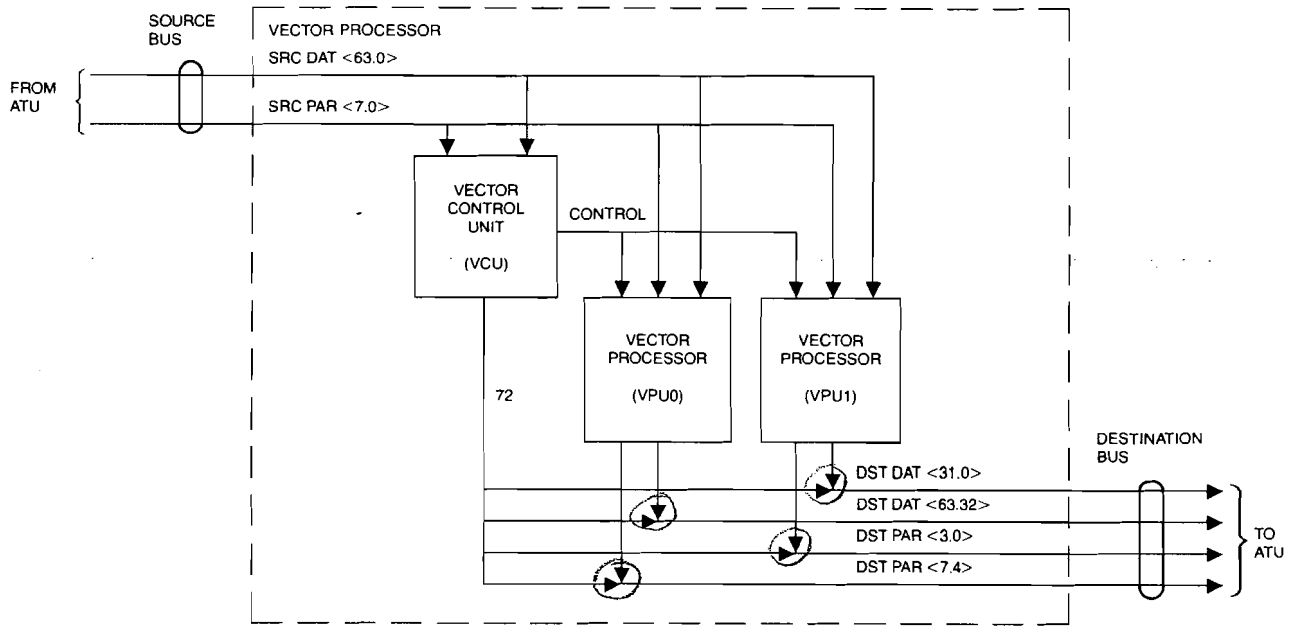
The Vector Control Unit is the master of the vector processor. It directs data to and from the Vector Processors. All vector processing and some scalar processing, ie, floating point operations, multiplication and division, are handled by the Vector Control Unit.

There are three independent microcoded controllers on the VCU:

- Load and Store Controller
- Multiplier and Divider Controller
- Adder and Logical Controller

All of these controllers are idle until started by the instruction dispatch control. The instruction dispatch receives vector register specification fields, an operand size code, a PSW hazard bit, and an entry address from the Instruction Processor Unit (IPU). Hazard conditions with the utilization of registers or the controllers are checked prior to issuing an instruction to one of the three microcoded controllers. The IPU is signaled with a ready signal when the registers and function controller specified by the opcode are not busy and the opcode is accepted. The instruction dispatch sends a microcode entry address, function unit opcode, and address counter claims to the appropriate controllers as the instruction is issued within the VCU.

Figure 2–3: Vector Processor



2.2.5.2 Vector Processor Unit

There are two identical VPU's in the CONVEX C-1. VPU0 contains the even data elements and VPU1 the odd elements of the eight vector accumulators. During 32-bit vector loads and stores, an even and an odd data element are moved on each 100 nanosecond clock cycle. In addition, both VPU's execute concurrently. This effectively doubles the data element processing rate for 32-bit operands relative to 64-bit.

The source bus provides the input data to the VPUs for vector load data, scalar floating point arithmetic operands, and VCU to VPU or VPU to VPU transfers. The output bus of VPU0 is wired to the upper 36 bits of the Destination Bus and VPU1 to the lower 36 bits of the bus. The ATU board comprehends this odd-even connection of VPU0 and VPU1 when vectors are transferred out of the vector registers. Long word (64 bit) vectors are transferred to and from the VPUs upper word first followed by the lower word. Longword vector stores proceed with the same order of vector elements being sourced by the VPUs.

2.2.5.3 Floating Point Operations

The ASU executes most scalar operations in a single 100 nanosecond clock cycle, however, it has no floating point capabilities. Hence, it sends scalar floating point operands to VPU0 for execution. For this reason, the VPU's receive the entire 64-bit source bus, since it allows both operands to be transferred at once during 32-bit operations.

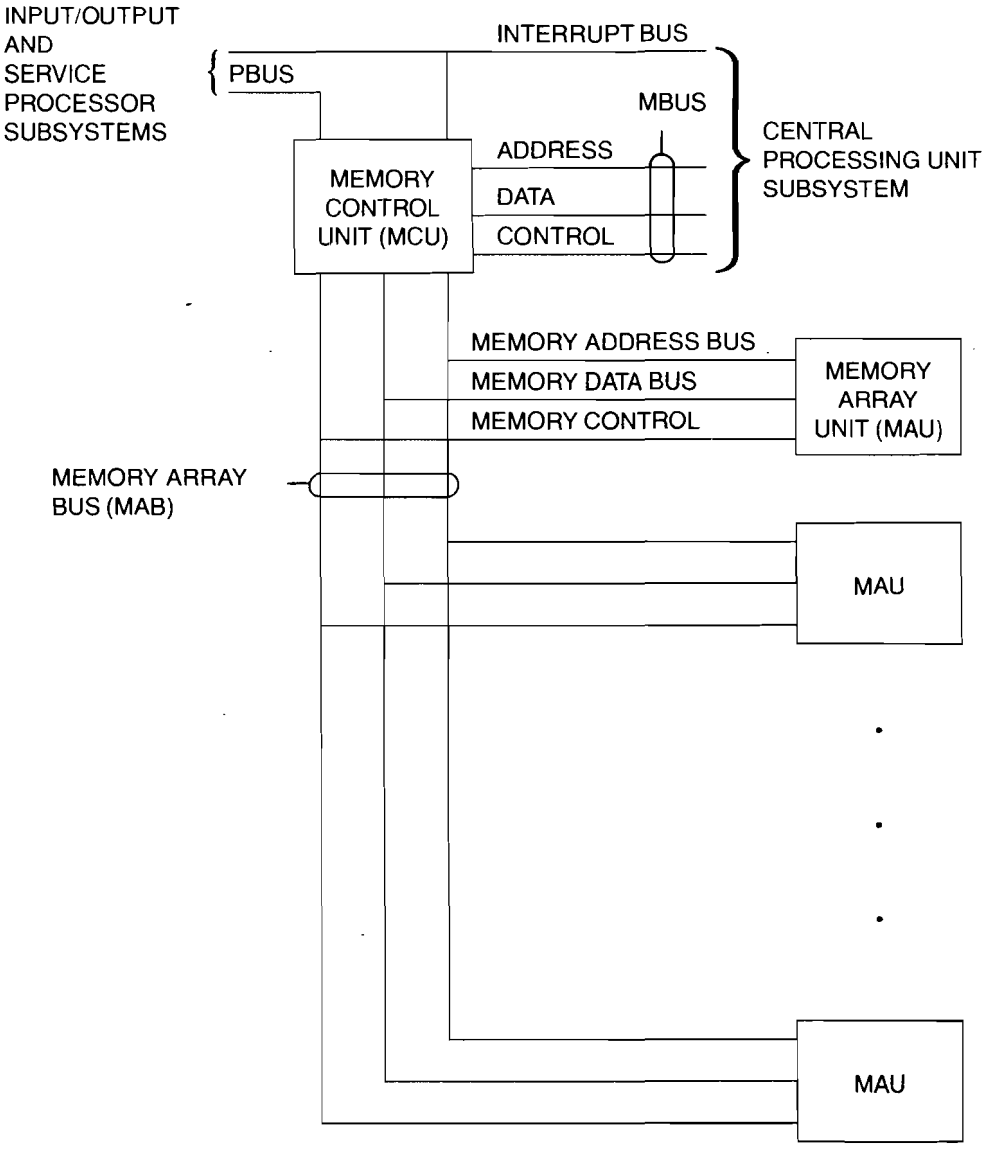
2.2.5.4 Chaining

The vector processor also provides for a technique whereby the output of one set of vector operations can be directed to the input for a different set of vector operations. This technique is called chaining and allows multiple operations to occur simultaneously. For example, one sequence of operations might call for the multiplication of two vectors of numbers, the addition of a third vector and the store of the result. The multiplication of the two vectors would be started; after the start-up delay of the multiplication unit the result from the first two operands would be available and would then be routed to the vector addition unit to be added to the third vector. The sum of these two operands would be given to the store functional unit for storage in the memory system. The resultant concurrency is made possible by the chaining of the three distinct operations.

2.3 Memory Subsystem

The memory subsystem provides the main memory for the CONVEX C-1. It includes one memory control unit (MCU) and from one to eight memory array units (MAU's) and connects to the CPU and I/O subsystems through the MBUS and PBUS paths respectively. Refer to Figure 2-4.

Figure 2–4: Memory Subsystem



NOTE:

From one to eight MAU's can be installed.

2.3.1 Memory Control Unit

The MCU provides all control functions and interfaces the two memory ports, the P-Bus and M-Bus, to main memory. It sequences the MAU's, both for data transfers and refresh operations, and maintains an error log for memory operations. It contains a configuration map which monitors memory accesses and traps accesses to non-existent or deconfigured memory. It also contains the arbiter for the system interrupt bus.

The MCU also contains a tag store which keeps a record of all data blocks which have been accelerated into the P-cache (PCU). If an I/O access is attempted to such a block, the PCU will ensure that only the most recent copy of the block is used for the I/O access.

2.3.2 P-Bus

The P-Bus provides a high-bandwidth, block oriented, path to main memory for the IOP's and SPU. Arbitration of the P-bus is performed by the MCU. A ring-grant algorithm which sequentially allocates the bus to requesters prevents any user from being starved. A set of latency timers assures that the bus is reallocated frequently during high usage.The P-Bus is optimized for burst transfers. Each transfer begins with the requester passing a control and address word, followed by continuous data transfer.

2.3.3 M-Bus

The M-Bus provides both high bandwidth and low-latency, but has a single user which eliminates the need to arbitrate usage or to propagate data on a multi-drop bus. It has parallel paths for address, control and data. It supports block transfers to and from the P-cache, but otherwise requires the PCU to supply one address for each data cycle.

2.3.4 Memory Array Unit

64-MB w/ 1-Mbit RAMS

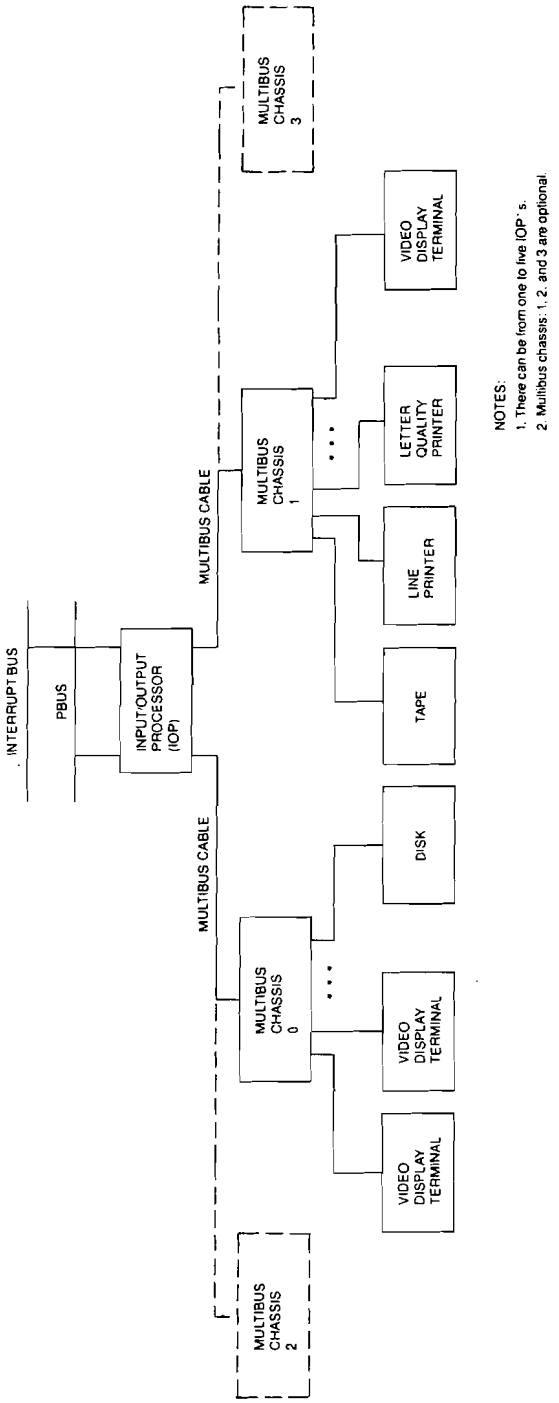
The memory array unit (MAU) is populated with 256K DRAM devices, for a board capacity of 16 Mbytes. The 16 Mbyte board is organized as 2 Mwords by 8 bytes. Cycle time of the array is 400 nanoseconds, but four way interleaving based on the least significant two word address bits allows sequential accesses to be started every 100 ns.

128-MB max w/16-MB boards
512-MB max w/64-MB boards

2.4 Input/Output Subsystem

The C-1 system performs I/O via distributed intelligent I/O processors, which transfer data to and from system memory over the P-Bus, a CONVEX-proprietary 80 Mbyte/sec bus. One to five IOP's may be configured on a C-1. Refer to Figure 2-5.

Figure 2–5: Input/Output Subsystem



2.4.1 Input/Output Processor Unit

The C-1 I/O Processor (IOP) interfaces Multibus controllers to the P-Bus. It incorporates a 68000 microprocessor with 512 Kbytes of local memory and a high speed 32Kbyte cache. I/O device drivers execute on the 68000, under a resident executive called SPOKE. Data transfers may, under device driver control, be either to/from local IOP memory, or to/from main memory, buffered through the IOP's cache.

The IOP provides two multibus ports, each of which may support up to two multibus cages, although highest performance is available when only one cage per port is used. Each port is capable of a 4 Mbyte/sec transfer rate.

2.4.2 Multibus Chassis

Each chassis contains one Multibus Control Unit (MBCU), and from one to eight Multibus device controllers. The MBCU buffers the cable interface and provides a multibus compatible protocol within the card cage. This allows industry standard multibus controllers to be used to interface I/O devices.

A second version of the Multibus Chassis is available that enables two MBCU controllers. This effectively provides two logical expansion chassis, for up to three device controllers each, in a single physical unit.

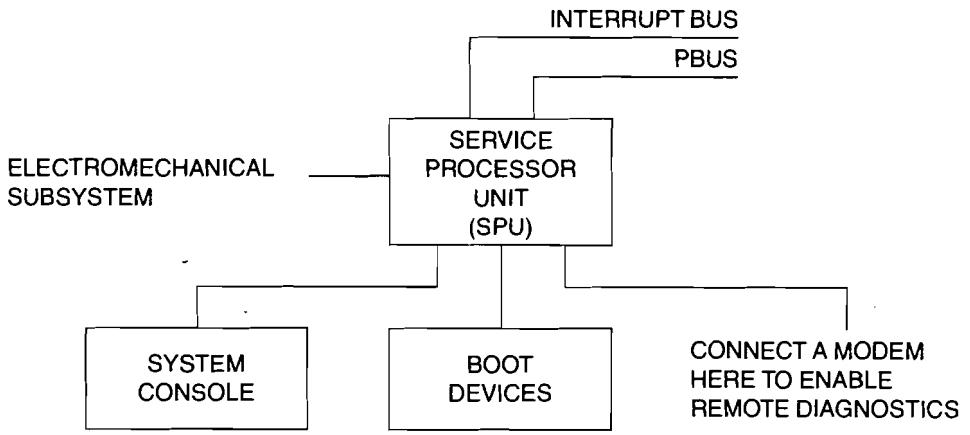
2.5 Service Processor Subsystem

Integral to the CONVEX C-1 is a Service Processor Unit (SPU). The is a microprocessor-based unit that controls the operation of diagnostic programs, as well as maintaining a log of detected and corrected errors. The SPU runs under its own diagnostic UNIX operating system, and contains a 20 MByte Winchester disk and cartridge tape peripheral devices as well as communications ports for the operator's console and for remote diagnostics. The disk maintains an on-line log of errors as well as code for the diagnostics. The cartridge tape, with a capacity of 20 MBytes, is used for software distribution for both system and diagnostic software and may be used as a distribution medium for user software. Refer to Figure 2-6.

2.5.1 Service Processor Unit

The Service Processor Unit (SPU) is a microcomputer that is connected to the PBUS and interrupt bus. The system console is connected to the SPU through one RS232 port. Another RS232 port is provided as a remote diagnostics port. Two boot devices -- the Winchester disk and cartridge tape drive -- are connected to the SPU. The Winchester disk is connected to the SPU through a Small Computer System Interface (SCSI). The cartridge tape drive is connected to a controller on the SPU.

Figure 2–6: Service Processor Subsystem



NOTE:

Boot devices include a Winchester disk and a cartridge tape drive.

2.5.1.1 SPU Functions

During normal operation, the SPU:

1. Provides an interface to the system console
2. Provides an interface to the boot devices
3. Provides all of the system clocks
4. Provides a battery backed-up real-time clock/calendar
5. Executes UNIX

During diagnostics, the SPU:

1. Provides a remote diagnostics port
2. Executes UNIX
3. Controls the system with:

    a. Scan rings(diagnostic bus)
    b. The PBUS

2.5.1.2 System Initialization

The SPU is the only system element which comes up running from a system reset. A system reset is generated automatically when the system is powered up, or may be generated manually by the reset switch when the keylock switch is not in the "secure" position. Following the reset, the SPU will begin execution from on-board EPROM. Depending on the switch settings, the SPU's next actions will vary from waiting for further input from the system console to commencing a boot of the full system.

Assuming the latter, the will load UNIX into SPU memory and begin execution of the CPU boot procedure, which runs under UNIX. This boot procedure will determine the size and configuration of main memory, and load this information into a protection map on the MCU. It will also generate a logical to physical page translation table in main memory, which allows the CPU to begin execution using logical addresses.

With memory initialized, the SPU will load CONVEX UNIX boot image from its hard disk into main memory, and then initialize the CPU scan rings, registers, and writable control stores. The IOP's must also be loaded from the hard disk with a SPOKE I/O executive image. Finally, the SPU turns the system clocks on, and the CPU and IOP's then proceed to bring up UNIX.

2.5.1.3 Interface to Electromechanical Subsystem

The SPU senses the environmental monitors and front panel switches via a cable connection to the System Monitor Board (SMB). These environmental monitors allow it to sense air temperature, airflow, and ac and dc voltages. In addition to being able to read the state of these monitors under program control, the SMB will interrupt the if it detects an out-of-range value. Certain conditions, in fact, will cause the system power to be shut down. The also controls the front panel lights via its connection to the SMB.

2.5.1.4 Battery Backed-up Clock

The battery backed-up clock on the provides the year, month, date, day, and time when the system is initialized. The battery backed-up clock is a clock/calendar hybrid module that contains a CMOS integrated circuit, a crystal, and a lithium battery. When the system is running, time is derived from the 60 Hz ac power line. When ac power is not applied to the system, the lithium battery powers the clock/calendar module.

2.5.2 System Console

The system console is physically connected to the SPU, and serves as the console during system boot or diagnostic execution. In addition, processes on the SPU and CPU logically connect it to the CPU, and during normal UNIX execution, it serves also as the UNIX system console.

2.5.3 Boot Devices

Two magnetic storage devices are attached to the SPU, a 20 Mbyte Winchester disk, and a 1/4 inch cartridge tape drive. The disk is required by UNIX, and additionally contains the load images used to boot the CPU and IOP's. It also contains diagnostic programs used to maintain the system.

The cartridge tape transports data to and from the SPU. If the disk is empty, as on a new system, or has been corrupted, a special program can format and load the disk from the cartridge tape. Additionally, the cartridge tape provides a low-cost mechanism to distribute software, both for the SPU and CPU.

2.5.4 Remote Console Connection

The remote console connection is nearly identical to the system console connection, except that it is hardware disabled when the keylock switch is not in the "remote" position on the front panel. It is intended to be connected through a modem to a dial-up phone line. This allows a remote service center to have the same access to the system as a local field engineer, when enabled by the customer.

2.6 Reliability/Availability/Serviceability

The SPU subsystem controls one of the most sophisticated diagnostic systems available. Each subsystem is interfaced to the SPU via a proprietary diagnostic scan bus. Once initiated, the diagnostic software scans in test patterns and scan out results without using any of the buses used during normal computations. This approach results in high resolution diagnostics thereby significantly reducing the mean time to repair. In addition, the provision of a remote diagnostics capability improves the efficiency of maintenance personnel, resulting in minimum downtime.

Additional advanced features include: extensive parity checking throughout the entire design, single-bit error correction and double-bit error detection for main memory accesses, alterable control store within the ASU and VPS, as well as diagnostic-readable serial numbers contained on each logic board. This last feature permits the diagnostic system to ensure that boards are up to the latest revision level and in the proper slots.

2.7 Compact, Rugged Packaging

The CONVEX C-1 system is housed in a standard 19 inch RETMA rack, 5 feet high with a 'foot print' of less than 7 square feet. Contained within this rack are: the CONVEX C-1 central processing unit, up to 128 Mbytes of main memory, up to 5 IOP's, the SPU with its cartridge tape and winchester disk, and one MULTIBUS card cage. A typical system configuration includes a second, adjacent rack, which could contain a tape drive and two

414MB (formatted) winchester disk drives, resulting in an extremely powerful system in a very compact package. The CONVEX C-1 is air cooled, with cooling provided by two fans drawing air through the front of the rack and exhausting through the rear.

All CONVEX processor boards are vertically mounted and attached to the backpanel via rugged 96-pin DIN connectors. To ensure rigidity, these boards are fastened to the card cage with a screw-type ejector and injector mechanism.

The components packaged in the processor cabinet consume approximately 3200 watts of input power in the standard configuration, and less than 4500 watts when fully configured with 128 MB of main memory. The system operates in a temperature range between +15 and +32 degrees centigrade.

3 Peripherals

The CONVEX C-1 offers a wide range of peripherals for input/output operations.

3.1 DKD-101 Disc Drive

The DKD-101 Disk Drive is a compact state-of-the-art moving head disk drive with a storage capacity of 474 MBytes (unformatted) in a very compact package. The DKD-101 disk drive is a 10.5-inch SMD Winchester, non-removable, sealed disk system offering high recording density, data transfer rate, superlative performance and high reliability. The DKD-101 is appropriate for large capacity, high speed, data storage for both on-line and large-scale data base applications.

The DKD-101 consists of a disk enclosure, four printed circuit boards, and a DC power supply unit. The disk enclosure is completely sealed and integrates six disks, Winchester-type contact start/stop heads, a rotary actuator, DC spindle motor, and IC read pre-amplifiers. The DKD-101 is rack-mountable in the standard CONVEX C-1 cabinet and interfaces to the CONVEX Disk Controller.

The DKD-101 includes the SMD Disk Controller. The controller interfaces to the CONVEX MULTIBUS as generated by the CONVEX C-1 IOP(Input Output Processor). It is a high performance DMA channel interface for maximum system throughput.

3.1.1 Storage Capacity

The DKD-101 has 842 cylinders with a 28,160 byte-unformatted track capacity, and their servo-controlled track-following system assures accurate head positioning on extremely high-density tracks of 880 tracks per inch. The DKD-001 offers the user a storage capacity of 474 megabytes on six surfaces.

3.1.2 Performance

The adoption of an advanced rotary actuator and a direct-drive DC spindle motor of 3,961 rotations per minute, as well as two heads per surface, allows for exceedingly high performance: 18 millisecond average positioning time (5 milliseconds for track-to-track, 35 milliseconds maximum), 7.5 millisecond average latency time, and 1.859 megabyte-per-second data transfer rate.

3.2 Magnetic Tape Drives

Today's ever increasing disk drive capacities in dramatically reduced packaging sizes are dictating the need for economic, flexible, and performance-oriented tape subsystems. The CONVEX MTD-001 and MTD-002 tape systems utilize state-of-the-art technology to meet the increased demands for full performance, general purpose tape subsystems. These systems also provide exceptional serviceability and maintainability through extensive resident micro-diagnostics.

3.2.1 MTD-001 Description

The MTD-001 tape subsystem is a dual-density (1600/6250 bpi), 50 ips, start/stop and GCR subsystem, complete with tape transport, formatter/controller, power supply, and resident microdiagnostics, all contained in a single self-contained package. The 24.5-inch high MTD-101 is mounted vertically in a standard 19-inch retma rack. Also included is a reliable mechanical tape buffering system that is capable of handling both high density Group Coded Recording (GCR) and convenient auto-threading.

3.2.2 MTD-001 Performance

The tape subsystem provides reliable start-stop performance. Disk/dump restore, interchange, archiving, batch processing, and journaling are important capabilities accomplished without special software modifications or repositioning. The MTD-001's average access time is only 5.6 milliseconds.

3.2.3 MTD-001 GCR Recording

GCR recording technology is an added plus for high capacity disk dump/restore applications. In 8K blocks, a 10.5-inch reel of magnetic tape will hold up to 134 megabytes of data. GCR also offers higher throughput. An average transfer rate of 233 kilobytes per second is typical using 8K data blocks. The instantaneous transfer rate is 312.5 kilobytes per second.

GCR also offers extensive read/write reliability with 1- and 2-track on-the-fly error correction; it will also indicate multi-track errors. The system verifies CRC, Aux CRC, and ECC characters during all GCR read and write operations.

3.2.4 MTD-002 High Performance Tape Subsystem

The MTD-002 is a high-performance, precision-vacuum, column-tape subsystem that meets all of the traditional high-performance start/stop digital tape requirements. These tape units operate at tape transport speeds of 125 inches per second. A choice of software or operator selectable recording densities is available featuring 800 bpi None-Return-Zero-Inverted (NRZI), 1600 bpi Phase Enabled (PE), and 6250 bpi Group Coded Recording (GCR) recording formats. The MTD-002 provides gentle but precise tape handling and reliable tape threading.

3.2.5 MTC-001 GCR Tape Controller

The CONVEX MTC-001 GCR Tape Controller is a MULTIBUS board which will interface either the MTD-001 or the MTD-002 tape subsystems to the CONVEX MULTIBUS as generated by the IOP. It is a high performance DMA channel with minimal intelligence, and depends on the IOP to perform I/O control block interpretation, error recovery and retry operations, and so forth.

It supports standard blocked tape formats, and also contains features to allow the tape subsystem to read and write gapless format tapes. It contains numerous diagnostic features to facilitate test and maintenance operations. The MTC-001 generates a full 24 bit DMA address to allow transfers to and from the IOP via the CONVEX MULTIBUS.

3.2.6 MTC-001 Controller Operation

MTC-001 has two levels of command buffers: the "pending" level and the "execution" level; status bits describe the state of both levels. The Input/Output Processor (IOP) writes commands to the pending level of the MTC-001 , one byte at a time. When the IOP loads a complete tape command into the pending level, a write to the start flag notifies the controller. This action also sets the command pending flag. As soon as the execution level is available, the controller copies the contents of the pending buffer to the execution level, clears the command pending flag, and sets the command executing flag. The IOP will then load the pending level with the next command.

This double buffering of tape commands allows the device driver to keep the tape as busy as possible. Rather than starting and stopping in the inter-block gaps, the tape will move through them at maximum velocity. However, if an unrecoverable error occurs during execution of one command, and another command is enqueued behind it, the controller will abort the second command, so that recovery procedures can be invoked.

The MTC-001 offers two modes of data transfer, and uses a bit in the MTC-001 control register to select the current mode. In normal blocked mode, inter-block gaps are expected during reads and generated during writes at the conclusion of each transfer. Blocked mode transfers will occur whenever the controller resets the chain mode bit.

In gapless mode, the controller does not expect nor generate inter-block gaps. Hardware support for double buffering allows the controller to transfer data continuously to or from the tape, generating interrupts to the IOP whenever data buffers are switched. Gapless transfers will occur when the controller sets the chain mode bit.

3.3 PRT-001 High Speed Printer

CONVEX Computer Corporation offers the PRT-101--a 600 lines per minute printer/plotter. In addition, the controller PRC-001 is also available as a separate product. The features of both products are detailed below.

3.3.1 Features

- 600 lines per minute dot matrix printer
- 96 character ASCII set of upper and lower case characters
- Full plotter capability
- Self-test feature with built-in diagnostics
- The PRC-001 provides dual printer support with a single controller
- Proven reliability and modular design
- Fully supported by CONVEX UNIX operating system and graphics software

3.3.2 Description

The PRT-101 dot matrix printer can overlap dots both horizontally and vertically. Characters are formed by electronic random access to PROM character sets, which place dots at appropriate positions on each horizontal line. As the paper advances, each successive line of appropriately placed dots overlaps the previous line to form characters which appear solid.

The PRT-101's hammer energy is optimized to print dots only, thereby maintaining uniform character density and quality, regardless of character size. This optimization allows for enhanced print quality, particularly noticeable on multi-part forms, and prevents character shadowing or ghosting from occurring.

The PRC-001 Dual Line Printer Controller will support two PRT-001 printers from one MULTIBUS slot. The controller accesses both printers over two independent high-speed Direct Memory Access (DMA) channels. Among the unique features are automatic printer selection and an on-board self test capability.

3.3.3 Plotting Capability

Full plotting capability stems from the inherent accuracy and ability of the PRT-101 line printer to place a single dot anywhere on the paper or print a solid sheet of overlapping dots. A single computer command puts the printer into the plot mode, enabling it to plot drawings, graphs, charts, and characters of any size or shape.

4 Software

4.1 CONVEX UNIX

The CONVEX UNIX operating system provides the highest level of programmer productivity and interactivity to the supercomputer user. Designed, optimized and enhanced for the C-1 supercomputer, CONVEX UNIX creates the ideal environment for the scientific user.

The C-1 architecture has many extensions specifically designed to support the UNIX environment of high interactivity among multiple users as well as computationally-intensive tasks. These extensions include hardware virtual memory support, memory protection (based on a ring structure), and full support for the multiple intelligent input/output subsystems of the C-1. Derived from 4.2 bsd, the CONVEX UNIX operating system provides the ideal environment for users of the CONVEX C-1 computer system. CONVEX UNIX is a virtual memory operating system that provides extensive functionality to span the range of applications in the scientific computing environment. By combining the excellent timesharing features that the UNIX operating system has with sophisticated batch processing capabilities, the CONVEX C-1 computer system is an excellent environment for interactive users as well as large production-oriented computing applications.

In addition to the facilities provided in CONVEX UNIX, the operating system also serves as the foundation for a variety of additional software for the management and sharing of information and additional program development tools.

4.1.1 Features

- Sophisticated virtual memory management enables programmers to concentrate on the application rather than on physical memory limitations by providing up to 4 Gigabytes of virtual memory.
- Full software support of the hierarchical hardware protection system, based on a ring structure, provides total program integrity.
- The CONVEX UNIX input/output is partitioned between the CPU and intelligent IOP subsystems, ensuring maximum throughput.
- User-level preemptive priority based task scheduling allows both interactive and batch applications to run efficiently at the same time.
- "Pipes" and filters are a powerful feature to redirect input and output from one process/command to another.
- Extensive set of cooperative software tools greatly improves the productivity of both programmers and users.

- The CONVEX UNIX file system is a hierarchical structure made up of directories and files to ensure optimum flexibility and ease of use.
- The kernel is the innermost layer of the operating system. It manages all resources of the computer system, including interrupts, memory management, process control, and I/O.
- Disk Striping for maximum I/O thruput.
- Fundamental to the development of CONVEX UNIX is the CONVEX C compiler. The CONVEX C compiler is an industry-standard compiler based on the C programming language developed at Bell Laboratories.
- Easy-to-use flexible command language through the C and Bourne shells allows for rapid program development.
- Subroutines written in C, Fortran, and assembly language can be linked together to make one program.
- The product is fully documented. Manuals include the *CONVEX UNIX Programmer's Manual, the CONVEX UNIX Tutorial Papers,* and the *CONVEX UNIX System Manager's Guide,* among others.

4.1.2 Description

CONVEX UNIX sets a new standard for efficiency and ease of use on supercomputers. The CONVEX UNIX operating system retains the features that have made UNIX one of the most popular operating environments in the world. In addition, CONVEX has enhanced this system to provide the additional features needed by scientific and engineering users.

CONVEX UNIX is a demand paging virtual memory operating system. In addition to supporting the full complement of up to 16 megawords (128 Mbytes) of physical memory, virtual memory support enables programmers to use more memory than is physically available. Programmers need not concern themselves with physical memory limitations; they can concentrate on applications program development. The virtual address space for the C-1 is 4 Gigabytes, evenly distributed between system and user space. Performance and security of the virtual memory system are further enhanced by the hardware paging system and support for the hierarchical memory protection system. These advanced features of CONVEX UNIX coupled with the hardware provide the user with a highly productive development environment that is optimized for total system performance.

The hardware architecture of the C-1 has been designed to address specifically total system performance and throughput. The C-1 is composed of a high performance CPU and from one to five Input/Output processors (IOP's), that make up the intelligent IOP subsystem. CONVEX UNIX takes full advantage of this underlying architecture with the kernel distributed between the CPU and the IOP's. Each IOP contains a real-time executive and peripheral device drivers and controls and manages all peripheral device functions: device addressing; interrupt handling; overlap seeks; data transfers, and so on. In other words, the CPU is not impeded handling mundane I/O. This effective use of distributed processing

means that more computational power is devoted to solving application problems quickly and efficiently. Further, asynchronous I/O is supported that enables a user program to issue an I/O request and to proceed with program execution.

CONVEX UNIX also manages multiple concurrent processes for efficient sharing of the computational power of the C-1 supercomputer system. User-level preemptive priority scheduling of processes allows users to manage the work-load on the system efficiently. This scheduling method allows interactive computer users to remain productive even when large batch processing jobs are also running on the system. The amount of computational time allocated to the batch jobs can be controlled. This capability, along with facilities for establishing multiple batch streams enables effective use of the CONVEX C-1 supercomputer.

4.1.3 CONVEX UNIX Operating Environment

The CONVEX UNIX operating environment avoids some of the traditional distinctions between system programs and user programs, between operations performed interactively from a terminal and operations under program control, and between files and peripheral devices. By avoiding many of these traditional distinctions, CONVEX UNIX provides a highly productive environment for users and programmers.

The UNIX command interpreters, called shells, can operate in both a command mode and in a program language mode. The most simple function of the command processors is to provide a method for users to issue commands to the system. In UNIX, a command has a very simple structure: a command name possibly followed by one or more parameters. In one sense, commands are like subprograms. However, unlike subprograms, the number and types of parameters may vary, so a command can handle a broader range of capabilities and options.

The shells also allow programmers to combine commands together in two ways. In addition to using commands interactively at a terminal, programmers can combine commands as part of a command-level procedure located in a file. These procedures are called shell scripts. Shell scripts appear to the user of the system as just another command. The shells also provide control flow statements that allow users to write programs that are composed of high-level commands.

Another method of combining commands together to form more powerful or more specific commands is piping. Pipes provide a simple, yet powerful, data-flow model for connecting the output of one command to the input of its successor. In this way, simple commands can be efficiently combined together with pipes to perform higher level operations.

Much of the power of the CONVEX UNIX operating system is derived from a large number of small programs that have been developed to aid in the day-to-day operation and use of the C-1 computer. With the features provided by the shells, programmers can use these small programs, called tools, independently or in combination with other tools. In fact, much of the success of UNIX is derived from this facility: users can combine tools and do not have to develop new programs for each specific application task.

4.1.4 CONVEX UNIX Utilities

CONVEX UNIX provides over 200 tools to help users code, test, debug, and control their programs. To aid in program development, CONVEX supplies several text and screen-oriented editors, including *vi*, for source code creation; a loader, and a revision control system--rcs--for source-code maintenance. As an additional product, CONVEX also supplies the CONVEX CONSULTANT--for profiling program execution and debugging. The CONVEX language translators--Fortran, the assembler, and C-- are fully integrated and supported by these program development tools.

4.1.5 The File System

CONVEX UNIX provides a file system for managing and controlling information. The file system consists of a single tree-structured name space for user and system software components. Unlike some systems, the file system makes no assumptions about file structure - that is left to the programs. The CONVEX UNIX file system provides a highly uniform name space by minimizing arbitrary distinctions and special cases. For example, directories within the file system are also files; tools that operate on files can also be used on directories when appropriate, and peripheral devices are in the file system name space. This uniformity and consistency of name space frees user programs from much of the concerns of dealing with all of the special cases found in many file systems.

4.1.6 Disk Striping

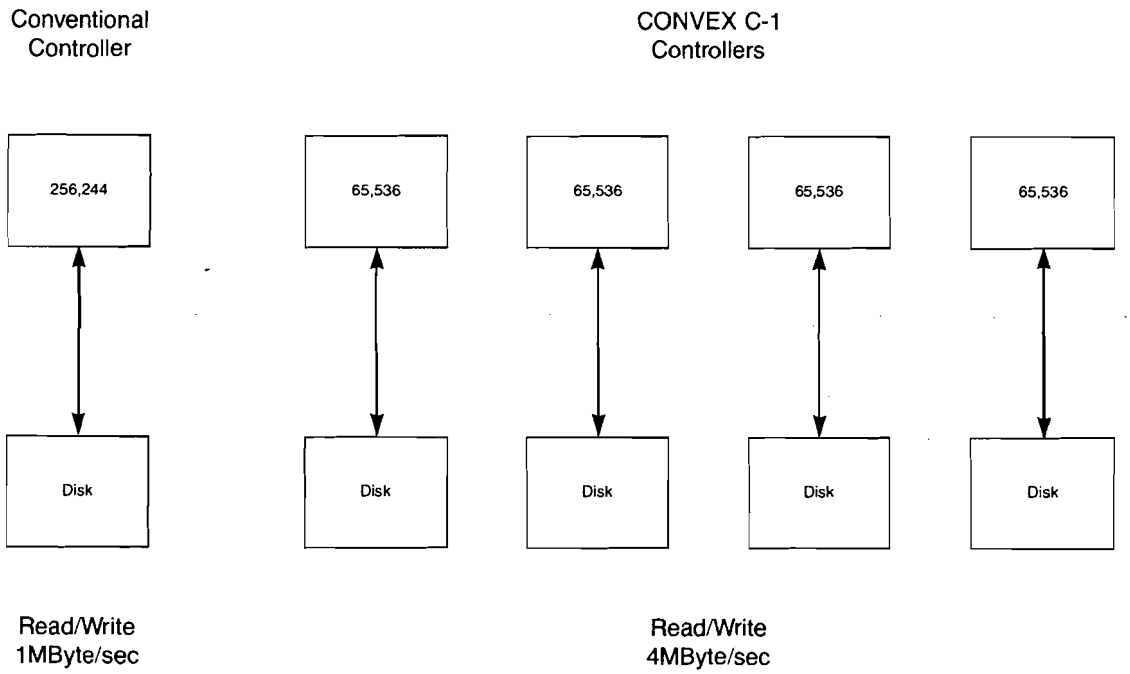
Conventional Unix file systems split each physical disk into one or more logical disk partitions, usually named *a* through *h*. The system manager has relatively little flexibility in defining a disk layout. In many cases, despite the best efforts of the system manager, the disk traffic is very unevenly distributed among the available disks. Carried to an extreme, uneven disk load distribution can cause disk throughput to become the limiting factor on system performance, even for jobs which might otherwise be CPU bound.

A less serious problem is that disk partitions are often of less than optimal size for a given user environment. The typical Unix system allows relatively little flexibility in disk partition layout.

These two problems may have a negative impact on the efficiency of the system. As a result, the productivity of the system's users may suffer.

The CONVEX Unix disk striping capability addresses these problems by allowing the system manager much greater flexibility in the layout of the disk system. A striped disk partition may span two or more conventional disk partitions, and may be split across multiple physical disk drives. The drives which contain these partitions may be on different Multibuses on the same IOP, or even on different IOPs. The splitting of a logical disc partition over several multiple discs, multibuses, and IOPs results in greater concurrency of I/O operations, and higher performance in most cases. (Refer to Figure 3-1.)

Figure 4–1: Conventional versus Striping



Striped file systems may take advantage of the increased disk basic block size feature of CONVEX Unix for additional performance gains. Disk partitions may be defined as normal disk partitions, striped disk partitions with nonstandard basic block sizes, or a combination of these.

Any program which runs under CONVEX Unix may take advantage of striped file systems without the need for special coding. No special source code modifications are required.

4.1.7 Advantages of Disk Striping

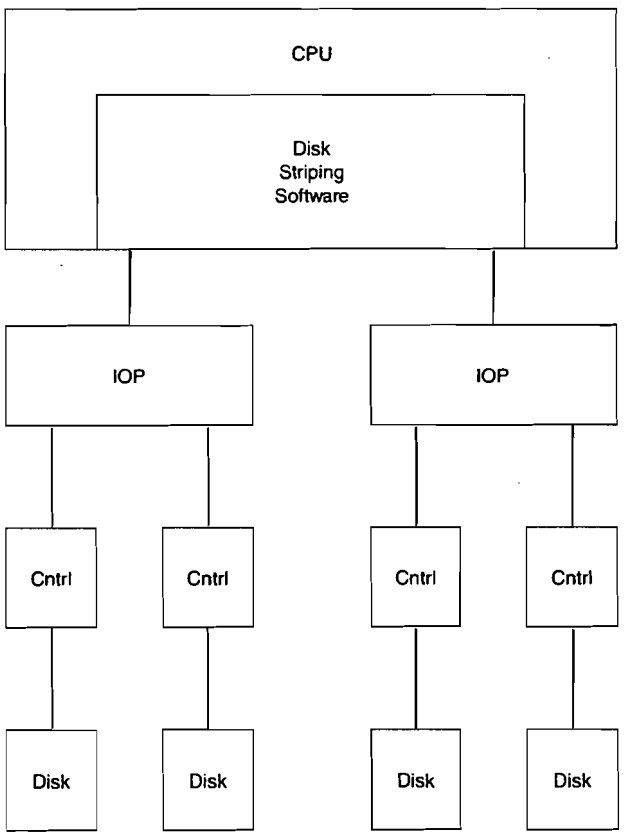
There are two primary advantages to using disk striping. The most important one is the capability to increase the performance of the disk subsystem. Although disk striping does not guarantee increased performance, it is possible to achieve a significant increase in overall disk throughput through the intelligent use of striping. It should be stressed that the actual difference in system throughput is highly dependent upon the particular system configuration and application.

The other major advantage of the use of disk striping is flexibility in disk layout. It is possible to create a single striped partition which holds more data than any single physical disk drive can hold. By combining two or more physical disk partitions (potentially of different sizes) you can create a striped disk partition which has the same size as the sum of the sizes of the conventional disk partitions. For example, with DKD-101 disks, you could combine two "a" partitions (~17 Mbytes each) and an "h" partition (~106 Mbytes) to create one striped partition which holds ~140 Mbytes.

4.1.8 Multiple IOP Support

Disc striping is completely transparent to the UNIX user and requires no source code changes. CONVEX UNIX will handle all the data repartitioning automatically. This includes full support for systems with multiple IOPs, where further concurrent I/O performance is possible.

Figure 4–2: Multiple IOPs



4.2 Asynchronous I/O

Further performance enhancements to the file system are provided by the asynchronous I/O feature. Asynchronous I/O permits concurrent disk and tape I/O so that the applications program does not have to wait for the completion of the I/O to continue executing. This is an important capability in many key applications areas such as geophysical processing and finite element analysis. This I/O structure fully utilizes the CONVEX parallel I/O structure based on independent and asynchronous Input Output processors(IOPs). The benefits to the include improved system throughput because I/O transfers overlap with the CPU and I/O processors.

4.3 The CONVEX C Programming Language

C is a general-purpose high-level programming language that has been used extensively for systems programming. Although it has been associated with the UNIX operating system, it provides many of the language features of other modern high-level languages. Subroutines written in C can be linked with both Fortran and assembly language routines to make one program. The CONVEX C compiler is an extended implementation of the C language originally defined at Bell Laboratories.

The data types and control structures of the C language are efficiently supported by the CONVEX architecture. One significant example of the implementation of systems programs in C is the CONVEX UNIX operating system. The operating system kernel and its associated utilities have been implemented in C.

The C language provides a variety of control-flow statements that allow for the development of well-structured, readable programs. Decisions can be made using the if statement with optional else clauses or with the switch statement which transfers control to one of several cases depending on the value of an expression. Looping constructs are supported by the for, while, and do statements. With the for and while statements, the loop termination condition is tested at the beginning of the loop. With the do statement, the loop terminates at the end.

CONVEX C directly supports integer data types of several different sizes. Integers can be declared as "short int" (16 bits), "int" and "long int" (32 bits), and "long long int" (64 bits). In addition, C supports a character data (8 bits) that can also be used for 8 bit integers. Floating point numbers are also supported in two different sizes: "float" (32 bits) and "double" (64 bits). There are also enumerated data types, structured data types, pointers, and arrays.

Unlike many other high-level languages, the C language does not define I/O operations. However, there are extensive facilities for performing I/O operations and accessing other operating system services with the standard CONVEX run-time libraries.

An additional utility program, called *lint* applies strict rules for checking many aspects of a program for correctness. In addition, the CONVEX C language is fully supported by the CONVEX symbolic debugger (csd).

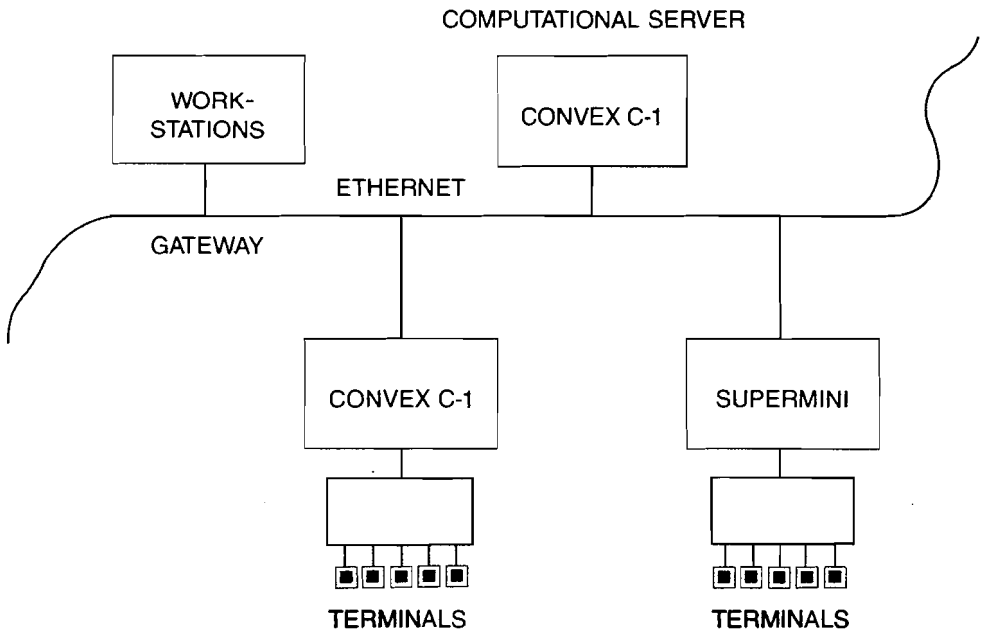
The CONVEX C provides an excellent language environment for the implementation of systems programs on the C-1 computer system.

4.4 Networking

Local area networks (LANs) have evolved rapidly in the last few years. The rapid development of LANs is driven by the need for efficient information exchange in a local computer environment. Since more computer systems are being installed, computer users both need and expect to have reasonable methods of exchanging information between these potentially heterogeneous computer systems.

The CONVEX network architecture has been specifically designed to address the growing need to share information. This integrated distributed supercomputer network will address the needs of the future as well as current requirements. This distributed networking system not only enables C-1 systems to communicate and share data; it also allows the C-1 to be a computational serving supercomputer in a network of workstations and minicomputers. Refer to Figure 4-3.

Figure 4–3: Networking



The CONVEX networking product, LAN-001, is a local area network based on the industry standard Ethernet. The communications protocol is IP/TCP(Internet Protocol/Transmission Control Protocol). IP/TCP provides an industry and DOD standard method of medium performance communication amongst multiple host computer systems. Communication amongst the hosts can involve file transfers, mail, virtual terminal services, remote execution, etc.

The CONVEX Product LAN-001 enables the user to do the following:

- Network a C-1 to other C-1s
- Network a C-1 to other systems supporting IP/TCP running UNIX 4.2
- Network to a VAX running VMS using the IP/TCP software

LAN-001 includes everything needed to connect a C-1 to an existing Ethernet network, specifically:

- Network Software Utilities
- Documentation
- Multibus Controller
- Transceiver
- Transceiver Cable

4.4.1 Network Commands/Utilities

Below is a brief description of some of the commands and utilities supported by LAN-001:

ftp - File Transfer Program. This utility allows a user to transfer files between host computer systems.

rcp - Remote Copy. Copies files from host to host.

rlogin - Remote Login. This connects a user terminal on the current host system to a remote host system.

rsh - Remote Shell. Connects to a specific host and executes the specific command.

TELNET - User interface to TELNET protocol. This is used to communicate with another host using the TELNET protocol.

TELNET and ftp enable networking with VAX/VMS, and appropriate IP/TCP software, systems. A C-1 terminal user can login to VMS using the TELNET utility command, or visa versa. Or the user can transfer a file using the *ftp* utility.

4.5 Fortran

CONVEX Computer Corporation's Fortran compiler is one of the most sophisticated compilers available. Designed to ANSI '77 standards, the CONVEX Fortran compiler is a multipass, optimizing, vectorizing compiler that includes both language and run-time library (RTL) extensions. The compiler will process existing Fortran programs without modification to the source, to take advantage of the full performance potential of the C-1 supercomputer's architecture. As a result, CONVEX Fortran increases programmer productivity, and maximizes maximum software portability and speed of execution.

4.5.1 Features

- CONVEX Fortran is an ANSI standard Fortran '77 compiler with optional support for programs conforming to the previous standard ANSI X3.9-1966.

- VAX-11 Fortran source code compatibility allows existing VMS programs to be easily ported.

- CONVEX Fortran offers optimization and vectorization as major features. Classical global and local optimization and vectorization of object code mean significant improvement in performance. The optimizing compiler performs dataflow analysis on iterative sequential procedures to produce parallel executable code that fully utilizes the integrated vector processing capabilities of the CONVEX hardware. In essence, the CONVEX Fortran compiler makes the hardware run faster.

- After compilation, a vectorization summary is generated that provides the programmer with information relating to the vectorized Fortran.

- Three levels of optimization are available with the CONVEX Fortran compiler: levels 0, 1, and 2. Level 0 is a set of local optimizations that are performed where 'local' means a sequence of consecutive code with one entrance and one exit. Level 1 is global optimization which refers to optimizations performed over the entire program unit, in particular IF statements and loops. Finally, level 2 is vectorization optimizations which replace DO loops with equivalent intermediate vector code. All three levels of optimization are object code compatible, which means that a programmer can link and execute programs that have been compiled with any combination of optimization level.

- No special vector syntax is required to utilize this highly advanced integrated vector processing architecture and the automatic vectorizing compiler fully. In other words, the user writes in standard Fortran for maximum programmer productivity.

- The user can specify compiler directives that provide information to the compiler to override conditions that inhibit optimization or vectorization. For example, the NO_RECURRENCE directive instructs the compiler to vectorize the following DO loop which would not normally vectorize because of an apparent recurrence. These compiler directives result in enhanced

program execution efficiency.

- Vector intrinsics are provided with the compiler. These are intrinsic functions within a vectorized loop, which are faster than the corresponding scalar intrinsics in a loop.

- Fully integrated with the CONVEX UNIX environment, the shareable, reentrant compiler operates under the CONVEX UNIX operating system to take full advantage of the C-1's architecture and virtual memory system, and the wide range of UNIX utilities.

- The CONVEX Fortran compiler has been designed to anticipate Fortran 8x, the proposed ANSI standard for a vector Fortran syntax, when it is introduced.

- CONVEX Fortran extensions include a wide range of data types with automatic vectorization across all numeric data types.

- The CONVEX CONSULTANT program development tools, which include a run-time performance analyzer and the interactive high level symbolic debugger, csd, are fully integrated with the CONVEX compiler to simplify program development.

4.5.2 Optimization and vectorization

The CONVEX Fortran compiler's optimizations help to produce code that results in enhanced performance levels. Optimization involves the careful manipulation of operations in the source programs being compiled--essentially, data flow analysis is performed on iterative sequential procedures to produce parallel executable code. The result is an object program that can run more efficiently.

The CONVEX Fortran compiler uses both machine independent and machine dependent optimization techniques, where:

- Machine-independent optimization techniques are performed at the source program level and do not depend on the object language to be produced.

- Machine-dependent optimization techniques improve the efficiency of the object code by making the best use of the machine language and the underlying architecture of the machine, including the integrated vector processor.

To achieve the highest performance possible, the compiler must perform many sophisticated scalar and vector optimizations.

The compiler performs typical scalar optimizations, like constant propagation and folding, common subexpression elimination, strength reduction, dead code elimination, code motion,

precomputed subroutine argument packets, and many others.

After scalar optimizations are performed for all code, vectorization is performed. The compiler invokes a vectorizer, which replaces DO loops with vector code. Assembly language that makes use of vector registers and instructions is later generated from the intermediate code by the compiler's code generator.

Vectorization is an optimization that identifies DO loops and then performs a dependency analysis to determine what can be vectorized.

Among the vectorization optimizations performed by the CONVEX compiler are:

- Vectorization across all nested loops
- Loop interchange
- IF statements within DO loops
- Recognition of reduction operations (e.g. SUM, PRODUCT, AND, OR, EXCLUSIVE OR, MAX, MIN, POPULATION COUNT)
- Recognition of scatter/gather (vector of indices)
- Recognition of vector edits (MASK, MERGE, and COMPRESS)
- Partial vectorization of statements (partitions a loop into two or more loops, one which is vectorizable and one which is not)
- Strip mining (vector operations into 128 element groupings)
- Vector register optimizations
- Vectorization for logical, integer, floating point (single and double) and complex data types

Further machine-dependent optimizations are used to enhance the object code produced by the compiler. Typical examples include register allocations to maximize the number of registers in order to achieve more parallelism, and strength reduction operations on instruction-level operations.

One of the final optimizations performed is instruction scheduling. This machine dependent optimization determines an order of instructions which effectively uses the asymmetric processing architecture and pipelining of the C-1 system. Instruction scheduling results in a high level of execution concurrency and chaining of vector operations. A user's application program can automatically achieve peak C-1 performance in excess of 60 million operations per second.

4.5.3 Data Types

Standard and extended data types allow for total flexibility. In addition to the standard data types of LOGICAL*4, REAL*4, REAL*8, INTEGER*4, COMPLEX*8, and CHARACTER, seven new data types are provided:

- LOGICAL*1
- LOGICAL*2
- LOGICAL*8
- INTEGER*1, (BYTE)
- INTEGER*2
- INTEGER*8
- COMPLEX*16

4.5.4 CONVEX Consultant

The run-time performance analyzer and interactive source level (csd), are made available through the CONVEX CONSULTANT. To help the user enhance the performance of Fortran applications on the C-1, the CONVEX CONSULTANT provides an execution profile/analyzer as a tuning aid. Typical profile information for a subroutine includes the amount of CPU time required for each subroutine, the percent of overall program execution assigned to subroutines, the number of times each subroutine is called and from where. The CONSULTANT provides a complete run-time diagnosis of the application program execution.

The CONVEX CONSULTANT also includes an interactive screen oriented source level debugger. The CONVEX source level debugger (csd), provides a rich repertoire of commands that enable such capabilities as variable or source line trace, support for the breakpoint instruction, symbolic access to program variables, single step, and edit capability.

4.5.5 Portability and Support

ANSI '77 CONVEX Fortran with VAX/VMS extensions allows users to port existing Fortran applications programs with minimal effort. Through the many features provided and by compliance to industry standards, the CONVEX Fortran compiler assures maximum portability of programs written on other computer systems.