COMPUTER CORPORATION OF AMERICA

DATACOMPUTER PROJECT

SEMI-ANNUAL TECHNICAL REPORT

July 1, 1975 - December 31, 1975

Contract No. MDA903-74-C-0225

ARPA Order No. 2687

Submitted to:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

Computer Corporation of America
575 Technology Square
Cambridge, Massachusetts Ø2139

DATACOMPUTER PROJECT
SEMI-ANNUAL TECHNICAL REPORT

July 1, 1975 to December 31, 1975

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by the U.S. Army Research Office, Defense Supply Service -- Washington under Contract No. MDA9Ø3-74-C-Ø225. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Table of Contents

1 Introduction

This report describes our work on the Datacomputer system from July 1, 1975 to December 31, 1975. The project is supported by the Information Processing Techniques Office of the Advanced Research Projects Agency of the Department of Defense. The current work is being carried out under contract MDA9Ø3-74-C-Ø225. Related work discussed herein is supported by the Nuclear Monitoring Research Office of ARPA under contract MDA9Ø3-74-Ø227.

Work during the reporting period falls into two main categories: installation and operation of Datacomputer Version 1, the first full service version of the Datacomputer; and preparation for the next version, which is to incorporate an Ampex Terabit Memory System. Parallel operation of Version Ø/11 continued into this period, and various other activities received attention as will be reported.

Sections 2 - 7 provide the detailed descriptions of this work. Section 2 is a discussion of the Datacomputer architecture, with emphasis on the increasing levels of functional abstraction beginning with the hardware and moving outward. Section 3 is a report on the usage of the Datacomputer during the reporting period. Section 4 is a detailed discussion of the work on the Datacomputer and user support software. Section 5 discusses the on-going work of documenting the Datacomputer. Section 6 describes Datacomputer site, hardware, and operations work. Section 7 is a brief overview of the NMRO work and its implications for the Datacomputer in general. Finally, Section 8 reports various miscellaneous but important areas of Datacomputer activities.

2 System Description (1)

The Datacomputer is a very large scale data storage facility with substantial data-management capabilities. Its design is optimized for use as a data resource in a network of large-scale computers which are connected via medium speed (50,000 bits/second) communications lines. The data storage functions of the Datacomputer will support the storage of files as large as a trillion bits, and the hardware facilities will include a device with appropriate storage capacities (currently, an Ampex TBM is being installed; other mass storage systems could also be used).

The development of the Datacomputer has been strongly affected by its nature as a network data utility. Its design does not preclude very fast data transfers -- the Datacomputer can feed data to the network or some other interface at speeds approximating the bandwidth of its storage devices, as long as no special processing is required -- but this is not how it operates in the most frequent case. The combination of very large storage capacities and only moderately fast communications facilities implies the Data-

(1) The preceding Semi-annual Technical Report contained a lengthy tutorial section titled "System Description," regarding which a number of positive comments were received form readers. In the interest of making the present document self-contained, a somewhat shortened and updated version of the same material is included here.

computer must provide powerful facilities for data selection and subsetting, in particular to minimize data transmission back and forth through the network. (To transmit a trillion bits through a 50,00 bits/second channel requires about 231 days, assuming no errors or interruptions!)

The Datacomputer's existence in a network environment also implies that other computer systems mediate between the Datacomputer and its ultimate human users. Consequently, functions that are not intrinsic to data management, such as carefully human-engineered terminal interfaces, are relegated to those other systems. This has several benefits: it allows work at the Datacomputer to concentrate on issues intrinsic to data storage and management. It allows other systems which concentrate on human engineering to provide better interfaces than would otherwise be produced. Most importantly, it ensures that the Datacomputer is developed as a resource available to other systems, capable of being incorporated into larger projects.

2.1 Levels of Functional Abstraction

Many large computer systems may be usefully examined in terms of their functional hierarchies. A level may be characterized in two ways. First, more fundamental operations which are provided by the previous level (and may already be abstractions themselves) are combined into new, more powerful, and more abstract operations. For example, the stream of magnetic flux reversals seen by the disk controller becomes a stream of fixed (or variable) length blocks of binary words when seen by the operating system. A subset of this arbitrary collection of unformatted words is presented to user programs as a "file" in a "file system". Second, intermediate functions exist to prevent certain combinations of operations which would damage system integrity from occurring, and to hide other functions entirely from the next level out.

The term normally used for the particular collection of functions available to any given level of a system hierarchy is "virtual machine". In many ways, the programmer working at level _n_ in such a system may behave as if level _n_-1 were hardware; All _n_-1 functions are immutable and part of the machine environment. Using terms that will be explained in the rest of this section, the TENEX implementor programs a PDP-10; the SV programmer programs a TENEX (which looks a lot like a PDP-10 with some major abstractions); the Request Handler programmer programs an SV machine, and the ultimate user programs a Datacomputer. (We shall see that the set of functions presented by the Request Handler is equivalent to the Datacomputer virtual machine.)

The following levels will be discussed in detail:

1) The hardware consists of a Digital Equipment Corporation (DEC) PDP-1Ø and its supporting peripherals, communications links to terminals and the ARPA Network, and a very large storage device.

2) The TENEX operating system is in direct control of the hardware resources and provides many services to the Datacomputer.

3) The programs known collectively as SV or Services are a pseudo-operating system which interacts with TENEX, managing input/output, scheduling, and storage strategies for Datacomputer files.

4) The Request Handler (RH) is the interface to external processes using the Datacomputer. It accepts control and data-management statements in "Datalanguage", provides messages concerning the state of the Datacomputer job to the user, and supervises data flow in both directions under the control of Datalanguage statements, and with the help of the other levels of the system.

Above these four levels of processing, there are an indefinite number of levels of functioning, outside the actual Datacomputer. These are the processes on other machines which interface to the Datacomputer, and co-operate with it in the accomplishment of whatever tasks its ultimate users undertake.

2.2 The Hardware Level

Conceptually, the hardware for a Datacomputer is quite simple. A processor of some sort is required along with some form of primary store (e.g., core). In addition, one needs a very large store (e.g., TBM) and a medium-to-high speed communications port. A great deal of efficiency can be gained by adding one or more levels of intermediate storage such as disk.

The hardware base of the Datacomputer as it is currently implemented consists of a processor, an address mapping device, three levels of store, medium and low speed communications lines, and several I/O devices.

The processor is a Digital Equipment Corporation PDP-1Ø. CCA has a KA-1Ø CPU which is the oldest of several models of PDP-1Ø processor currently available. A Bolt Beranek and Newman "Pager" provides address translation for all memory references, and (along with software in TENEX) provides the illusion of a 256K (1K = 1024) word primary store regardless of the size of the physical memory.

The real primary store in the current Datacomputer is 336K words of 36 bit core memory. This includes five 16K DEC ME-10's and two 128K STOR-10's from Cambridge Memories, Inc. PDP-10 characters are typically stored five to a word, so this is the equivalent of slightly more than a million characters of memory.

The system has two types of secondary store. Six spindles of DEC RP02 disk (IBM 2314 equivalent) provide space for the TENEX file system. These hold about four million 36 bit words each, for a total of 24 million words. In addition, four spindles of CalComp 230 disk (IBM 3330 equivalent) are attached to the PDP-10 via a Systems Concepts SA-10 IBM data channel simulator. These disks each will store approximately 20 million words of data. Currently, they serve as the main file storage medium of the experimental Datacomputer service. Ultimately, they will serve as staging devices between the tertiary store and the PDP-10.

The main data repository of the Datacomputer is its tertiary store. The tertiary store planned for the CCA Datacomputer site is an Ampex Tera-Bit Memory (TBM); but the Datacomputer is designed to incorporate other types of tertiary store easily.

The set of such devices available today can be characterized as having very large storage capacities (several trillion bits), very high transfer rates (typically 6 megabits/second), very large physical blocksizes (on the order of a million bits), and unfortunately, very slow access times (tens of seconds on the average).

The Datacomputer´s communications equipment consists of a connection to an Interface Message Processor (IMP) which is in turn connected to the ARPA Network. The Arpanet connection is the Datacomputer´s only channel to the outside world. All Datacomputer usage consists of messages and data passed back and forth through this port. Nodes in the ARPANET are connected by 5Ø,ØØØ bit/second phone lines, so the combined traffic of all concurrent Datacomputer users cannot exceed this transfer rate (except for the special case of usage from another host connected to the same IMP).

2.3 The Primary Operating System - TENEX

The second level in the Datacomputer´s functional hierarchy is the TENEX operating system. An excellent overview of the nature and facilities of TENEX can be found in "TENEX, a Paged Time Sharing System for the PDP-1Ø." (1)

(1) Bobrow, Daniel G., _et al._, "TENEX, a Paged Time Sharing System for the PDP-1Ø," _Communications of the ACM_, V. 15, no. 3, March 1972, pp. 135 - 143.

2.3.1 TENEX Modifications for the Datacomputer

The virtual machine provided by TENEX - a PDP-10 arithmetic processor with full memory capabilities and file/process address space integration - has proved to be quite hospitable to Datacomputer development. However, a few changes to the TENEX monitor have been necessary to optimize the Datacomputer's performance. The Datacomputer's Network Control Program has been modified to optimize 32-bit data transfers. The scheduler was modified to give special considerations to the resource utilization patterns of the Datacomputer. For example, there is little urgency in the Datacomputer to satisfy highly interactive jobs, but it is necessary to respond promptly to events like high-speed disk operations. Higher throughput with reduced overhead is achieved by rescheduling at shorter intervals than in a normal TENEX, but giving larger quanta of CPU time when jobs are scheduled. Routines to support the Calcomp 230 disk drives have been added. When the TBM is installed, additional device handling code will be necessary.

2.4 The Pseudo Operating System - Services

The preceding two levels of the Datacomputer system were not products of the development effort being discussed. They are described here because an understanding of their functions and capabilities is important to understanding the functions and capabilities of the two outer layers of the system - Services and the Request Handler.

These two levels constitute what could reasonably be called the "Datacomputer proper", and are the primary output of the Datacomputer project. They are conceptually and functionally separate - to the point of having separate staffs. This section discusses the Services programs (hereafter known interchangeably, and in accordance with time-honored tradition, as SV).

SV functions as a pseudo operating system for the Datacomputer. It provides the basic functions of a traditional operating system in a form which is maximally convenient for the construction of a user-level Datacomputer interface (of which the current Request Handler is but one example). In particular, SV provides a specialized file system, stream oriented input/output facilities, and a set of scheduling/-monitor functions.

Access to SV functions for the Request Handler is via a special instruction known as "SVCALL". SVCALL´s exist to manipulate the state of Datacomputer files including reading and writing pages from them; to perform input and output over the Datacomputer´s Arpanet connections, and to handle special error conditions.

2.4.1 The SV File System

The primary function of SV is to provide a convenient interface to the data storage facilities of the Datacomputer: the tertiary store and the staging device. A Datacomputer file as seen by the Request Handler programmer consists of an arbitrary number of "sections" (or sub-files), each of which is an ordered set of pages. (For convenience, SV pages are the same size as TENEX pages - 512 36 bit words.)

2.4.2 The Directory System

The Datacomputer file system may be thought of as a tree-structured hierarchy. At the top of the tree is a node whose conventional name is "%TOP". There are two types of nodes in the directory system - terminal and non-terminal. Terminal nodes (files) contain only data, and non-terminal nodes (directories) contain other nodes which exist at a lower level in the tree. Node creation is independent of the intended use of the node. In other words, a node in the tree is created, then at a later time it is specified whether it is terminal (a file) or non-terminal (a directory). Levels of the hierarchy are specified as a list of names connected by periods, such as "%TOP.DFTP.CCA". In the example, %TOP and DFTP are non-terminal nodes, and CCA may be either terminal or non-terminal (in the example, not enough context is present to determine which).

In addition to maintaining the directory hierarchy, the directory system provides protection for contents of nodes, whether other nodes or data. This protection takes the form of a set of "privilege tuples" associated with each node. A privilege tuple describes two things: the set of privileges allowed (or denied) to the user accessing the node, and the specification of the class of users to whom this particular set of privileges applies.

Just to give the flavor of privilege tuple application, one tuple might specify that, for a particular node, a user may login to the node and create new nodes under it, but only if connected to the Datacomputer from socket number 1000001 on ARPANET host number 31. Another privilege tuple might grant the same privileges to any user who knows that the password is "WASHINGTON". A third might only grant read access to files under that node to users giving the password "DC". For a full discussion of privilege tuples, please refer to the latest Datalanguage manual.

This external view of the Datacomputer's file system - a tree-structured hierarchy with multiple protection classes enforced on each node in the tree - is dealt with transparently by the Request Handler. This means that the structure seen by, and the functions available to the ultimate Datacomputer user are essentially the same as those provided by Services to the Request Handler.

2.4.3 Access to Datacomputer Files

As mentioned above, a Datacomputer file is stored as an arbitrary number of sections, each of which is broken into 512 word blocks called pages. When the Request Handler wishes to access some page of a Datacomputer file, the following sequence of events must take place:

1) The file is opened. To open a file, RH supplies SV with the string representing the file's pathname in the Datacomputer file system (along with any needed passwords). SV determines that the current user is allowed to access the file in the manner requested (and the file exists), then returns a small integer, known as a Relative File Number or RFN. The RFN is the handle used by RH in all future references to the file until it is closed, at which time the RFN becomes invalid.

2) A buffer is allocated in the user process´s address space. Buffers are managed by SV, but their allocation, freeing, and use is under the control of RH. A buffer is exactly the same size as a Datacomputer file page (and of a TENEX page). The buffer is identified by yet another small integer returned by SV.

3) If the page is being read (data already exists and is being referenced), an SVCALL known as PGRD is executed. This takes the RFN of the file, the section number, the page number within the section, and the buffer number into which the page is to be read as inputs. After the call, the page is available in the buffer.

4) If the page is being created, data is first entered into the buffer by the Request Handler, then the page is written to the file by the SVCALL PGWR. Arguments are the same as with PGRD.

5) If the page is being modified, the sequence is PGRD, modify, PGWR.

6) When the Request Handler is through with the buffer and the file, the buffer is released by an explicit SVCALL, and the file is closed.

There are some problems with this file access strategy, primarily as a result of the fact that data pages lose their identity when in buffers. Since SV is perfectly happy to read the same page into two or more different buffers concurrently, it is possible for the Request Handler to unwittingly make two copies of a page; modify them both in the buffers; then write both back to the file, making the contents of the page at best uncertain. This case actually arose during the development of Datacomputer Version 1, and a non-trivial amount of time was spent locating and correcting the undesirable interaction.

2.4.4 The SV Input/Output System and Monitor

The input/output and monitor facilities provided by Services are fairly rudimentary when compared with the directory system. Input/output consists primarily of a set of connections to the ARPANET, with the ability to read and write buffers of data to/from a given connection. A special set of SVCALL´s are provided for communication with the Datacomputer operator´s console. The operator is consulted before particularly large requests are executed for user jobs, and certain kinds of messages about the state of the Datacomputer are routed there.

The Services monitor provides no particular facilities of its own, but is responsible for the creation/destruction of TENEX forks which represent particular Datacomputer sub-jobs. As users contact the Datacomputer via the network, they are assigned to a particular sub-job by the master process known as "Job Ø", which is just like any other Datacomputer process, except it has the monitor code enabled.

2.5 The User´s Level - RH

The "outermost" level of the Datacomputer is known as the Request Handler. RH is in some sense an application program, since it is possible for a reasonably naive user to interact directly with it, via a specialized data-management language known as "Datalanguage". It would not be unreasonable to consider Datalanguage as the Datacomputer´s order code.

Datalanguage and the Datacomputer were designed to be used by PROGRAMS running in other hosts Since all control interactions with the Datacomputer are expressed as strings of human-parsable ASCII characters, it is possible (and in fact the norm at CCA) for a human user sitting at a terminal which is capable of generating the ASCII characters to interact directly and successfully with the Datacomputer. To avoid the anthropomorphization which usually creeps into descriptions of machine-machine interactions, this section is written as if the Datacomputer user were a real human being at a terminal. The reader is cautioned to bear in mind the system is not intended to be used in this mode; that the Datacomputer is a resource for machines and their programs.

2.5.1 User-Datacomputer Interactions

The Datacomputer maintains one or more input/output channels for the user. These are called "ports". All Datalanguage interactions flow over a particular port known as the "default port" or the "Datalanguage port". This port is the connection established when the user first contacts the Datacomputer from the ARPANET. Data may flow over the default port or over auxiliary ports which are created by Datalanguage statements as the session progresses. It is preferable to use auxiliary ports for data for two reasons: first, only ASCII data may pass through the default port; and second, even though the data being passed is ASCII, care must be taken to insure that it contains no characters which are treated specially when passed through the default port.

Datalanguage statements fall into two categories - commands and requests. In general, commands control the state of the user's Datacomputer process; open and close files, create nodes, modify privilege tuples, *etc*. Requests refer directly to the contents of files. A large part of Datalanguage is devoted to the detailed description of the contents of files, and the Request Handler makes extensive use of such descriptions in planning its actions.

2.5.2 Request Handler Structure

When the user first connects to the Datacomputer, Services initializes a new Datacomputer process, then passes control to the Request Handler. RH does some initialization of its own, then asks SV for the next line of input from the Datalanguage port. If the input line is a command, it is executed immediately. Requests are compiled, then executed. There is no provision for storing requests in their compiled form. There are two reasons for this strategy: first, there is an assumption that two requests that are _exactly_ alike are very rare indeed; second, the compilation time is trivial when compared with the execution time of requests on really large files.

2.5.3 The RH Compiler

The Request Handler's compiler is invoked for most requests. (A special subset of easy-to-handle requests are interpreted by a special module known as "SLURP".) The compiler consists of three parts.

1) The first phase of compilation is handled by a routine known as the "pre-compiler". The pre-compiler takes the request as received from the

user, does validity/syntax checking, and produces a new representation of the request known as "intermediate language". Intermediate language consists of a set of functions which are an abstract description of the entire set of operations which are legal on Datacomputer data. These functions are essentially the low-level machine language of the Datacomputer. They represent elementary operations such as "move an item from container 1 to container 2" with appropriate ancillary information such as the type and location of containers 1 and 2. Most of the "smartness" of the Request Handler lies in the pre-compiler. It is completely responsible for the syntactic and semantic interpretation of user requests (but not their execution).

2) After the pre-compiler has abstracted and simplified the request, the intermediate language generated, and descriptions of the real files which are named in the request are fed to the rest of the compiler. This section is responsible for generating the instructions for actually moving data from one file (or port) to another under the control of the request. The output of this phase of the compiler is a data structure which contains all the messy loops, skips, and such for plowing

through and pulling the data specified in the format requested from the file. The descriptions of these operations are called "tuples".

3) Finally, the routines which actually execute the request on the data are, in some sense, part of the compiler. Many of the tuples have distinct sub-routines which are responsible for their execution, and those routines constitute both the run-time environment and part of the compile-time data base of the compiler. Because of the multitude of data-types, byte sizes, etc. allowed by the Datacomputer, each tuple has many "modes", which are identified by bits in the data structure. For any given request, a particular set of modes is used, and a particular subset of the tuple code is executed. The last phase of the compiler walks through the tuple list that defines the request, and extracts the instructions which perform the tuple functions as constrained by the active mode bits in the tuples, producing the final "compiled request", which is executed with the real data.

3 Datacomputer Usage

The uses of the Datacomputer began to show a certain maturity during the reporting period, with existing applications receiving the major allocation of resources. At the same time, there was a steady current of experimental usage, by researchers investigating how they might apply the Datacomputer to their own tasks.

3.1 Phaseout of Version Ø/11

The Ø/11 version of the Datacomputer continued in service into the month of October, to provide a period of parallel operation with Version 1, and to allow for transferring data from the one version to the other. This process involves some difficulties; since the storage formats between the two versions are not necessarily compatible, data must actually be copied into its new form. A special purpose program was written which connects the two versions of the Datacomputer through the network, while sending Datalanguage commands to both, to accomplish this transfer. Since there is not enough on-line storage for two copies of all data to be present simultaneously, a procedure was used which allowed space to be assigned to the new version as it became available on the old. All data was transferred without user intervention, with the exception of cases where users had more data than they would be allowed to store in the new system; these users were required to select and transfer their own data. Version 0/11 was retired by the end of October.

3.2 Regular Version 1 Usage

3.2.1 DFTP

The largest category of use of the Datacomputer was the DFTP system. This program, which was running at 14 hosts on the network by the end of the year, provided on-line file archiving services to 104 users in the Arpanet community, with more than 1700 files stored. The demand for this service had grown by early fall to the point that the Datacomputer could no longer rely on informal agreements and user self-restraint to keep usage within bounds. Consequently, strict allocation limits at the site and user levels were established and enforced with the conversion to Version 1.

3.2.2 SURVEY

The SURVEY application continued its usage of the Datacomputer. This system begins with a program at MIT-DMS which conducts regular surveys (approximately every 20 minutes) of all the nodes in the Arpanet -- 99 hosts and IMPs. The results of a day's surveys are stored at DMS until an off-hours period, when a second program is automatically triggered which sends that day's data to the Datacomputer. The file at the Datacomputer is available for consideration from other points in the net; there are programs at ISI, SRI-AIC, CCA, and DMS which access it in various ways.

The space limitations which affected DFTP use were felt by the SURVEY application as well. By the end of the year, it had become necessary to migrate data from several of the earlier quarters stored onto tape, pending space with the installation of a TBM.

3.2.3 IMP Data

The Network Control Center at BBN continued to use the Datacomputer for storage and retrieval of performance statistics on the sub-net of the Arpanet; approximately 200 million bits of the most recent sub-net traffic statistics were stored in the Datacomputer throughout this period, with new data replacing old within a fixed allocation.

3.2.4 Seismic Data

The real-time seismic data application, discussed in Section 7 below, is still several months in the future. However, significant utilization of the Datacomputer in this period was accounted for by various seismic activities. Two which seem most worthy of note are the storage of a file of approximately 118,000 seismic events, dating from 1900 (for use by seismic researchers), and storage of the text of Datalanguage data descriptions and expected requests (along with several generations of changes and comments) which will eventually be used in the main seismic application!

3.3 Investigation by Prospective Users

Prospective users of the Datacomputer continued to examine its suitability for their needs throughout the reporting period. A study of possible use by the NSA continued through the second half of the year. A group at the University of London studying methods of facsimile message transmission began investigation of the Datacomputer as the "store" part of a "store-and-forward" scheme in September, and continue their use. Personnel from the Network Information Center, the National Software Works project, and the Air Force Data Systems Design Center all expressed interest in eventual use of the Datacomputer on their projects. A group at the SRI Artificial Intelligence Center, working on a large scale study of data management systems which might be appropriate to a proposed Navy Command and Control project, began an intensive study of the Datacomputer, which was continuing at the end of the year.

4 Software Development

Datacomputer software development can be broken down into three broad categories: SV development, RH development, and development of support programs running as separate jobs to give users access to the Datacomputer.

4.1 Services

The Services group spent the second half of the calendar year working in several areas. Primary among these was the implementation and design extension of SDAX (see below). By year-end SDAX Version 1 was implemented and running in test versions of the Datacomputer. In addition, implementation of a number of support routines (utilities, drivers) was completed.

Other work in the Services area included: a substantially improved data-lock capability to prevent simultaneous updating of various system tables; and implementation of an "intelligent" systems tester which exercises critical Services´ pathways.

4.1.1 SDAX

The Special Disk Area Index module, known as SDAX, serves two critical functions. First, it controls the staging mechanism which brings required data pages from relatively slow tertiary memory (TBM) to relatively fast secondary memory (3330 and Tenex disk). Pages thus staged are kept on disk as long as they are in use, and are migrated back to TBM only after they are no longer needed by active users. The second function of SDAX is to provide a mechanism for permitting access to several versions of a file by any number of users. It does this by a complex map-chaining technique which currently permits multiple readers and a concurrent updater to access any number of active versions of a file.

A contemplated major enhancement of SDAX would permit multiple updaters (and readers) concurrent access to data-computer files. Substantial design work has been done in this area.

To review: SDAX Version 1 supports access to multiple versions of a single file from multiple readers and a single updater. It is essentially complete, with the exception of some TBM-dependent Tenex driver and recovery routines. Version 1 includes the following features:

- Staging of partial file read/write;

- Supports opening of multiple versions of same file w/ multiple readers and single updater (MVMR);
- Basic recovery: Code for warm- and cold-starts after errors;
- Page Mover, 333Ø optimized: Page Mover is a set of slave routines which optimize the moving ("sloshing") of large numbers of file data pages between the same or different devices, such as 333Ø-disk and TBM.

4.1.2 LOCKS

The routines which handle data locks were substantially re-worked in this period. Service routines use data locks to prevent simultaneous updating of the directory tree and various other system tables. The lock routines support four functions - lock initialization, shared lock, non-shared lock and unlock. Communication between the lock commands and lock routines is normally established through the UUO processor. When increased performance is desired, special "quick lock" macros are available which provide a fast inline lock/unlock capability.

4.1.3 New Tester

A major addition to Services´ checkout and debugging facilities was accomplished with the completion of a non-sequenced, semi-intelligent tester. This tester exercises large portions of Services´ code. It does not perform its exercises according to any predictable pattern, however. It is directed by a decision tree which is searched according to an essentially random sequence, and internally maintained status tables. This tester (QT) performs such operations as:

- Creating and deleting nodes and files;
- Opening, closing and allocating space on files;
- Reading, writing and validating patterned data to files.

4.2 The Request Handler

During the second half of the year, the work involving the request handler section of the Datacomputer system fell into three areas: software development, system integration, and design.

4.2.1 RH Software Development

Although no new features were scheduled for development just prior to the release of Version 1 of the Datacomputer, some time was devoted to the completion of tasks which were begun during the first half of the year and the implementation of enhancements. The majority of the effort was directed toward chaptered files and the file description mechanism.

The implementation of the chaptered file scheme, as discussed in the semi-annual technical report for the first half of the year, was completed. The Version 1 Datacomputer supports appending records to and removing records from a chaptered file as well as full variable length updating.

The description of every container in the Datacomputer is represented by a container descriptor node. A complete description for a file or port is a tree of container nodes. This tree is built at the occurrence of a CREATE or an OPEN command and is used by the compiler to generate the appropriate code for transmission and conversion of data. Each node of the tree stores the physical and logical attributes of one container, such as data type, data set, representation, byte size, length and count, etc. Previously each container descriptor tree was built in a double buffer. The size of a file or port was limited, in terms of the number of containers, by the number of tree nodes which could be built in a double buffer. A generalized buffer allocation scheme was incorporated into the description mechanism, thus eliminating this restriction.

The skeleton of the descriptor tree is built as the Datalanguage is parsed. The remainder of the tree is filled in with appropriate defaults and the container lengths are calculated from their descriptions. The length algorithm is complex, taking into account delimiters, count bytes and skipped bits at the end of each word, and recurses for each embedded container. Previously, due to extensive use of the system stack by the length algorithm, the complexity of containers was restricted, in terms of the depth of STRUCTs and LISTs. This depth restriction was also removed from the Version 1 description mechanism.

An inversion is a secondary data structure that the Datacomputer can use to improve its efficiency in retrieving data by content from a file. Specifically, an entry in the inversion is constructed for every container with the inversion option. For each data value which occurs for the container, the inversion contains pointers to all the outermost list members for which that container has that value. When an inverted file is created, storage space is allocated for the secondary data structure. Previously, unless an operator was present, the system supplied default inversion parameters. Three new options were added to the description mechanism so that the user can specify, at the time of creation, inversion size, number of hash pages and the number of hash slots per page.

The "LIKE" feature was also modified to simplify the creation of inverted files. If the target file is inverted, the newly created file, by definition, is also inverted. The new file now automatically uses the same inversion parameters as the target file, regardless of the way they were specified when the target file was created (operator supplied, description options, defaulted or copied as a result of a LIKE creation).

4.2.2 System Integration

One of the most important and time consuming aspects of releasing a new version of any system is system integration and shake down. Because large sections of the Datacomputer were rewritten and Datalanguage greatly expanded since Version Ø/11, extensive testing was performed.

A large number of test decks, developed for testing previous versions of the Datacomputer, were available. Many of these were modified to conform to the changes in Datalanguage. In addition, new test decks were written to exercise new features. A consistent testing environment was set up and all of the test decks run.

A formal bug reporting system was instituted. All bugs discovered by running the test decks, as well as those reported from other sources, were catalogued and assigned for debugging. Several problem areas were uncovered, some had been part of the system from previous versions and some newly created. Once the influx of newly reported bugs dropped off, the problem areas reconciled and all of the major bugs as well as most of the minor bugs were fixed, all of the test decks were rerun.

4.2.3 Design

Although the emphasis during this reporting period was on the Version 1 Datacomputer, some time was also spent thinking about future versions. A ´wish list´ was drawn up which included features desirable by both users and the system.

The request handler group concentrated on the file group feature which gives the user control over how to break up large volumes of data into physically smaller and more manageable units. Each unit can be individually accessed, dumped, validated, etc. At the same time, the user can create one or more groups out of a subset of these logically related units or subfiles. When a request is issued on a group, the system decides which subfiles must be accessed.

With the installation of a TBM at CCA in early 1976, the implementation of the file group feature was considered to be of high priority. Much thought and discussion was given to desirable characteristics of groups, the interface and amount of control exercised by the user and integration into the Version 1 software. A preliminary design was written, including new Datalanguage, affect on existing commands, modifications to system tables and impact on the compiler and run time environment.

4.3 User Support Software

As previously discussed, the Datacomputer serves primarily as a resource for programs and software systems written and residing on various other Arpanet hosts. Nonetheless, one subroutine-level and three user-level interfaces to the Datacomputer system are supported by CCA. These programs allow users at various sites to make elementary use of the Datacomputer´s features.

Work had begun in the previous reporting period on adapting and upgrading three of these programs to be ready to work with the new version of the Datacomputer. This work was completed in the early part of this period, and new versions distributed to users at a number of sites around the country.

4.3.1 DCSUBR

DCSUBR is a package of MACRO subroutines which user-level programs can call to manage their interactions with the Datacomputer (via the network). It had already been incorporated into several programs used from CCA and other hosts, and it was also becoming an example of what a Datacomputer interface should look like, even to programmers who were not using it directly.

A major cleanup of the package was initiated in the previous reporting period; some aspects of that effort contributed to the last stages of the design of version 1, offering a test case for various protocol options. As the Version 1 design was fixed, DCSUBR was then settled into a corresponding design. Also underway at the start of this period was an effort to instrument and optimize the different routines in DCSUBR, and in the RDC program in which it is used. The new version of DCSUBR was extensively tested, and then released to users across the network.

In addition to its incorporation in the program RDC (see below), DCSUBR is currently included in programs which process the IMP traffic data from BBN, in a FORTRAN interface developed for testing purposes at the NSA, and in a number of small programs developed for internal use at the Datacomputer. Among these latter, a significant use was a preliminary simulator for the Seismic Input Processor (SIP), which allowed testing of the Datacomputer's processing of seismic data to begin long before the SIP was ready to provide any. It was also used as a model for interfaces written in LISP, MUDDLE, and PL/1 at CCA, MIT-DMS, and MULTICS.

4.3.2 RDC

RDC is a program to Run the Datacomputer for a user at a terminal. Its basic function is to pass lines back and forth between the user and the Datacomputer, providing some (variable) screening of error/status messages on output, and standard type-in editing functions (such as character and line delete) on input. It also provides an interface to the TENEX file system, useful both for feeding files of Datalanguage to the Datacomputer, and for transferring data from or to devices other than the user's terminal.

Most of the work of adjusting RDC to deal with Version 1 was done in the preceding reporting period; what remained for this was to ensure its compatibility with the new version of DCSUBR, when that became available. RDC was distributed to 5 TENEX sites in this reporting period.

4.3.3 DFTP

This Datacomputer File Transfer Program provides the user with an on-line file archival facility. Files are stored as an entire unit without internal structure. DFTP provides a tree-structured directory. Files can be accessed at any time without operator intervention.

This style of use is most appropriate for seldom-used files when disk space on the host is either scarce or expensive. It also provides a means for sharing files in a network environment while avoiding the mutual distrust problems involved with FTP, where at least one user must have access to both the source and destination directories. Since none of the internal file structuring capabilities of Datalanguage are used, DFTP is a very simple, but very practical application of the Datacomputer.

DFTP underwent a major re-write and upgrade for release with Version 1 of the Datacomputer. It has now been distributed to 14 sites around the network (9 TENEX, 2 TOPS-1Ø, and the 3 ITS systems); an equivalent program for use from MULTICS was also written during this period. DFTP has been extensively used; by the end of the reporting period, it accounted for nearly half of the bits stored at the Datacomputer.

5 Datacomputer Documentation

5.1 User Manual

The most important effort in Datacomputer Documentation was the production of a new user manual for the Version 1 release. This project began in the middle of the previous reporting period, when we hired an outside technical writer, who began by familiarizing himself with Datacomputer concepts and facilities, and preparing draft descriptions of new features as their implementation was begun. By the start of this reporting period, writing was well underway, and members of the programming staff began to be involved in review and correction of the text. At the same time, effort was devoted to developing a concise chart of Datalanguage features and syntax, which now appears as Appendix A in the Version 1 User Manual. Final preparation of the document proceeded through the month of August, and it was returned from the printer in the first week of September. More than 200 copies had been distributed by the end of the reporting period.

5.2 Datacomputer Introduction & Philosophy

Another major effort was devoted to the predecessor to this report. It was decided to make the Semi-Annual Technical Report which approximately coincided with the first full release of the Datacomputer be a moderately lengthy discussion of the philosophy and design of the Datacomputer, as well as a simple listing of that term's accomplishments. This aspect of the report has been well received, and portions of that document have been carried over into the current report.

5.3 User Program Memos

Several user-support programs were rewritten to keep in step with the new version of the Datacomputer; the documentation for these programs was correspondingly updated. In particular, DFTP, the program which accounts for the greatest activity in the current Datacomputer, was extensively rewritten, and required a complete new document. DCSUBR, a collection of MACRO subroutines which provides an extremely convenient and general interface to the Datacomputer, was also extensively rewritten, and required a new document. This set of functional specifications and calling conventions has become a de facto Datacomputer interface protocol. Well over a hundred copies of each of these memos have been distributed.

5.4 Program Logic Description

Other documentation efforts were begun, but had not borne fruit by the end of the reporting period. Chief among these was the effort to update and extend system (Program Logic) documentation. Old memos were reread, expanded and corrected. Areas which had not been documented were identified, and a general scheme for organizing system documentation was proposed. Some internal memos were produced, which will be incorporated into the full documentation, and are available in the meantime to programmers using those elements of the system.

6 Hardware / Site / Operations

Several additions and improvements were made to the CCA computer site in the later half of 1976. Most of these were in preparation for the installation of the Ampex Tera-Bit Memory system (TBM).

6.1 Site Improvements

Starting December 8th, 1975, for one week, all computers at CCA's computer site were turned off while the most disruptive of the site enhancements for the TBM were performed. This included modification of the walls and ceilings and expansion of the false floor and utility connections. This shutdown affected the reliability of computers at the site for several subsequent weeks.

Only a few minor site enhancements remain to be done. A new air-conditioner has been received and is being installed.

6.2 TBM Negotiations

Delivery of the Ampex TBM, contracted for by CCA in a 2ØØ billion bit configuration, is now expected in February 1976. This represents a six month slippage Ampex, due to difficulties with the TBM-PDP-1Ø interface. Ampex reports that, at year end, the difficulties were being overcome.

6.3 TENEX Changes

During the second half of 1975 both hardware and software enhancements were made to the PDP-1Ø on which the Datacomputer runs. A second 128K word memory system was added, increasing core memory to 336K words, and TENEX was modified to be able to use more than 256K words The additional memory was required for TBM block transfers.

A facility was added to TENEX for reading and writing a track or cylinder at a time on 333Ø type disks. A DEC RP-Ø2 drive was also added to the system to relieve congestion.

Finally, routines were designed to greatly improve the efficiency of 32 bit ARPA network connections. These routines may be implemented and used for seismic data.

7 Seismic Data Base Support

As mentioned in the Introduction, some work on the Datacomputer is funded under a separate contract from the Nuclear Monitoring Research Office of ARPA. A short discussion of this work is included here because of its intimate relation to the work of the primary Datacomputer development contract. In particular, the TBM and additional core mentioned above are being paid for by the NMRO contract.

This work is specifically directed at establishing an online, real-time data base of seismic data from a worldwide network of sites, and making this data available to seismic analysts in various locations, and for various purposes.

7.1 Overview

Since the system will work in real time, the ARPA Network was chosen as the most appropriate communications medium available (as opposed to mailing tapes, high speed dial-up or leased lines, etc.). Seismic data is collected at the Seismic Data Analysis Center from sensors scattered all over the world, then transmitted from SDAC to CCA over the network. At CCA, a small computer known as the Seismic Input Processor, or SIP, absorbs the incoming data and stores it on its own disk. Periodically, the SIP connects to the Datacomputer (again via the network) and bursts the collected data into the Datacomputer at a very high rate.

7.2 IMP/TIP Considerations

There has been some question as to whether the data rates planned for the seismic data application are achievable through the local CCA ARPA Network node. This local node used to be a TIP, which had therefore to handle terminal lines as well as host and network lines. The TIP also had a Very Distant Host (VDH) connection that used a disproportionately high number of buffers.

To improve the performance of the CCA local network node, the TIP was replaced with a 516 IMP. This change not only removed the computational load from terminal lines but replaced the 316 TIP processor with the faster 516 processor.

7.3 Seismic File Descriptions and Efficiency

During the reporting period, the file descriptions to be used for the seismic data were refined, and it was found useful to utilize such fairly recent features of the Datacomputer as virtual indexes.

Some experimentation with typical seismic requests pointed out some deficiencies in the Datacomputer which have since been remedied.

8 Other Activities

8.1 VLDB Conference

CCA participated in the recent International Conference on Very Large Data Bases. Discussions of the Datacomputer were held, and a technical paper titled "Datalanguage: The Access Language of the Datacomputer" was presented by Jeffrey M. Hill.

8.2 NSW Planning Meeting

CCA participated in the National Software Works Conference Planning Meeting at Gunter AFB in November, with Datacomputer presentations to the NSW planning group and to data management personnel from the Air Force Data Systems Design Center at Gunter.

8.3 Visits to the Datacomputer Site

CCA continues to be host to people interested in the Datacomputer for possible future use, or in the development of Datacomputer applications. Elizabeth Feinler of SRI´s Augmentation Research Center came to discuss application to the NIC data bases. Peter Kirstein of the University of London came to discuss applications in the field of facsimile message processing. Yutaka Kuwahara from Hitachi

Corporation came to discuss the Datacomputer´s approach to issues of distributed databases. Earl Sacerdoti of the SRI Artificial Intelligence Center visited in November to discuss possible use of the Datacomputer for the Navy´s proposed Command and Control project. Personnel from SDAC and VSC visited on several occasions to co-ordinate preparations for the Datacomputer´s handling of seismic data.

8.4 Testing and Bug Monitoring

The efforts discussed in the previous Semi-annual Technical Report have been continued and expanded. The set of standard tests applied to new modules of the Datacomputer has been augmented by tests derived from actual user processing. A formalized system for reporting the existence of bugs and ensuring they receive proper attention was established in this period, and has expedited our response to error indications.

MFG. 1-74

CAUTION — REMOVE PROTECTOR SHEET BEFORE TYPING

"TO BE STORED IN A COOL DRY LOCATION"

MATERIAL INSPECTION AND RECEIVING REPORT	1. PROC. INSTRUMENT IDEN(CONTRACT) MDA903-74-C-0225	(ORDER) NO.	6. INVOICE NO. DATE	7. PAGE 1 OF 1; 8. ACCEPTANCE POINT D
2. SHIPMENT NO. CCA0010	3. DATE SHIPPED 5MAR76	4. B/L TCN	5. DISCOUNT TERMS	

9. PRIME CONTRACTOR CODE 6A046 Computer Corporation of America 575 Technology Square Cambridge, Massachusetts 02139	10. ADMINISTERED BY CODE S2202A Mr. J. McDonough Defense Contract Administration Services Region, Boston 666 Summer Street Boston, Massachusetts 02210
11. SHIPPED FROM (If other than 9) CODE FOB: Same as 9. above	12. PAYMENT WILL BE MADE BY CODE S2202A Disbursing Officer Defense Contract Administration Services Region, Boston 666 Summer Street Boston, Massachusetts 02210
13. SHIPPED TO CODE W73QQB Defense Advanced Research Projects Agy Architect Building 1400 Wilson Boulevard Attn: Mr. S. Walker Arlington, Virginia 22209	14. MARKED FOR CODE Same as 13.

15. ITEM NO.	16. STOCK/PART NO. DESCRIPTION (Indicate number of shipping containers - type of container - container number.)	17. QUANTITY SHIP/REC'D *	18. UNIT	19. UNIT PRICE	20. AMOUNT
0002 AB	Semi-Annual Technical Report December 31, 1975	14			NSP

21. PROCUREMENT QUALITY ASSURANCE

A. ORIGIN

☐ PQA ☐ ACCEPTANCE of listed items has been made by me or under my supervision and they conform to contract, except as noted herein or on supporting documents.

DATE | SIGNATURE OF AUTH GOVT REP

TYPED NAME AND OFFICE

B. DESTINATION

☐ PQA ☐ ACCEPTANCE of listed items has been made by me or under my supervision and they conform to contract, except as noted herein or on supporting documents.

DATE | SIGNATURE OF AUTH GOVT REP

TYPED NAME AND TITLE

22. RECEIVER'S USE

Quantities shown in column 17 were received in apparent good condition except as noted.

DATE RECEIVED | SIGNATURE OF AUTH GOVT REP

TYPED NAME AND OFFICE

* If quantity received by the Government is the same as quantity shipped, indicate by (✓) mark, if different, enter actual quantity received below quantity shipped and encircle.

23. CONTRACTOR USE ONLY