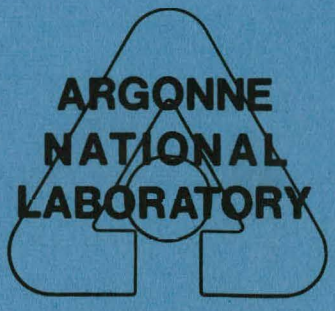
ARGONNE NATIONAL LABORATORY

GUIDE TO COMPUTING AT ANL

Edited by

Jim Peavler



APPLIED MATHEMATICS DIVISION

DISCLAIMER

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) among the U. S. Department of Energy, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

MEMBERS OF ARGONNE UNIVERSITIES ASSOCIATION

The University of Arizona
Carnegie-Mellon University
Case Western Reserve University
The University of Chicago
University of Cincinnati
Illinois Institute of Technology
University of Illinois
Indiana University
The University of Iowa
Iowa State University
The University of Kansas
Kansas State University
Loyola University of Chicago
Marquette University
The University of Michigan
Michigan State University
University of Minnesota
University of Missouri
Northwestern University
University of Notre Dame
The Ohio State University
Ohio University
The Pennsylvania State University
Purdue University
Saint Louis University
Southern Illinois University
The University of Texas at Austin
Washington University
Wayne State University
The University of Wisconsin-Madison

GUIDE TO COMPUTING

AT

ANL

Technical Memorandum 336
D R A F T

edited by

Jim Peavler

Contributors:

Al Burger
Paul Messina
Fred Moszur
Jim Peavler
Gail Pieper

*Work performed under the auspices of the
U.S. Department of Energy

PAGES ii to iii

WERE INTENTIONALLY

LEFT BLANK

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1

INTRODUCTION

Persons interested in using the computers at Argonne National Laboratory (ANL) need basic information about the physical layout and facilities of the computing center. The Guide to Computing at ANL is intended to provide this information, giving details about hardware, software, procedures, and services of the Central Computing Facility, as well as information about how to become an authorized user.

RELATIONSHIP TO OTHER DOCUMENTS

This document, along with the other new documents, Data Management at ANL (Tech. Memo. 338), Batch Processing at ANL (Tech. Memo. 337), and Graphics 1.0 (Tech. Memo. 335), replace the old ANL Computer User's Guide. We have reorganized the material presented in that guide, putting the most important and useful information first, so that you can find what you need to know in a logical sequence. We have also reduced the number of chapters, combining closely related topics and deleting others where a separate document seemed more appropriate (e.g., data management). In addition, we have reduced the level of detail in this introductory manual, to make the it more readable. The Guide to Computing at ANL assumes that you know basic computer and programming concepts; however, you need not be an expert in using assembler or compiler languages.

This guide also replaces the Interim Supplement, which provided information on the new procedures and equipment of the expanded Central Computing Facility. We have updated that information and revised various sections to reflect the most recent system changes.

The Guide to Computing at ANL represents a major change in our approach to documentation. In the past, changes in software, hardware, procedures, or services at the Central Computing Facility made it difficult to keep the User's Guide up to date. Rather than constantly revising the guide or issuing supplements, we have decided to limit this document to general information applicable to all the main processors and to all computer users at Argonne. For more detailed discussion of individual hardware and software sys-

tems in the Central Computing Facility, you are referred to other more specific documentation or to special reports, distributed by the Documentation Center (Building 221, Room A142B, 2-5406).

AMD will continue to publish The HEX, a monthly newsletter describing recent modifications to the Computer Center hardware, software, and administrative policies. Copies of The HEX are available from the Documentation Center, A142B, 2-5406, where you may also have your name entered onto the mailing list for the HEX. (Everyone having a computer account is automatically placed on the mailing list.)

ORGANIZATION

The first three chapters are introductory: Chapter 1 introduces the new Guide to Computing at ANL itself, Chapter 2 the hardware and operating systems of the Central Computing Facility (CCF), and Chapter 3 the various services offered by the CCF.

Chapter 4 describes accounting and rationing. The service rates are outlined, batch and TSO rationing is explained, and user responsibilities are discussed.

Chapter 5 discusses the various languages and compilers in use on the Central Computing Facility. We also make some suggestions for debugging programs.

Chapter 6 is devoted to libraries and Chapter 7 to applications packages available at Argonne.

ADDITIONAL REFERENCE MATERIAL

Each chapter also lists references to more detailed documentation on hardware and software systems in the Central Computing Facility.

Chapter 2

HARDWARE AND OPERATING SYSTEMS

Argonne's Central Computing Facility maintains and operates four IBM computers: a 3033, with ASP, batch, and WYLBUR timesharing; a second 3033, with CMS and batch; a 370/195, used for batch processing; and a 360/75, with IBM's TSO (TimeSharing Option). The processors have a total of 20 megabytes of main memory. Twenty-five channels (including selector subchannels) provide paths between the CPU's and the peripheral devices to keep the system performing satisfactorily, even under heavy utilization.

In addition, the computational resources listed in Table 1 are provided by the Central Computing Facility:

TABLE 1

ADDITIONAL RESOURCES

Disks	
No. of drives	97
Tapes	
No. of drives	24
Fixed-head storage	
No. of drives	5
Printers	
No. of printers	7
Nominal capacity (60-line pages/hr)	19,500
Timesharing access	
No. of ports with modems	197
No. of ports available for growth	107

To ensure an orderly flow of work through this computer complex, the Central Computing Facility uses several different control programs. The most basic of these is IBM's OS/MVT

Operating System/Multiprogramming with a Variable Number of Tasks). Each user-initiated task (e.g., a batch job step, a TSO command, or a WYLBUR command) requires the services of OS, or CS in conjunction with another control program such as ASP, VM, TSO, CMS, or WYLBUR.

Most of the time you need not be concerned with the problem of selecting a particular computer. ASP automatically routes jobs to the CPU that can provide the best turnaround, subject to the constraints of specific user requests, data accessibility, and job class. Nevertheless, if your programs use a lot of CPU time, you may want to experiment to determine which machine is more efficient. Should you find it desirable to specify a particular processor, you will have to include an ASP control card //*MAIN SYSTEM= in your job.

ASP will also handle remote job entry. A front-end system supports connections to national networks (e.g., ARPANET) and HASP workstations (remote job entry stations with one or more card readers and line printers), while remote batch stations are supported through the RADS (Remote Access Data Stations) stations.

This chapter is an introduction to the hardware and operating systems at Argonne's Central Computing Facility. Further information is provided in subsequent chapters and in the additional documentation listed.

PROCESSORS

The Central Computing Facility supplies four IBM processors.

IBM 3033's

Each IBM 3033 has 6 million bytes of main storage and access to nearly 6.7 billion bytes of direct-access storage.

The IBM 3033 computer is often compared with the 370/195. It runs small jobs and operating system overhead work faster than the 370/195, while the 370/195 runs "number crunchers" faster than the IBM 3033. However, the performance ratio can vary. The 3033 provides faster access to memory than the 370/195, but performs floating point operations more slowly. Furthermore, since the 3033 does not have parallelism in the execution unit, it is somewhat slower than the 370/195 in the actual execution of instructions.

In addition to running batch jobs, the 3033's also run the WYLBUR text-editing system and the CMS timesharing system. About two hundred dial-up lines are available (shared with TSO), and several transmission rates are supported, including 110 baud, 134.5 baud, 300 baud, and 1200 baud.

For further information on the 3033, see IBM 3033 Processor Complex and 3033 Multiprocessor Complex Functional Characteristics (GA22-7060).

IBM 370/195

The IBM 370/195 has 4 million bytes of main storage and access to nearly 6.7 billion bytes of direct-access storage.

The machine is used to process batch jobs, performing floating-point operations particularly fast. It is capable of executing from 1 to 8 MFLOPS (million floating-point operations per second), and 2 to 14 MIPS (million instructions per second), depending on the particular program structure. Part of this speed is attributable to the two-level memory structure. A buffer containing 32,768 bytes, running at 54 ns for each 8 bytes, is used by the hardware to contain the most recently accessed blocks of main memory. Main memory runs at 756 ns for each 8 bytes. Measurements have shown that when the information desired by a program is in core, it is available in the buffer 98 percent of the time.

The parallel organization of the 370/195 also contributes to its speed. Instruction fetch and decoding are performed independently from instruction execution, floating-point operations are executed independently from fixed point operations, and the floating point adds are executed independently from floating point multiplies. This organization allows many logically independent operations within a program to be executed at the same time.

For further information on the structure of the 370/195, see IBM System/370 Model 195 Functional Characteristics (GA22-6943).

IBM 360/75

The IBM 360/75 has two million bytes of main storage, 2 million bytes of slow core storage, and access to 1.8 billion bytes of direct-access storage. It can overlap main storage fetches and stores with instruction decodes and execution, using main memory, with a storage cycle time of 750 ns for each 8 bytes. It also overlaps instruction decoding with execution of previously decoded instructions.

The machine runs the timesharing option (TSO) of OS/MVT. Currently about 200 dial-up lines are available for timesharing.

ARPANET is also supported on the 360/75. The same ARPANET front-end implementation that is connected to the 3033 provides connection to the timesharing facilities of the Central Computing Facility. This allows remote ARPANET users to gain access to a front-end command process that will auto dial a local timesharing port. All of the normal WYLBUR and TSO commands can then be executed.

For further information on the 360/75, see IBM System/360 Model 75 Functional Characteristics (GA22-6889).

DIRECT-ACCESS STORAGE

User-accessible Drives

Three types of direct-access storage are accessible from the main processors:

1. IBM 3350-equivalent disk drives. Thirty-two ITEL 7330-12 disk drives have recently been added to Argonne's system. Each is functionally equivalent to the IBM 3350 disk units and has a track capacity of 19,069 bytes. These are Argonne's fastest drives, with a seek time of less than 40 ms.

2. IBM 3330-equivalent disk drives. Smaller than the 3350's, with a 13,030 byte track capacity, these drives include 14 IBM 3330's and 58 ITEL 7330-Model 10's and 11's. Both single and double density packs are available.

3. IBM 2314 disk drives. Shared by the 3033 and the 360/75, these devices require the use of the channel for the entire time that a record is being located and read.

In addition, a fourth type of direct-access storage is available to only the 370/195:

4. IBM 2321 data cell drives. The slowest of the disk drives are the IBM 2321 data cells. A data cell is a direct-access device that consists of strips of magnetic tape stored in subcells. Each data cell drive contains 10 cells, each of which contains 20 subcells of 10 strips each. Cells are suited for storing large but infrequently accessed datasets. A bin is rotated under a picking

mechanism that extracts the desired tape strip and wraps it around a drum for data access.

The type of disk drive you select depends, of course, primarily on its use for long- or short-term work, setups, etc.

Table 2 summarizes capacity, timing, and utilization characteristics of the drives.

TABLE 2

CHARACTERISTICS OF DISK DRIVES

	7330-12	3330-1&2	2314	2321
Track size (max. block size), bytes	19,069	13,030	7,294	2,000
Cylinder size				
tracks	30	19	20	20
bytes	572,070	247,570	145,880	40,000
Timings				
aver. seek, ms	25	30	75	387.5
aver. rotat. delay, ms	8.4	8.4	12.5	47.5
transfer rate, bytes/sec	1,198	806,000	312,000	55,000
utilization for recommended blksize				
card image				
2960 - cards/trk	222	148	74	NA
- % used	93	91	81	NA
load module				
- blks/trk	3	2	1	NA
- % used	97	94	84	NA
unformatted I/O				
- blks/trk	3	2	1	NA
- % used	98	96	86	NA

NA = not applicable

Drums and Disks

The IBM 2305-II fixed head disks are used to contain heavily accessed system modules that are essential to system support. These devices are not available to user data.

Also inaccessible are the IBM 2301 drums, used by the TSO system to swap user programs into and out of core when the user is not active and to contain the operating system (OS) job queue.

For further information, see IBM System/370 Component Descriptions--2820 Storage Control and 2301 Drum Storage (GA22-6895).

TAPE DRIVES

Twenty-three IBM tape drives are accessible to the batch processors, as well as one tape drive used to initialize the 360/75. These include both seven-track models with recording densities of 200, 556, or 800 bpi and nine-track models with reccrding densitites of 800, 1600, or 6250 bpi. Corresponding to these densities are the data transfer rates and capacity of the tape drives, as shown in Table 3.

TABLE 3

CHARACTERISTICS OF TAPE DRIVES

Density bytes/in.	Data Rate bytes/sec	Capacity, bytes (2400' at 10,000 bytes/blk)
7-track		
200	22,500	5,438,424
556	62,500	14,731,304
800	90,000	20,830,189
9-track		
800	90,000/160,000	21,068,702
1600	320,000	40,291,971
6250	1,250,000	125,454,545

UNIT RECORD EQUIPMENT

The Central Computing Facility supports seven IBM printers, a CalComp microfiche printer, and two IBM card reader/punches.

Printers

Seven printers provide a nominal printing capacity of 19,500 pages per hour. Six of the printers function by impact, printing at a rate of 1100 lines per minute; the seventh, the IBM 3800 Printing Subsystem, uses electrophotographic/laser technology to achieve printing speeds up to 20,000 lines per minute, while operating at a constant paper speed of 32 inches per second. The latter is further distinguished in that it is a page printer. Pages are 11 x 8 1/2 inches, somewhat smaller than the typical fanfold paper of the impact devices, with a 1/2-inch margin required on all sides.

The PN train character set for uppercase letters, plus PL/I special characters, is standard on five impact printers. In addition, uppercase and lowercase text printing, requiring the TN train character set, is supported on a sixth impact printer and on the 3800.

For further information, see IBM 1403 Printer Component Description (GA24-3073) and IBM 2821 Control Unit (GA24-3312).

Microfiche

Computer-generated microfiche output is produced by a CalComp 2100 which, while functionally equivalent to an IBM 1403, is approximately 15 times faster. One fiche, a 105x145-mm piece of film, contains 288 frames, each of which is equivalent to a page of standard printer output. A 48X reduction scale is used.

For further information, see "Producing Microfiche Output at Argonne's Central Computer Facility," Tech. Memo. 260.

Card Reader/Punches

There are two IBM 2540 card reader/punches that read at the rate of 1000 cards per minute and punch at the rate of 300 cards per minute. Two modes are available: Up to 80 columns of EBCDIC information may be punched into a card, or a card may be punched in column binary format.

For further information, see IBM 2540 Card Reader/Punch (GA21-9033) and IBM 2821 Control Unit (GA24-3312).

GRAPHICS EQUIPMENT

Several offline graphics output devices are operated by the Central Computing Facility. These include three CalComp plotters and a VERSATEC plotter, as shown in Table 4.

TABLE 4

PLOTTERS

	VERSATEC	CalComp 580	CalComp 780	CalComp 936
tape drive (unlabeled)				
tracks	9	7	7	9
density, bpi	1600	200	556	1600
paper size, in. x ft	10.56x500	12x120	12x120	36x120
plotting speed, in./sec	1.1	3.0	1.125	3.6 (pen down) 5.0 (pen up)
resolution, in.	0.005	0.01	0.0025	0.002

Also operated offline is a III FR-80 film recorder, driven by an unlabeled nine-track 800 bpi tape. It plots on 35-mm unsprocketed B&W film and 35-mm and 16-mm sprocketed B&W or color film. Color combinations are produced by selecting white, red, green, and blue filters under program control. A wide range of intensities and thicknesses is also available, with a maximum resolution of 1 part in 16384 for one direction on 35-mm film and a minimum resolution of 1 part in 4096 for one direction of 16-mm film.

Operating online from the 360/75 under TSO are several Tektronix graphics terminals. Among these are the 4010 and 4014 models, with cursor coordinate input capabilities for interactive graphics programming.

Further information on plotting devices and graphics packages is presented in Graphics 1.0 (Tech. Memo. 335).

TERMINALS SUPPORTED

Terminals that can be connected to the Argonne timesharing systems are widely dispersed around the Laboratory. Most of these computer terminals are general-purpose units that can also be connected to other off-site timesharing systems, just as most off-site computer terminals could be used on our system.

Instructions for the use of specific terminals, including instructions for dialing the computer and selecting alternative timesharing systems, vary considerably. We urge you to prepare and post specific instructions near each terminal in your area. Each issue of the HEX contains a current list of telephone numbers; the several locally-written timesharing reference manuals contain LOGON/SIGNON instructions; and the AMD User Services consultant will answer questions about the use of various terminals.

Most of the time you can use any available terminal, regardless of its type, to perform common tasks such as changing the contents of datasets, submitting batch jobs, and running programs interactively. Special terminals are required, however, to plot graphs or to produce high-quality typescripts.

The four terminal types most commonly used at Argonne are discussed below.

ASCII Printing Terminals

ASCII printing terminals are the most common type of timesharing terminals in use at Argonne. This category includes Model 33 Teletypes, which operate at 10 characters per second, and Texas Instruments Silent 700 terminals, which are typically set to operate at 30 characters per second. They communicate with the computer in the American Standard Code for Information Interchange (ASCII), which defines 128 characters; ninety-four of these may be printed as unique graphics, and thirty-four are non-printing control characters.

Some ASCII terminals send only 97 of the possible 128 ASCII codes, omitting lowercase letters and the five special characters: acute accent, opening brace, closing brace, vertical line and tilde. Moreover, there are variations in the way several other characters are printed; for example, on Model 33 Teletypes, underline appears as a left-pointing arrow and the circumflex appears as an upward-pointing arrow.

For compatibility with the IBM internal character code (EBCDIC), which differs from ASCII, several characters have been assigned special meanings: The ASCII tilde is stored in the computer as the NOT sign, and the ASCII vertical line as the logical OR symbol. Other characters transmitted by these terminals have no counterparts in EBCDIC and are variously ignored or translated into "unassigned" characters by the timesharing systems programs.

Most of these terminals use an acoustical coupler and an ordinary telephone to connect to the computer system.

ASCII CRT terminals

ASCII CRT terminals display their output on a television screen rather than printing it on paper. Examples of CRT terminals in use at the Laboratory are Infoton, Beehive, and Tektronix 4023. They are functionally equivalent to ASCII printing terminals but have the added advantage of running at higher speeds; typically terminals of this type at Argonne run at 120 characters per second, using a special coupler and telephones equiped with a white "exclusion" button.

ASCII graphics terminals

ASCII graphics terminals are represented by Tektronix Models 4006, 4010, 4012, and 4014. They have green screens on which text can be displayed, or dots and lines can be plotted. Like ASCII CRT terminals, they typically are set to run at 120 characters per second, using a special coupler and telephone.

If a Tektronix graphics terminal is left idle for more than 90 seconds, the scope will go into a "hold" status (the screen brightness dims); if the shift key is depressed or any character is entered, the scope will return to its normal display status. Keeping the same image on the screen for 15 minutes in display status, or 1 hour in hold status, could damage the screen. Therefore, you should always clear

the screen by hitting the RESET-PAGE button between sessions.

IBM typewriter terminals

IBM typewriter terminals, based on Selectric typewriters, have changeable type-spheres, and print at 15 characters per second. There are IBM 2741 terminals, IBM CMCSTs (Communicating Magnetic Card Selectric Typewriters), and various compatible units marketed by other companies. There are two different keyboard arrangements: (1) Correspondence, which is identical to that of a standard electric typewriter, and (2) PTTC/EBCDIC, which is designed for ease of use by programmers. Not all IBM-compatible typewriter terminals will work on the Argonne timesharing systems, because several special options are required.

OPERATING SYSTEMS

Several different control programs are used by Argonne to run jobs through the computer system. Each of these control programs has different tasks to perform and interacts when necessary with one or more of the other control programs.

OS/MVT

OS/MVT (Operating System Multiprogramming with a Variable Number of Tasks) is the primary control program running the four central processors. OS is responsible for allocating the resources (memory, CPU, I/O devices, channels, direct-access storage) required by a particular job step, arbitrating when there is contention for one of these resources, and ensuring that a job step does not use more resources than allowed.

ASP

Used alone, OS would be insufficient to meet the needs of Argonne's workload (typically a large number of short-term jobs). Indeed, in many cases, the time involved in setting up the devices would far exceed the time required for job execution.

Consequently, OS is used in conjunction with ASP, the Asymmetric Multiprocessing System control program. ASP manages

job input and output, setup scheduling, and communications between the computers.

VM

Another control program developed by IBM, VM (Virtual Machine), is intended to help minimize the stand-alone systems time needed to test modifications to the other control programs. VM is installed on one of the 3033's.

Running as part of VM/370 is IBM's Conversational Monitor System (CMS), an outstanding modern interactive system. CMS allows the development of new interactive work and will also permit use of the 3033 for OS/MVT batch and for extensive system testing leading to the installation of new systems, VM Release 6, and MVS.

For more information about CMS see CMS at ANL: A Primer (Tech. Memo. 339).

WYLBUR and TSO

Neither OS alone nor OS in conjunction with ASP provides for interactive computing. To satisfy this need, two timesharing control programs are used: WYLBUR, developed at SLAC, and TSO, the Timesharing Option of OS/MVT.

WYLBUR provides easily used, powerful, interactive text editing, and the facilities to create, submit, and control jobs to be run on one of the batch systems. Since WYLBUR's capabilities include those required by the majority of the timesharing sessions and since WYLBUR uses relatively few computer resources, it is the preferred timesharing system at Argonne.

TSO is used to provide other interactive computing capabilities. A TSO user can compile and execute a source program, produce graphical output on a CRT terminal, and invoke a number of utilities to manage data on direct-access storage devices. TSO also includes text-editing facilities and the ability to create, submit, and control jobs to be run on one of the batch systems.

Both TSO and WYLBUR run as jobs themselves under the control of OS. Both also interact with the batch systems under the control of ASP.

MORE INFORMATION ON OPERATING SYSTEMS

For more information about the Operating Systems at ANL, consult:

CMS at ANL: A Primer (Tech. Memo. 339)

IBM Virtual Machine Facility/370: Introduction (GC20-1800)

Batch Processing at ANL (Tech. Memo. 337)

Hylbur Reference Manual (Tech. Memo. 294)

Applied Mathematics Division's Implementation of the Time-Sharing Option (Tech. Memo. 262)

Chapter 3

SERVICES OFFERED BY THE CENTRAL COMPUTING FACILITY

Various computing services offered by the Applied Mathematics Division (AMD) are listed in Table 6dir. below, together with information on the person or group to contact for assistance.

TABLE 5

DIRECTORY

TO OBTAIN	CONTACT	ROOM	PHONE
Setup Disk Pack	Systems Specialist: Joe Marentic	A134	2-5450
Account number, CUA Card	Data Management and Accounting Services: Ethel Fearnow	A143	2-5425
Tape	Tape Librarian: Janet Stonebrook	A143	2-7681
Keypunch Information, Data Preparation	Data Services, 221: Joanne Nachtman	A120	2-5426
	Data Services, ADP, 801: Eileen Graff	C123	2-6924
Systems and Applications Office		A259	2-7182
Systems Status Information			2-5466
Courses, Information	User Services	A142A	2-5415
General Information	Program Consultants	A142B	2-5405
Documentation, Manuals, Program Library Material	Documentation Center	A142B	2-5406
Equipment Malfunction Reports			2-7273
RADS Hardware Maintenance	RADS Maintenance (RCA Service Corp.)	D156	2-7273 2-7273
Scheduling Area		A134	2-5421
Operations		A134	2-5421
IBM Maintenance for 2741 Terminals		C112	2-5419

DATA PREPARATION

In addition to the services listed, AMD provides data preparation facilities, consultation, code tuning, and code export services. Each of these is discussed below in greater detail. The following services are available in the Data Preparation Room, A120, 2-5426:

- Keypunching (all types).
- Preparation of drum cards.
- Instruction from Data Preparation operators as to the use of machines in the Data Preparation Area.
- Wiring of boards for the reproducer (IBM 519) or for the interpreter (IBM 557).

As time permits, Data Preparation operators will also perform the following functions:

- Card reproduction.
- Card interpretation.

The following facilities are available for you to do your own:

- Keypunching. (IBM 026 and IBM 029)
- Card reproducing or gang-punching (IBM 519).
- Card interpreting (IBM 557).
- Card reproducing and interpreting on Burroughs PC8.

No special procedures are required to use the available machines in the Data Preparation Area. If you are unfamiliar with their operation, please ask the Data Preparation personnel for help.

CONSULTATION

The Consultant's Office (A142, 2-5405), provides help to persons using the computer system. In addition to giving on-the-spot assistance, the Consultant's Office receives reports of system and hardware malfunction and is the collection point for requests for system design modifications. The Consultant's Office is open from 8:30-11:30 and 13:00-17:00, Mondays through Fridays. AMD will help you improve the performance of programs that consume substantial

amounts of computer time. These programs will be run under a software monitor (PROGLOOK) to isolate the most active blocks of code. The results of these runs will be made available to you, and the consultants will offer suggestions aimed at improving performance. Coding changes suggested will generally be in the program source language and will attempt to take advantage of hardware performance features available on the processors.

If you feel your codes are candidates for tuning, contact User Services at 2-5405. AMD goals and resources restrict consideration to only those codes with major system impact.

CODE EXPORT SERVICE

AMD provides the ability to use the computer facilities of five other organizations.

- Lawrence Berkeley Laboratory
  2-CDC 6600's, 1-CDC 7600
- Brookhaven National Laboratory
  2-CDC 6600's, 1-CDC 7600
  (See TM-279)
- Stanford Linear Accelerator Center
  2-IBM 370/168's, 1-IBM 360/91
- University of Chicago
  1-IBM 370/168
  (See Tech. Memo. 280)
- Control Data Corp. Cybernet
  1-CDC 6600, 1-CDC 7600

Data communications and remote job entry equipment have been installed in AMD to allow use of these remote facilities. Procedures for remote submission of jobs have been established. You may bring job decks (and tapes with card images) to the code export counter, located in the AMD Scheduling Area. Output will be delivered to your bin.

User Services provides colsulation and information about all off-site facilities. Guides for the most popular sites are also available from User Services.

The costs of computing at other sites are determined by formulae that differ from AMD cost computing methods. Different types of programs will either benefit or be hindered by different computing hardware. AMD's policy is to pass the computing costs on to the users without adjustment.

Chapter 4

ACCOUNTING AND RATIONING

Both the batch and timesharing computing systems are rationed. Laboratory management allocates to each cost center a fixed amount of dollars that can be spent on rationed resources during each week. Each cost center appoints a ration manager to oversee the proper use of its allocation. You should may direct any questions about rationing to your cost center ration manager.

Batch rationing is based on limits for the total number of dollars that can be spent on CPU, WAIT, and CORE during each week. A week begins on Monday morning (0700 hours). Computer center personnel monitor computer usage daily. When a cost center exceeds its allocation, a job priority barrier is set to allow the cost center to submit only stand-by priority jobs. Jobs of a higher priority are automatically reclassified to stand-by status. You cannot raise the pricrity of a stand-by job in this instance. Stand-by jobs are not charged any ration, and they are released to be run only when the machine is otherwise idle. Batch charges in excess of the allocations have no effect on the following week's allocation.

TSO rationing is based on resource units (RU's). A measure of CPU time, terminal connect time, and disk I/O operations is used to calculate the RU's for a session, specifically:

RU's = mx(cpu-rate x CPU-seconds + connect-rate x connect-hours + I/O-rate x number of EXCP's)

where m is a multiplier that is currently equal to 1 for a prime-time session (see below), .25 during the evening hours, and 0 during late hours and holidays.

The hours of the week are divided into three periods for TSO rations:

- PRIME: Weekdays from 07:00 until 17:00
- EVENING: Weekdays from 17:00 until 22:00
- LATE: Weekdays from 22:00 until 07:00 and all weekend.

Logon requirements will vary for these periods:

- PRIME: During this period, you are not allowed to logon unless

  charges < ration + 13 units

  The 13 units are known as the 'cushion' and allow you some flexibility.

- EVENING: During this period, you can logon unconditionally and spend your rations at the rate of 25% of PRIME.

- LATE: During this period, you can logon unconditionally and are not charged for ration usage.

TSO charges are defined as follows:

charges = carryover from prior weeks* + usage - refunds

*The carryover from prior weeks (negative carryover) can be accumulated as a result of an extremely costly session or overusage during the prime or evening periods. Negative carryovers are zero if your cost center in aggregate did not overspend. When center overspending occurs, your negative carryover is proportional to your overspending.

You must logoff and logon at period boundaries to gain the benefit of the time period changes in charges. Currently, ration dollars are equivalent to billed dollars during PRIME and EVENING periods. Although no ration charges are assessed for LATE period usage, cost codes are billed at current off-prime rates.

Chapter 5

LANGUAGES AND COMPILERS

The major programming languages available at ANL include FORTRAN, PL/I, Speakeasy, Assembler Language, COBOL, ALGOL, and RPG. All except Speakeasy are maintained and supported by the Applied Mathematics Division; Speakeasy is a high-level language maintained by the Speakeasy Computing Corporation. Since COBOL, ALGOL, and RPG are seldom used in the ANL environment, discussion of these languages will be minimal.

Many levels of programming manuals for all of these languages are available in the Documentation Center.

CHOICE OF A LANGUAGE

Once you have decided to solve a problem using the computer, you are faced with the choice of a programming language. The choice is simple if you have had prior experience with a major language and are satisfied with it: The reasons for learning a new language, rather than building on the prior investment of time, are few. Similarly, if you know several languages, you will be able to evaluate the suitability of each as the potential medium for the expression of your new problem. However, the new programmer has to make a choice that will significantly affect his use of computer resources. This section is intended to guide new users in making a rational choice.

The individual factors affecting choice of a language are treated separately.

> NOTICE: Languages and compilers available on CMS are not discussed here. See CMS at ANL: A Primer (Tech. Memo. 339), and the June, 1979 issue of the HEX for more information.

Nature of Application

The nature of the application will sometimes completely override other considerations, because some languages support certain types of applications much more naturally than other languages do.

FORTRAN is generally suited to all types of numerical computation and has the advantage that almost the whole language is devoted to that goal. FORTRAN is very widely used, and compilers for it are available on most computers. FORTRAN programs tend to be more transportable from one machine type to another than programs in other high-level languages. However, even strict adherence to the 1966 ANSI FORTRAN Standard syntax seldom results in a program that can be run on different machines without modification.

PL/I is intended to be an all-purpose language serving business, scientific, and systems applications. Business applications make use of its string-manipulation, editing, and file-handling capabilities. Its mathematical operations and library support scientific applications. The list-processing, multitasking, and teleprocessing capabilities provide support for systems applications. Some of these features are applicable to an intermediate class of problems generally called "non-numerical". For instance, PL/I's broad string-manipulation capability makes it eminently suited for specialized editors and other processors of textual data; and its list-processing capability makes it ideal for the writing of compilers and other applications, such as modeling, where information is represented by the dynamic relationships among items of data. PL/I is a much more extensive language than FORTRAN; to code a purely numerical problem in PL/I, you should know what parts of that language to avoid.

Speakeasy has basically the same range of applicability as FORTRAN. However, common scientific operations, such as matrix arithmetic, statistical calculations, and tabulation of functions are expressed concisely in the high-level notation of Speakeasy, whereas in FORTRAN they would have to be programmed out of simpler constructs, including the application-independent loop.

Assembler Language is not application-oriented; any problem that can be solved on the hardware can be coded in Assembler Language. It is generally chosen only for small "modules" of a large system, where a great degree of efficiency and program compactness is necessary, or to provide a service that the higher level language cannot perform.

COBOL is oriented toward business programming applications, in which file handling and output reports play a more signi-

ficant role than numerical calculations. ALGOL is a high-level language suitable for expressing a large class of numeric processes concisely. RPG is a problem-oriented language that provides an efficient, easy-to-use technique for manipulating files and writing reports.

Transportability Requirements

If you are going to write a program here that may be used at another installation, such as another DOE laboratory, you should choose a widely supported language. To be on the safe side, you should also probably restrict yourself to standardized features of that language. FORTRAN, PL/I, and COBOL have been standardized by the American National Standards Institute (ANSI). The FORTRAN standard, dating from 1966, is recognized as the "transport" language. (A major revision of the ANSI FORTRAN Standard was completed in 1977 but has not been widely implemented.) PL/I was standardized only as recently as 1976 and thus has not yet been as widely implemented as FORTRAN.

Learning Time and Difficulty

The investment in time required to learn a new language can be a significant factor in the choice of a language. Scientists may choose Speakeasy for several reasons: (1) few new concepts must be learned; (2) an interactive version with a tutorial facility and an extensive HELP facility is available on CMS or TSO; and (3) several documents from the introductory to the advanced level are available. FORTRAN requires somewhat more time to learn, but, for large-scale numerical calculations, its use is fairly straightforward; it is certainly easier to learn than PL/I. PL/I is a large and very rich language, and the IBM manuals are not too well organized for identifying and learning a subset of it. In addition, there are hidden costs in learning to use PL/I properly when one's experience with it is not extensive: The generality of the language permits many incorrect programs to run, producing erroneous results the cause of which may be hard to find. COBOL's English-like syntax makes it relatively easy to learn and use. The expenditure of time required to learn Assembler Language is much greater than for the high-level languages.

Availability of Libraries

Unnecessary effort can often be eliminated by using programs or routines from libraries. Libraries may be standard (manufacturer-supplied) or local (user-developed), run-time (code inserted when program is executed) or compile-time (code inserted when program is compiled). Assembler Language has an extensive standard compile-time library of system macros. FORTRAN, Speakeasy, and PL/I have extensive standard run-time libraries of mathematical and miscellaneous functions. Speakeasy has a small standard compile-time library. If you use PL/I and Assembler Language, you can develop your own local compile-time libraries. If you use Speakeasy, you can develop your own local run-time libraries of functions; the conventions surrounding their use are considerably different from those for the other languages. You can also create and use libraries of named "objects", such as programs or data values, resulting from Speakeasy runs.

Interlanguage Communication

Interlanguage communication means that a compiler provides the necessary code to allow subprograms or library routines written and compiled in different languages to run together as a single program. Interlanguage communication may be advantageous, for example, to the PL/I user who wants to utilize a package such as EISPACK, which is written in FORTRAN.

Interlanguage communication between PL/I and either FORTRAN, COBOL, or Assembler Language is handled by the PL/I Program Product compilers, regardless of whether PL/I is called or does the calling. See Chapter 19 of the PL/I Checkout and Optimizing Compilers: Language Reference Manual (SC33-0009) for more details.

Interlanguage communication is also possible with Speakeasy. The run-time environment of Speakeasy is essentially that of FORTRAN; thus, locally written run-time routines, for use by Speakeasy programs, are usually written in FORTRAN.

Efficiency and Optimization

Efficiency and optimization are relatively unimportant in one-shot applications, or in any simple application that has a very short running time. Longer running programs, particularly those that are CPU-bound, can benefit significantly from optimization. FORTRAN H, FORTRAN H Extended, and the PL/I Optimizer all have optional sophisticated optimization

capabilities. WATFIV is slow because of extensive run-time error checking code inserted. Speakeasy and the PL/I Checker are interpretive systems and are thus relatively expensive to use for large or repeated calculations. PL/I F has only rudimentary optimization capabilities. The most efficient code can be written in Assembler Language, but only at the cost of greater development effort.

COMPILERS

In the following section, we discuss the features of the various compilers that can be used for FORTRAN, PL/I, Speakeasy, Assembler, COBOL, and RPG.

FORTRAN

The FORTRAN (FORmula TRANslator) language is especially useful in writing programs for applications that involve mathematical computations and other manipulations of numerical data. ANL supports four FORTRAN compilers (G, G1, H, H Extended), the WATFIV compiler, and the MORTRAN preprocessor. We recommend using WATFIV for early program development and debugging, the G1 compiler for initial program checkout or short-running programs, and the H Extended compiler for long-running codes.

WATFIV

WATFIV is a one-pass FORTRAN compiler that achieves extremely fast compilation times and performs run-time checking for such errors as using uninitialized variables, array subscripts out of range, or mismatched arguments in subprogram invocation. It is a very powerful debugging tool and should be used only for that purpose. The overhead involved in extensive error checking causes execution times to be as much as an order of magnitude larger with WATFIV than with FORTRAN H Extended with OPT=2.

Many of the WATFIV extensions to FORTRAN have been removed from ANL's implementation of the interpreter, to prevent anyone from writing programs that are dependent on any special feature of WATFIV. The current version of WATFIV also has some restrictions: No searches of load module libraries can be performed, and the inclusion of object modules from other compilers requires non-trivial effort.

WATFIV is available to TSO through the WATFIV command. Its interactive debugging capability greatly enhances the tools available for program development and checkout. With this facility, you can trace portions of your program, halt execution at various points, display or modify program variables, alter the logic flow of a program, and correct certain execution-time errors interactively. The description of ANL's implementation of WATFIV is found in the WATFIV Reference Manual available in the Documentation Center.

FORTRAN G1

The FORTRAN IV G1 compiler is suitable for relatively short checkout runs. Its MAP option associates a hexadecimal address with each executable statement (not just those with FORTRAN statement numbers, as with the H and H Extended compilers); this is a very useful debugging aid. G1 also supports list-directed input/output, which frees you from FORMAT statement restrictions, and a debug facility, which enables you to trace program flow and to check for errors.

The G1 compiler is the one most easily accessed in TSO. Two methods of invoking the compiler are available: 1) the RUN command, which, when issued in edit mode, compiles and executes a FORTRAN program, and 2) the FORT command, which, when used in command mode, compiles a program followed by LINK and CALL commands to linkedit and execute the program or the LOADGO command to load and execute the program. Further information can be found in the FORTRAN IV G1 TSO Terminal Users Supplement, in the Applied Mathematics Division's Implementation of the Time-Sharing Option (Tech. Memo. 262), and in the TSO Command Language Reference (GC28-6732).

A complete explanation of the options available for the G1 compiler is found in the FORTRAN IV G1 Programmer's Guide (SC28-6853). The detailed description of the G1 compiler diagnostic messages is found in the FORTRAN IV G1 Processor and TSO FORTRAN Prompter Installation Reference Material (SC28-6856).

FORTRAN H Extended

The FORTRAN IV H Extended compiler offers the following advantages:

- Optimization: Compiler options OPT=1 or OPT=2 typically result in much more efficient object code than is produced by the default OPT=0 or by the G1 com-

piler, at the cost of moderately longer compile times. OPT=2 often yields 30% or greater reduction in execution time of the resulting object module.

- Extended Precision: REAL*16 and COMPLEX*32 variables are accommodated, and a library of extended precision versions of the common elementary functions is available.

- Automatic Function Selection: The GENERIC statement causes the appropriate function to be selected, based on the length and type of arguments specified.

- Automatic Precision Increase: This option (together with the GENERIC statement) facilitates conversion of programs from single to double or double to extended precision.

- EXTERNAL Statement Extension: An extension to the EXTERNAL statement provides a means of declaring a user-supplied subprogram to be executed, instead of a FORTRAN library or built-in function.

- Asynchronous Input/Output: This facility enables the transmission of unformatted sequential data between direct-access or sequential storage devices and arrays in parallel with other computations.

- List-Directed Input/Output: This feature allows input/output without specific FORMAT statements (also available in FORTRAN G1).

You can also access the FORTRAN H Extended compiler from TSO. Because it needs considerably more CPU time and memory than the G1 compiler, however, its use on TSO should be limited to interactive applications requiring its special features. The compiler options available for FORTRAN IV H Extended are discussed fully in the FORTRAN IV H Extended Programmer's Guide (SC28-6852).

FORTRAN H

The FORTRAN IV H compiler, although quite reliable, is no longer supported by IBM and can generally be replaced by the H Extended compiler. The major reason for using this compiler is to take advantage of its excellent optional optimizing capabilities to increase execution speed and to reduce the size of the object module. Two levels of optimization are available. With OPT=1, the compiler treats each source module as a single program loop and optimizes the loop with regard to register allocation and branching. With OPT=2, the

compiler treats each source module as a collection of program loops and optimizes each loop with regard to register allocation, branching, common expression elimination, and replacement of redundant computations.

If you are interested in gaining access to the FORTRAN IV H compiler in TSO, you should read the Applied Mathematics Division's Implementation of the Time-Sharing Option (Tech. Memo. 262) and in the TSO Command Language Reference (GC28-6732).

Options available with the H compiler are explained fully in the FORTRAN IV G&H Programmer's Guide (GC28-6817), which also contains other information about the compiler-generated diagnostic messages.

MORTRAN

MORTRAN is a structured language translated by the preprocessor into FORTRAN statements, which are then processed in the usual way. A set of macros is provided to help you structure your program, and user-supplied macros may also be specified to define other problem-oriented applications. Some of the features of MORTRAN include free form coding, alphanumeric labels of arbitrary length, comments inserted anywhere in the text, nested block structure, nested conditional statements, loops that test for termination at the beginning or end or both or neither, multiple assignment statements, abbreviations for simple I/O statements, and interspersion of FORTRAN text and MORTRAN text.

Further information on MORTRAN is available in the Documentation Center.

FORTRAN G

The FORTRAN IV G compiler is not recommended; it has been superseded by the G1 and H Extended compilers. Should you wish to use this compiler, however, you should get a copy of FORTRAN IV G&H Programmer's Guide (GC28-6817).

PL/I

The designers of PL/I have produced a language that will accommodate a variety of users whose needs are sufficiently diverse to have formerly required the implementation of several separate languages. Its significant features include a

very large collection of data types, expressions, four types of storage allocation, extensive I/O capabilities, multitasking, and list processing.

The following features of PL/I are not supported by the current configuration of our hardware and operating systems: transient files (telecommunications), VSAM datasets, and checkpoint/restart.

The PL/I language is described in PL/I Checkout and Optimizing Compilers: Language Reference Manual (GC33-0009).

Other useful references are A Guide to PL/I for FORTRAN Programmers (SC20-1637), A Guide to PL/I for Commercial Programmers (SC20-1651), Class Notes for a PL/I Course, (ANL 76-70), and PL/I F Language Reference (GC28-8201)

Three PL/I compilers are available to ANL users: Checkout, Cptimizing, and F.

Checkout Compiler

The PL/I Checkout Compiler (also called Checker) is intended for program development. It is organized as a translator, which produces intermediate text, and an interpreter, which interprets the intermediate text. The translation phase is very fast, because it does not need to provide optimized executable code, and because some of the global checking is deferred until interpretation. The interpreter is much slower but is capable of detecting errors and illegal language use and reporting these in source-language terms. Many of the errors that it diagnoses are not detectable by more conventional compilers. The Checkout Compiler offers complete facilities for interactive debugging in CMS and TSO.

Further information on using the Checker in TSO, including essential information on source margins, is found in the Class Notes for a PL/I Course (ANL 76-70) and in the PL/I Checkout Compiler: TSO User's Guide (SC33-0033).

A complete explanation of the options available with the Checkout Compiler is found in the PL/I Checkout Compiler: Programmer's Guide (SC33-0007). The diagnostic messages issued by the compiler are explained in the PL/I Checkout Compiler: Messages (SC33-0034).

Optimizing Compiler

The Optimizing Compiler produces efficient, optimized machine code for PL/I programs that have been debugged using the Checkout Compiler.

A very helpful discussion of some special features and considerations for using the Optimizing Compiler in batch processing is found in the last half of Chapter 13 of Class Notes for a PL/I Course (ANL 76-70). The PL/I Optimizing Compiler: TSO User's Guide (SC33-0029) contains complete instructions for using the Optimizer in the TSO environment.

All of the options available with the Optimizer are described in the PL/I Optimizing Compiler: Programmer's Guide (SC33-0006), which also contains a great deal of other information about the Optimizer. Diagnostic messages issued by the Optimizer are explained in PL/I Optimizing Compiler: Messages (SC33-0027).

F Compiler

The PL/I F Compiler was the original IBM compiler for full PL/I. The enhancement of this compiler ceased long ago when IBM chose to emphasize the Checkout and Optimizing Compilers. There is little to recommend the use of PL/I F for new development; this compiler, in fact, is no longer supported in the newer (virtual storage) operating systems. PL/I F can be invoked in batch processing but there are no public command procedures for invoking it in TSO.

A complete explanation of the compile-time options, as well as other information about the compiler and a description of diagnostic messages, is found in the PL/I F Programmer's Guide (GC28-6594).

Speakeasy

Speakeasy is a computer language maintained by the Speakeasy Computing Corporation. Ease of use, natural notation, and built-in capabilities for growth are important features of Speakeasy. The language is based on the concepts of arrays and matrices that are processed as entities, thus eliminating the need for many of the loops necessary in other programming languages. It has a large vocabulary (over 500 words) of functions and commands in the area of array manipulation, matrix algebra, including eigenanalysis, special mathematical functions, numerical integration and differentiation, statistics, graphics, and character processing. It

is also used at over eighty other installations around the world.

Speakeasy is available for batch processing, CMS and TSO. A Guide to TSO Speakeasy (ANL 75-44) is available in the Documentation Center. A complete description of Speakeasy capabilities is available in The Speakeasy-3 Reference Manual (ANL 8000 Rev. 1).

Assembler Language

Computer programs may be expressed in machine language, i.e., language directly interpreted by the computer, or in a symbolic language that is more meaningful to the user. Of the various symbolic programming languages, assembler languages are closest to machine language in form and content. They contain mnemonic machine-instruction operation codes, assembler-instruction operation codes (used to specify auxiliary functions to be performed by the assembler), and macro instructions (which stand for a sequence of machine and/or assembler instructions).

Four levels of assemblers are available to ANL users: F, G, VS, and H. Each of these is described below. Assembler language for the F and G levels is described in OS Assembler Language (GC28-6514).

Special features of the VS Assembler are presented in OS/VS-DOS/VS-VM/370 Assembler Language (GC33-4010). The language for the H level is discussed in OS Assembler H Language (GC26-3771).

Assembler F

The F Level Assembler is the oldest and least powerful one available at ANL. Available in both batch and TSO, it is explained in the OS Assembler F Programmer's Guide (GC26-3756), which also describes the diagnostic messages issued by the assembler and contains other useful information. Assembler F is standard with OS; code assembled by it will assemble at any IBM site.

Assembler G

The G Level Assembler was acquired from the University of Waterloo and is a modification of IBM's Level F Assembler. It considerably reduces the time required for assembly and

allows larger programs to be assembled. In addition, it provides a batch-processor for small assemblies and a more concise cross reference dictionary, allows suppression of the external symbol dictionary and the relocation dictionary, allows the concatenation of unlike datasets on unlike devices, and provides several language extensions (all under the control of the EXTEN option). The update facility of Assembler G allows the inclusion of IEBUPDTE-type additions and deletions to the master source dataset (which remains unchanged).

Assembler G is available in both batch and TSO. Further information is given in the Assembler G User's Guide.

Assembler H

The IBM program product, Assembler H, contains significant extentions in the functions and features of the assembler language and the macro language. Some of the features include increased maximum length for symbols, named common support, extended mnemonics, cross referenced literals, short XREF option, nested copy, macro redefinition support, new system variables ($SYSTIME, $SYSDATE, $SYSPARM), batch assembly, and error message printout at the point of error. Assembler H is recommended for large or complex assemblies or for applications that make extensive use of the macro facility. OS Assembler H Language (GC26-3771) describes the language features of this assembler, which is available only in batch.

> NOTICE: Code using extensions available in G and H may not assemble at some IBM sites that have only the F level assembler.

Further information is given in the OS Assembler H Programmer's Guide (SC26-3759) and in the OS Assembler H Messages (SC26-3770).

COBOL

COBOL (COmmon Business Oriented Language) was developed primarily for the programming of commercial problems. It uses a syntax closely resembling English and avoids the use of symbols as much as possible. It is especially efficient in processing business problems that involve relatively little algebraic or logical processing, but that manipulate large files of similar records in a relatively simple way. COBOL Version 4, which is installed at ANL, features optimized object code and advanced symbolic debugging capabilities.

The language is described in OS Full American National Standard COBOL Reference Guide (GC28-6396).

The COBOL compiler is available in both batch and TSO, but there are no commands to facilitate TSO use. Compiler options available with COBOL Version 4 are explained in the OS Full American National Standard COBOL Compiler and Library, Version 4, Programmer's Guide (SC28-6456), which also describes how you can obtain a listing of the compiler-generated diagnostic messages.

RPG

RPG (Report Program Generator) is a problem-oriented language designed to provide an efficient, easy technique for generating programs to manipulate data on files, perform calculations, and write reports. RPG uses a set of specification forms on which you make entries describing files, calculations to be made, and reports to be generated. This compiler is available only in batch.

DEBUGGING AIDS

Several facilities are available, both from the operating system and from the individual compilers, to help you check out and debug your programs. Additionally, compilers that are principally designed as debugging aids have been installed for FORTRAN and PL/I. This section will explore some of these features available at ANL.

General Hints

The MSGLEVEL parameter of the JOB statement specifies which job control statements and allocation/termination messages are to be written in the SYSMSG portion of the output. The default, MSGLEVEL=(1,1), causes all such messages to be written. These messages are useful in isolating JCL errors, assessing step completion codes, and determining dataset location and dispostion. Step summaries also show the number of blocks of data transmitted for each device.

The SYSMSG output should be scanned, step by step, for the step condition code, which is located between the device allocation and the dataset disposition messages. A message other than IEF142I - STEP WAS EXECUTED - COND CODE 0000 indicates possible trouble within that step. Abnormal ter-

mination (ABEND) messages appear in this portion of SYSMSG output if execution was halted because of severe problems. In this event, system and user completion codes are given. The explanation of system completion codes is found in Part I of CS Messages and Codes (GC28-6631). Part II of that manual describes the other system messages. Diagnostic messages issued by compilers are generally explained in the programmer's guide for the compiler or in a separate message manual (previous sections of this chapter indicate the location of compiler messages for each of the ANL supported compilers). Messages from the linkage editor and loader are found in OS Linkage Editor and Loader (GC28-6538).

A system dump may be obtained in the event of an abnormal termination by the inclusion of a special DD statement in the JCL for a job step. A dump of the processing program storage area, which is generally of greatest interest, can be obtained with the SYSUDUMP ddname. Dumps of this type are generally most useful in the execution step of a job (compiler ABENDS are usually self-explanatory). A typical use of the SYSUDUMP statement in a FORTRAN compile, load, and execute job would be:

```
//  EXEC  FTXCLG
//SYSIN  DD  *
    (source program)
//GO.SYSIN  DD  *
    (data)
//GO.SYSUDUMP  DD  SYSOUT=A
```

If the program terminates abnormally during execution (GO step), a dump will be written to the printer. Deciphering a dump, unfortunately, is not as easy as obtaining one. A complete explanation of the contents of the dump is contained in OS Programmer's Guide to Debugging (GC28-6670) (the applicable section is ABEND/SNAP Dump (MVT)). A generally more useful reference is Debugging System 360/370 Programs Using OS and VS Storage Dumps, D. H. Eindfleisch. This book is in the ANL library. User Services Consultants can help interpret dumps. The DDname SYSABEND can be used to define a dataset where a dump of the system nucleus, the processing program storage area, and possibly a trace table can be written. SYSABEND dumps are seldom useful for higher level language programs.

It is sometimes useful, particularly when one of the optimizing compilers is being used, to obtain a pseudo-assembler language listing of the compiled program. Most compilers provide a LIST option for this purpose. This listing can be used to pinpoint more precisely an offending statement or to determine the effects of optimization on compiled code (occasionally code removed from loops causes disaster). Other compiler options that generate various types of state-

ment label maps and cross-reference tables are also useful debugging aids (see the list of options for the compiler in question).

The program consultants in AMD's User Services Group are available in Bldg. 221, A142B, to help with stubborn debugging problems. They may also be able to make suggestions for using the facilities more efficiently.

For FORTRAN Users

An outstanding debugging tool available to FORTRAN users is the WATFIV compiler. It has extensive diagnostic facilities during both compilation and execution (when it checks for uninitialized variables and subscripts out of range). If a program meets the requirements of WATFIV, a run in its environment is recommended for initial program checkout (it should not be used for production runs because of its increased execution time).

Unfortunately, not all programs can be tested under WATFIV. The compiler is not currently able to search load module libraries (e.g., SYS1.DISLIB, SYS1.AMDLIB, or private libraries), nor is it able to accept object modules from other compilers without significant effort. These restrictions can be circumvented in several ways. Routines from private libraries for which FORTRAN source code is available can be included in that form along with the user program. Source code for AMDLIB routines is available in the Documentation Center. DISLIB (DISSPLA) plotting routines are not available in source form, but it is often possible to dummy these routines for testing purposes with the addition of subroutines of the form:

```
SUBROUTINE CURVE(X,Y,I,J)
RETURN
END
```

Once the non-graphic function of the program is working properly, the dummy plot routines can be removed, and the program can be further tested using another of the FORTRAN compilers.

The FORTRAN IV G1 Compiler offers a debug facility that provides for tracing the flow within a program, tracing the flow between programs, displaying the values of variables and arrays, and checking the validity of subscripts. The facility is described in Appendix E of the Language Reference (GC28-6515).

A formatted dump of portions of storage can be obtained using the DUMP or PDUMP service subroutines available for all of the FORTRAN production compilers. The DUMP subroutine terminates execution following the dump, while PDUMP allows execution to continue. These subroutines are explained in Appendix C of the Language Reference (GC28-6515).

Execution Error Messages in FORTRAN

When an error occurs during execution of a FORTRAN program, a FORTRAN system error recovery routine automatically receives control. This routine interprets the error, prints a message, and, if possible, applies a standard correction and resumes execution. For a discussion of the correctable errors and the standard corrections, see the programmer's guide for any of the FORTRAN production compilers under "Extended Error Handling Facility".

The description of the informational messages is incomplete, however, due to the fact that the 370/195 can generate imprecise interrupts. When such interrupts occur, the Program Status Word (PSW) is displayed in the hexadecimal form:

PSW JJJJIIIKJJAAAAAA

where K = 0 indicates an imprecise interrupt

and J = junk (irrelevant)

III = indicators of the error type

AAAAAA = address when the error was detected

The values of III (bits 16 through 27) must be inspected. Each bit corresponds to, at most, one interrupt condition. The occurrence of multiple interrupts is detected by noting multiple bits in this field being turned on (this is a relatively infrequent occurrence). Table 6 indicates the possible imprecise interrupts (many of which would not occur in a FORTRAN program).

In FORTRAN "240" messages (e.g., IHC240I, IHO240I, etc.), the value of III is always 000. However, the text of the message will contain the characters:

USER 1024 - indicating an addressing error (usually caused by a "wild" array subscript), or

TABLE 6

IMPRECISE INTERRUPT INTERPRETATION

PSW Bit	Value of III	Exception
16	800	Protection
17	400	Addressing (outside available storage)
18	200	Specification (boundary violation)
19	100	Data (conversion violation)
20	080	Fixed Point Overflow (result >(2 to the 31)-1 or <-2 to the 31
21	040	Fixed Point Divider (by 0)
22	020	Exponent Overflow (result >10 to the 75)
23	010	Exponent Underflow (result <10 to the -79 and ¬=0)
24	008	Significance
25	004	Floating Point Divide (by 0)
26	002	Decimal Overflow (result exceeds field size)
27	001	Decimal Divide (quotient exceeds field size)

USER 2048 - indicating an attempt to store into a memory location outside the region of the program (usually caused by a "wild" array subscript on the left side of an assignment statement).

The address, AAAAAA, in the PSW in any of the imprecise interrupt cases can be related to the FORTRAN source using the traceback information and the FORTRAN source listing with the MAP option as illustrated in the following example:

```
1.  IHO240I STAE ... USER 1024 ... PSW FF85000D021BA1EA
2.  TRACEBACK ROUTINE ISN     REG 14     REG 15 ...
3.            BAD     20      1A90C      1BA002
4.            MAIN            1612A      1A8010
```

The USER 1024 in line 1 means an addressing error by an instruction around 1BA1EA. Line 3 indicates the failing instruction was in subroutine BAD which was called from the MAIN program at ISN (Internal Statement Number) 20. Furthermore, REG 15 will have the address of the entry point of BAD. Thus the failing instruction was around location 1E8 (1BA1EA-1BA002) into BAD. Suppose BAD had the following structure:

```
      SUBROUTINE BAD (X)
      .
      .
      .
10    J=I
      .
      X=Y
      Z=SIN(X)
      B=A(J)
      .

      .
20    C=SQRT(X)
      .
      .
      RETURN
```

The MAP printed by the compiler would show:

LABEL	ADDR	LABEL	ADDR
10	190	20	2AC

The failing instruction is probably between statements 10 and 20. Since it is an addressing error, a subscript is likely to be out of range, and statements between 10 and 20 with subscripts, such as B=A(J), should be suspect.

Note that the MAP option of the FORTRAN IV G1 compiler produces a map of addresses for <u>all</u> executable statements, not just those having statement numbers as in the example above (which uses the H Extended Compiler).

Since several types of machine instructions can be executed simultaneously on the 370/195, it is possible that the address derived from the PSW will point to an instruction that could not have caused an interruption. In this case, instructions surrounding the address should be examined to find the offending one.

FORTRAN users may sometimes note that some lines of printed output that had clearly executed before the interrupt occurred do not appear on the output. This situation results from the blocking of the output (12 lines per block) that is

used in the cataloged procedures. If it is important for debugging purposes to see the last few lines, an overriding DD statement can be used to unblock the printed output:

```
//GO.FT06F001  DD  SYSOUT=A,DCB=(RECFM=FA,LRECL=133,BLKSIZE=133)
```

For PL/I Users

The PL/I Checkout Compiler, as the name implies, is eminently suitable for debugging PL/I programs. It checks programs very thoroughly, providing clear and detailed diagnostic information (during both translation and interpretation), couched in PL/I terminology for ease of understanding. The manuals, OS PL/I Checkout Compiler: Execution Logic (SC33-0032) and OS PL/I Optimizing Compiler: Execution Logic (SC33-0025), have chapters on debugging from dumps, and a built-in function is available in the language for requesting a dump. However, various high-level program checkout features of the PL/I language and its implementation should make it unnecessary to use a dump.

Additionally, in CMS or TSO, you can communicate with the compiler or the system during translation and with your program or the system during interpretation. You may interrupt execution, inspect or change the values of variables, make corrections to the source, and either continue or restart the execution. Extensive information on the facilities for, and techniques of, interactive debugging can be found in the PL/I Checkout and Optimizing Compilers: Language Reference (GC33-0009), the PL/I Checkout Compiler: TSO User's Guide (SC33-0033), and in Chapter 15 of Class Notes for a PL/I Course (ANL 76-70).

There are several other features for the batch user of both the Checkout Compiler and the Optimizer, which aid in debugging PL/I programs. These are outlined in Sections 13.15 to 13.24 of Class Notes for a PL/I Course (ANL 76-70).

For Assembler Language Users

The TEST command of TSO can be used to debug and verify the proper execution of a program. This command enables you to supply initial (test) values to the program being tested, establish breakpoints within a program where execution will be interrupted for the examination of interim results, display and/or modify register and main storage contents at a breakpoint, display the program status word (PSW), list the contents of control blocks, and step through a section of the program executing one instruction at a time. The TEST

command is described in the Applied Mathematics Division's Implementation of the Time-Sharing Option (Tech. Memo. 262) and in the TSO Command Language Reference (GC28-6732).

REFERENCES

Summarized below are the main references available on languages and compilers at Argonne:

> NOTICE: Documents on CMS and the languages and compilers available on CMS are not listed here. See CMS at ANL: A Primer (Tech. Memo. 339) for a list of relevant documents.

General

TSO Command Language Reference, GC28-6732.
Applied Mathematics Division's Implementation of the Time-Sharing Option, Tech. Memo. 262.
TSC Terminal User's Guide, GC28-6763.

FORTRAN IV

FORTRAN IV Language, GC28-6515.
FORTRAN IV G&H Programmer's Guide, GC28-6817.
FORTRAN IV G1 Programmer's Guide, SC28-6853.
FORTRAN IV G1 Processor and TSO FORTRAN Prompter Installation Reference Manual, SC28-6856.
FORTRAN IV G1 TSO Terminal Users Supplement,
FORTRAN IV H Extended Programmer's Guide, SC28-6852.

PL/I

Class Notes for a PL/I Course, ANL 76-70.
PL/I Checkout and Optimizing Compilers: Language Reference Manual, GC33-0009.
PL/I Checkout Compiler: Programmer's Guide, SC33-0007.
PL/I Checkout Compiler: TSO User's Guide, SC33-0033.
PL/I Checkout Compiler: Messages, SC33-0034.
PL/I Checkout Compiler: Execution Logic, SC33-0032.

PL/I Optimizing Compiler: Programmer's Guide, SC33-0006.
PL/I Optimizing Compiler: TSO User's Guide, SC33-0029.
PL/I Optimizing Compiler: Messages, SC33-0027.
PL/I Optimizing Compiler: Execution Logic, SC33-0025.
A Guide to PL/I for FORTRAN Programmers, SC20-1637.
A Guide to PL/I for Commercial Programmers, SC20-1651.
Preface to PL/I Programming in Scientific Computing, GR20-0312.
Introduction to KStructured Programming in PL/I, GC20-1777.
Introduction to the List Processing Facilities of PL/I, GF20-0015.
PL/I F Programmer's Guide, GC28-6594.

Speakeasy

Speakeasy-3 Reference Manual, ANL 8000, Revision 1.
Guide to TSO Speakeasy, ANL 75-44.

Assembler Language

OS Assembler Language, GC28-6514.
OS Assebler F Programmer's Guide, GC26-3755.
Assembler G User's Guide, handout.
CS/VS-DOS/VS-VM/370 Assembler Language, GC33-4010.
CS/VS-VM/370 Assembler Programmer's Guide, GC33-4021.
CS Assembler H Language, GC26-3771.
CS Assempler H Programmer's Guide, SC26-3759.
CS Assembler H Messages, SC26-3770.

COBOL

OS Full ANS COBOL Reference Guide, GC28-6396.
CS Full ANS COBOL Compiler and Library, Version 4, Programmer's Guide, SC28-6456.

ALGOL

OS ALGOL Language, GC28-6615.
OS ALGOL Programmer's Guide, GC33-4000.

RPG

OS RPG Language Specifications, GC24-3337.

Chapter 6

MATHEMATICAL AND STATISTICAL SUBPROGRAM LIBRARIES

This chapter describes four subprogram libraries available at Argonne: SYS1.AMDLIB, SYS2.AMDLIB, IMSL, and SSP.

AMDLIB consists of several hundred subroutines for the Central Computing Facility. These routines are written in programming languages such as PL/I, FORTRAN, and Assembler. The routines have been put into two on-line libraries, namely, SYS1.AMDLIB and SYS2.AMDLIB.

IMSL, a commercial library, consists of about 350 subroutines accessible through batch or TSO.

SSP is an older IBM package no longer supported by IBM but still available at Argonne.

> NOTICE: Libraries available on CMS are not identified here. For information on CMS see CMS at ANL: A Primer (Tech. Memo. 339), or the June, 1979 issue of the HEX.

SYS1.AMDLIB

The routines in SYS1.AMDLIB were developed at Argonne National Laboratory. These routines are well tested and well documented, and the library is fully supported by the User Services Group.

SYS1.AMDLIB is searched automatically when you invoke the language-processing cataloged routines. Because SYS1.AMDLIB contains the IBM FORTLIB replacement routines, which are more accurate than the equivalent IBM-supplied routines, SYS1.AMDLIB is searched before the standard FORTLIB. However, you may alter the order of search by giving appropriate values to the symbolic parameters.

The contents of the library are listed in AMD Tech. Memo. 261, "KWIK Index and Cross Reference Charts for the ANL Mathematical Subroutine Library" by J. Y. Wang and P. C. Messina. The source codes and the documentation of AMDLIB from the Documentation Center (2-5406) are available.

Below are discussed three of the major software packages in SYS1.AMDLIB.

EISPACK

EISPACK (Release 2) is a package of 58 FORTRAN subroutines primarily dedicated to finding eigenvalues and eigenvectors of matrices and matrix systems. One of the 58 subroutines performs the singular value decomposition of a matrix (SVD), another provides the information for a linear least squares fit (MINFIT), and a third can be used to solve linear systems with symmetric band coefficient matrices (BANDV) (in addition to its principal role in finding eigenvectors of such matrices). The other 55 subroutines are used to solve various classes of matrix eigensystem problems.

In addition to the 58 explicitly-called routines, there are 12 driver-subroutines and 1 control program (EISPAC) in the package. A call to one of the main drivers will automatically call the routines in its particular group. The names of the drivers are shown in Table 7.

For a list of all EISPACK routines and further information, see AMD Tech. Memo. 250, "Path Chart and Documentation for the EISPACK Package of Matrix Eigensystem Routines", by B. Garbow and J. Dongarra.

FUNPACK

The FUNPACK package of special function subroutines is organized into small packets of computationally related subroutines, with only one entry point each and a separate error handling packet containing all I/O statements.

The FUNPACK routines are highly machine-dependent. The versions in the AMD library are certified only for use on IBM System 360 computers. The National Energy Software Center (2-7250) has versions certified for use on the CDC 6000-7000 and Univac 1108 series computers. If FUNPACK routines are to be used on other than IBM System 360 hardware, you should obtain the appropriate version from NESC. Table 8 lists the FUNPACK subroutines.

TABLE 7

EISPACK DRIVERS

NAME	FUNCTION
CG	To determine the eigensystem of a complex general matrix.
CH	To determine the eigensystem of a complex Hermitian matrix.
RG	To determine the eigensystem of a real general matrix.
RS	To determine the eigensystem of a real symmetric matrix.
RSB	To determine the eigensystem of a real symmetric band matrix.
RSP	To determine the eigensystem of a real symmetric packed matrix.
RST	To determine the eigensystem of a real symmetric tridiagonal matrix.
RT	To determine the eigensystem of a certain real tridiagonal matrix.
RGG	To determine the eigensystem for the real generalized eigenproblem Ax=LAMBDA*Bx.
RSG	To determine the eigensystem for the real symmetric generalized eigenproblem Ax=LAMBDA*Bx.
RSGAB	To determine the eigensystem for the real symmetric generalized eigenproblem ABx=LAMBDAx.

LINPACK

The LINPACK package is a collection of FORTRAN subroutines for solving various types of linear systems. The package is concerned with general, banded, symmetric, infinite, symmetric positive definite, triangular and tridiagonal square

TABLE 8

FUNPACK SUBROUTINES

C373S-F2	DELIPK- DOUBLE PRECISION COMPLETE ELLIPTIC INTEGRAL K
C374S-F1	DELIPE- DOUBLE PRECISION COMPLETE ELLIPTIC INTEGRAL E
C375S-F3	DEI- EXPONENTIAL INTEGRALS IN DOUBLE PRECISION
C377S-F2	DDAW- DAWSON'S INTEGRAL IN DOUBLE PRECISION
C395S-F	K0- COMPUTE MODIFIED BESSEL FUNCTION OF 2nd KIND OF ORDER 0
C396S-F	K1- COMPUTE MODIFIED BESSEL FUNCTION OF 2nd KIND OF ORDER 1
C397S-F	BESI0- MODIFIED BESSEL FUNCTIONS OF 1ST KIND OF ORDER 0
C398S-F	BESI1- MODIFIED BESSEL FUNCTIONS OF 1ST KIND OF ORDER 1
C399S-F	PSI- LOGARITHMIC DERIVATIVE OF GAMMA FUNCTION
C301S-F	BESJ0- BESSEL FUNCTIONS OF 1ST KIND OF ORDER 0
C302S-F	BESJ1- BESSEL FUNCTIONS OF 1ST KIND OF ORDER 1
C303S-F	BESYN- BESSEL FUNCTIONS OF 2ND KIND OF NON-NEGA. REAL ORDER
Q056S-F	MONERR- ERROR HANDLING FACILITIES FOR FUNPACK

metrices, as well as with least squares problems and the QR and singular value decompositions of rectangular notices. Each routine has four versions: real single, real double, complex single and complex double precisions. Only the real double and complex double precision version routines are included in SYS1.AMDLIB.

The Basic Linear Algebra subprograms (BLAS) developed by Lawson, Hanson, Kincaid, and Krogh are also included in SYS1.AMDLIB. However, the routines were modified for inclusion in the LINPACK.

For a list of LINPACK routines and further information, see the book, LINPACK Users' Guide by J. Dongarra, J. Bunch, C. Moler, and G. Stewart.

SYS2.AMDLIB

SYS2.AMDLIB is a secondary subroutine library. This on-line version library is not searched automatically but can be accessed through batch cataloged procedures and TSO.

SYS2.AMDLIB serves three purposes:

- to make routines available that are candidates for SYS1.AMDLIB but do not yet meet all the standards.
- to provide a staging area for routines that are being deleted from SYS1.AMDLIB (and consequently from the on-line libraries SYS1.AMDLIB, SYS1.AMDLIB2 on the 370/195; SYS1.AMDLIBO, SYS1.AMDLIB on the 360/75).
- to make routines available that may never be added to SYS1.AMDLIB but are judged to be useful and reliable.

These subroutines may be classified into the following groups:

- Routines for Unconstrained Nonlinear Optimization
- FORTRAN Solutions of Partial Elliptic Differential Equations
- FORTRAN Routine to obtain CPU Timing Information (for segments of a program.
- Real single and complex single precision versions of LINPACK test version of MINPACK1.

The routines in SYS2.AMDLIB are compatible with Computers in the ANL Computer Center. They may not execute properly at other installations.

To access members of the subroutine library through batch cataloged procedures, specify code PRELIB='SYS2.AMDLIB' or

POSTLIB='SYS2.AMDLIB', for example:

```
// EXEC FTHCLG,PRELIB='SYS2.AMDLIB'
```

To access the library on TSO, specify 'SYS2.AMDLIB' in the LIB keyword of a Loader or Linkage Editor invocation, for example,

```
LOADGO XXXX FORTLIB LIB('SYS2.AMDLIB')
```

IMSL LIBRARY

The IMSL library, Edition 6, an extensive commercial library of approximately 350 FORTRAN-callable subroutines, is available in load module form in TSO or batch mode. IMSL subroutines may be classified into the following groups:

Analysis of Experimental Design Data

Categorized Data Analysis
Differential Equations; Quadrature; Differentiation
Eigenanalysis
Forecasting; Econometrics; Time Series
Generation and Testing of Random Numbers
Interpolation, Approximation, and Smoothing
Linear Algebraic Equations
Mathematical and Statistical Special Functions
Nonparametric Statistics
Observation Structure
Regression Analysis
Sampling
Utility Functions
Vector-Matrix Arithmetic
Zeros and Extrema; Linear Programming

The IMSL library is not searched automatically because its current level of usage does not warrant the increased overhead incurred by the search. The IMSL subroutines may be accessed by JCL of the form:

```
//stepname EXEC xxCLG,[PRELIB|POSTLIB]='SYS2.IMSLIB'
```

To access the IMSL subroutine library from TSO, specify 'SYS2.IMSLIB' in the LIB operand of the LINK or LOADGO commands. For example, a FORTRAN main program in the dataset INTEG.FORT calls the IMSL subroutines. The command, FORT INTEG, will cause the FORTRAN IV G1 compiler to create an object version of the main program and store it in the file, INTEG.OBJ.

The command

```
LOADGO INTEG FORTLIB  LIB('SYS2.IMSLIB')
```

will invoke the loader and make available both the FORTRAN library, SYS1.FORTLIB, and the IMSL library, SYS2.IMSLIB, for resolving external references. The resulting program will also be executed.

If the routine has both single- and double-precision versions, the double-precision version will be executed. However, if a single-precision routine calls other single/double-precision routines, the result will be in single precision.

Further information is available from the IMSL Library Users Reference Manual and the consultants.

SCIENTIFIC SUBROUTINE PACKAGE (SSP)

The Scientific Subroutine Package (SSP) is not supported by IBM or AMD. However, you may access SSP subroutines by specifying PRELIB='SYS2.SSP'. You are encouraged to use AMDLIB or the IMSL library as a source for most subroutines because of reported inaccurate results in SSP. Information about AMDLIB and the IMSL library is available from the consultants (2-5405).

To obtain the source codes of SSP, you can find the information in the AMD Tech. Memo. 205, "Retrieving Source Decks for the IBM FORTRAN Scientific Subroutine Package".

Chapter 7

GENERALIZED APPLICATIONS PACKAGES

In addition to the mathematical subroutines described in Chapter 9, we provide several general applications packages. These packages and their purposes are shown in Table 9.

TABLE 9

APPLICATIONS PACKAGES

For:	Sorting and Merging	SM1
	Statistical Analysis	BMDP, SAS, SPSS
	Text Processing	SCRIPT
	Critical Path Analysis, Pert, Pert Cost	PMS, EZPERT
	Database, Information Management Systems	MARK IV, SYSTEM 2000 LIBRARIAN
	Continuous Simulation	CSMP
	Heat-Transfer Systems Analysis	THTB

NOTICE: Applications packages available on CMS are not identified here. For information on CMS see CMS at ANL: A Primer (Tech. Memo. 339), or the June, 1979 issue of HEX.

SORT/MERGE

SORT/MERGE (SM1) is an IBM program operating under OS. It has two basic functions: to sort records in a given sequence and to merge up to 16 previously sorted record sequences into one sequence. Additionally, SM1 can, at various times during execution, be linked with your own routines to perform such tasks as summarizing, inserting, or altering records as they are being sorted or merged. These routines can be included as an object deck in the input stream at execution time or a part of a private library.

SM1 orders your records on the basis of one or more control fields. First the major fields (those specified first) are compared and sorted in the order specified. If the major fields in two records are the same, the program sorts according to minor fields. If the first minor fields in two records are the same, the program compares the second minor fields, and so on until it finds a difference. Up to 64 control fields, both alphabetic and numeric, are permitted.

Since SM1 tends to be I/O bound, you should run the program as a CLASS=A job. This will avoid tying up the I/O devices while SM1 waits for CPU time.

Two cataloged procedures are available: SORT, used when you include user routines, and SORTD, used when you do not include user routines or do not require linkage editing. To invoke SORT, specify:

```
//  EXEC SORT
//SORTIN  DD . . .
//SORTOUT DD . . .
```

To invoke SORTD, specify:

```
//  EXEC SORTD
//SORTIN  DD . . .
//SORTOUT DD . . .
```

If neither catalogue procedure is used, you must specify //SORTLIB.

The IBM manual detailing the product is OS SORT/MERGE Programmer's Guide, (SC33-4007).

STATISTICAL ANALYSIS

Three programs are available at Argonne for statistical analysis: BMDP, SAS, and SPSS.

BMDP Biomedical Computer Programs

The BMDP Biomedical Computer Programs, developed by the Health Sciences computing facility at UCLA for the processing and statistical analysis of data, are available on the batch processors. Although based on the earlier fixed format control BMD series, the 26 BMDP programs use parameter language control and take advantage of newly available statistical techniques and computing algorithms. The BMDP programs provide many new features, such as graphical output, increased options, and transformations not contained in the older BMD series. For example, with the BMDP programs, you can do repeated analyses from the same data file with only minor changes to the control input.

BMDP programs are classified into the categories shown in Table 10.

You need three types of cards to supply information to the computer: system control cards (JCL), program control cards, and data cards. The individual program writeups and the user's manual contain instructions for preparing these cards. For further information, see the BMDP User's Manual.

Statistical Analysis System (SAS)

Description of SAS

The Statistical Analysis System (SAS), supplied by the SAS Institute, is a production oriented system that provides an integrated approach to the editing, summary, and statistical analysis of data. Although SAS is a powerful statistical tool, it can also be used as an information retrieval system or a report generating program because of its extensive database management capabilities.

Two versions of SAS are available: SAS.72, the 1972 version of SAS, and SAS.76, the product of four years of improvements and additions to SAS.72.

TABLE 10

BMDP PROGRAMS

D-series:	Data Description Programs P1D - Simple Data Description P2D - Frequency Count Routine P3D - T Test and T-Squared Routine P4D - Alphanumeric Frequency Count Routine P5D - Univariate Plotting P6D - Bivariate Plotting P7D - Description of Strata with Histograms and Analysis of Variance P8D - Missing Value Correlation P9D - Multidimensional Data Description
F-series:	Frequency Table Programs P1F - Two-Way Contingency Tables
M-series:	Multivariate Programs P1M - Cluster Analysis on Variables P2M - Cluster Analysis on Cases P3M - Block Clustering P4M - Factor Analysis P6M - Canonical Correlation Analysis P7M - Stepwise Discriminant Analysis
R-series:	Regression Programs P1R - Multiple Linear Regression P2R - Stepwise Regression P3R - Nonlinear Regression P4R - Regression on Principle Components P5R - Polynomial Regression P6R - Partial Correlation and Multivariate Regression
S-series:	Special Programs P1S - Multipass Transformation P3S - Nonparametric Statistics
V-series:	Analysis of Variance Programs P1V - One-way Analysis of Variance and Covariance P2V - Analysis of Variance and Covariance Including Repeated Measures

SAS.76 provides a comprehensive set of routines for the statistical analysis of data, as shown in Table 11.

TABLE 11

SAS.76 ROUTINES

ANOVA	- analysis of variance for balanced designs
AUTOREG	- autoregression
BMDP	- calling of a BMDP program from within SAS
CLUSTER	- cluster analysis
CONTENTS	- printing of contents and history of SAS datasets
CONVERT	- converting of SAS72, SPSS, OSIRIS, BMDP or Datatext files to SAS datasets
CORR	- correlation coefficients
DISCRIM	- discriminant analysis
DUNCAN	- Duncan's multiple range test, Waller-Duncan K-ratio T-test
FACTOR	- factor analysis
FREQ	- frequency and crosstabulation tables
GLM	- least-squares fitting of univariate and multivariate models
GUTTMAN	- Guttman scaling
MATRIX	- matrix manipulation
MEANS	- simple univariate descriptive statistics
NEIGHBOR	- nearest neighbor discriminant analysis
NESTED	- analysis of variance and covariance for nested designs
NLIN	- nonlinear regression
PLAN	- randomized plans for experiments
PRINT	- printing of datasets
PROBIT	- probit analysis
RANK	- ranking
RSQUARE	- all possible regressions
SAS72	- calling of a SAS72 procedure
SCATTER	- plotting
SCORE	- factor scores
SORT	- sorting of SAS datasets
SPECTRA	- spectral analysis
STANDARD	- standardization
STEPWISE	- stepwise regression
SYSREG	- least squares for interdependent systems of linear equations
TTEST	- T-tests
VARCOMP	- variance component estimation

Use of SAS

In batch, the following cataloged procedures are available:

SAS	- to invoke the current version of SAS, SAS76	(150K)
SAS76	- to invoke the current version of SAS, SAS76	(150K)
SAS72	- to invoke the 1972 version of SAS	(200K)
SASBOTH	- to call the SAS.76 procedure SAS72	(200K)
SASBMDP	- to call the SAS.76 procedure BMDP	(300K)

You can invoke these by providing the appropriate statement in JCI, for example,

```
//  EXEC  SAS
//SYSIN DD *
     SAS statements
/*
```

In TSO, only SAS.76 may be invoked, and you must logon in at least the 140K region. The following TSO commands are available:

SAS	- to allocate the necessary files and invoke SAS.76
SASGO	- to invoke SAS.76 again in the same TSO session, using files previously allocated
SASSORT	- to allocate the work files needed for sorting

Table 12 shows the datasets required for SAS.

TABLE 12

DATASETS FOR SAS

Datasets for SAS76	
SYS1.SAS76.LIBRARY	load module library
SYS1.SAS76.MACRO	SAS OS/Assembler macro library
SYS1.SAS76.PL1MACRO	SAS PL/1 macro library
SYS1.SAS76.USER.LIBRARY	user written procedure library

Datasets for SAS72	
SYS1.SAS.LIBRARY	load module library
SYS1.SAS.SLIBRARY	supplemental procedures library

For further information, see:

A User's Guide to SAS.76

SAS Programmer's Guide

SAS Supplementary Library User's Guide

Statistical Package for the Social Sciences (SPSS)

Description of SPSS

The Statistical Package for the Social Sciences (SPSS), an integrated system of computer programs developed by the National Opinion Research Center at the University of Chicago, is available on the batch processors.

SPSS offers a variety of statistical routines shown in TAble 13.

TABLE 13

SPSS ROUTINES

1.	Descriptive statistics
2.	Simple frequency distributions
3.	Cross tabulations
4.	Simple correlation
5.	Partial correlation
6.	Means and variances for stratified subpopulations
7.	Analysis of variance (1 and n-way)
8.	Multiple regression
9.	Discriminant analysis
10.	Scatter diagrams
11.	Factor analysis
12.	Canonical correlations
13.	Guttman scaling

You can perform an analysis through 'natural language' control statements designed to make the system easily accessible to people with no prior computer experience.

Version 1.02, which will process up to 1000 variable and 200 subfiles, is currently available. A cataloged procedure, SPSS, which executes in 250K and contains the basic JCL statements required for execution of every SPSS job, can be invoked by the following JCL statement:

```
// EXEC SPSS
```

Two resource books are available from the AMD Documentation Center: SPSS, the complete guide including documentation for all versions of batch SPSS; and SPSS Primer, a brief introduction in simple nontechnical language.

SCRIPT -- A TEXT FORMATTING PROGRAM

SCRIPT is a computer program, developed by the University of Waterloo, that accepts input (text and control words), processes it (text formatting), and produces output (a document in which all the text has been formatted according to the instructions specified by the control words). To use SCRIPT, you must use a text editor, such as WYLBUR, to enter, correct, delete, add, and save text and control words.

Use of SCRIPT

The standard production version currently uses SCRIPT 3.3. To access an earlier version of SCRIPT, you should use the TSO commands, 'SCRIPTn' (on-line), and SCRIPTn (TSO batch submission) or the cataloged procedure SCRIPT (batch), or execute the program SCRIPT1. SCRIPT 3.4 is available under CMS.

The TSO command, SCRIPT, will automatically invoke the most current version; however, you must be sure to LOGON specifying at least SIZE(140): Logon Bnnnnn/password size(140). Similarly, users of batch SCRIPT should specify at least REGION=140K.

Manuals are available from the Documentation Center. Microfiche copies are also available.

CRITICAL PATH ANALYSIS: PERT, PERT COST

Two programs are available for critical path analysis: PMS and EZPERT.

Project Management System IV (PMS IV)

PMS IV is an IBM program, available on the batch processors, that performs functions related to many management applications. PMS IV contains an extended Network Processor, a Cost Processor, Resource Allocation Processor, and a Report Processor. Together, these processors provide critical path analysis, PERT, and PERT cost.

PMS IV was designed so that an analyst can understand, modify, and add to the program with minimum difficulty. To accomplish this objective, all the processor modules of PMS IV are divided into procedures, which in turn are divided into subroutines. The subroutines, procedures, and processors are usable in various combinations. Some of the procedures and subroutines perform functions that are unique to a given processor, while others perform functions common to more than one.

All the procedures of a processor are described separately. The descriptions explain the functions that each procedure is performing and refer to the subroutines that they are using. Similarly, all the subroutines have separate write-ups and flowcharts pointing out the boundary conditions required by each of these subroutines for successful operation. You can add new subroutines if you observe the appropriate boundary rules.

You may also change the system at the procedure level, through a set of procedure commands. These commands are used extensively in the Report Processor to enable you to change both the format and content of reports without reprogramming. You can specify a SYS2 prefix for the appropriate processor as shown:

```
                                SYS2.PMSXNP
//STEPLIB DD DUP=SHR,DSN=SYS2.PMSCOST
                                SYS2.PMSRAP
                                SYS2.PMSREP
```

Table 14 lists the component datasets for PMS.

Application Description (H20-0210) provides an overview of the components and capabilities of PMS IV. More detailed information on the various processors is provided in the following: PMS IV Network Processor Program Description and Operations Manual, PMS IV Report Processor Program Description and Operations Manual, PMS IV Cost Processor Program Description and Operations Manual, and PMS IV Resource Allocation Program.

TABLE 14

DATASETS FOR PMS

DSNAME	DESCRIPTION
SYS1.PMSREP	Report Processor load modules and link library
SYS1.PMSMAC1	Report Processor assembler macro library
SYS1.PMSMAC2	Report Processor assembler macro library
SYS1.PMSCOST	Cost Processor modules
SYS1.PMSXNP	Extended Network Processor modules
SYS1.PMSRAP	Resource Allocation Processor modules

EZPERT

Description of EZPERT

If you use IBM's Project Management System (PMS) you may be interested in Systonetic's EZPERT graphics package, available on the batch processors. EZPERT accepts input in the form of PMS REPFILE output to produce CalComp 936 plots showing cost, schedule, and resource information for project management.

EZPERT reads the project data, accepts English language control commands supplied by you, and generates an output file that drives the plotter to produce the desired networks, charts, etc. Any of seven different commands, shown in Table 15, can be used to specify the mode of operation.

The output from EZPERT consists of a plot file (tape or disk) that drives the plotter and a printout that contains information about each plot produced.

An extensive set of diagnostic messages is provided to help resolve any problems that may occur during a run.

> NOTE: You must use the newest version of EZPERT if you want to maintain compatibility with the new version of PMS IV. Specify:
>
> ```
> //GOEZPERT.STEPLIB DD DISP=SHR,DSN=SYS2.EZPERT1.FRAMED
>                                 SYS2.EZPERT2.FRAMED
>                                 SYS2.EZPERT1.NORMAL
> ```

TABLE 15

EZPERT COMMANDS

COMMAND	INTERPRETATION
PRENET(TASKCHART)	Operation is in the Prenet Mode, and all plots are to be Task Charts (no dependencies among activities).
PRENET(LOGIC)	Operation is in the Prenet Mode, and all plots are to be Logic Networks showing dependencies among activities.
BARCHART(SINGLE)	Operation is in the Gantt Barchart Mode. All charts are produced from a single schedule report. This command is used most frequently when the selection options are invoked to extract activities and produce several different charts from a single input report.
BARCHART(BATCH)	Operation is in the Gantt Barchart Mode. The input file contains a report for several different subnets, and each chart is produced from a separate report.
NETWORK(SINGLE)	Operation is in the Network Mode. All charts are produced from a single schedule report - typically used in conjunction with the selection options for producing several "fragnet" plots from a single large network report.
NETWORK(BATCH)	Operation is in the Network Mode. The input file contains several different subnets, and each network diagram is produced from a separate report.
XYCHART	Operation is in the Cost/Resource Mode. Any number of different graphs may be produced during the run.

Further information is available from the EZPERT User's Guide, Tech. Memo. 270,

DATABASE AND INFORMATION MANAGEMENT SYSTEMS

Three programs are supported by AMD for database management: MARK IV, System 2000, and LIBRARIAN.

MARK IV (File Management System)

MARK IV (Version 6.0) is a program designed by Informatics, Inc. for file manipulation and management. The description of the files is independent of a file itself. The structure and format of a file, the records within that file, and the transactions used to create and update files are defined to the MARK IV system and stored in a dictionary.

You can make requests to select entries from the file, to select specified data from the entries for computation and logical processing, and to specify the desired output. Requests that are used repetitively can be stored in the MARK IV system catalog and subsequently invoked by specifying the request group name. Output may take the form of reports, intermediate result files, subsets of the original file, or a combination of all of these. The system can process multiple input files simultaneously; in a single run, the system can read 1 master, 1 transaction and 9 coordinated files. It may produce 1 master-out, 10 subfiles, 1 deleted master, and 1 rejection roster, with a maximum of 255 different reports. Other features of MARK IV include:

- Unlimited storage for temporary fields
- Fixed or variable, blocked or unblocked record format
- Concatenation of unlike datasets
- Sequential or indexed sequential access method
- Hierarchical data structure with up to nine levels
- Non-procedural language
- Use of any IBM 360/370 valid data attributes
- Variable spanned record support
- Storage resident ISAM index

Job Control Language for MARK IV must be developed for each application. See the Consultants for assistance.

The following references are available from Informatics, Inc.:

MARK IV Reference Manual (SM-7501-P01B-1)

MARK IV User's Guide (SP-69-810-3)

MARK IV Operations Guide (SM-7501-P02B)

MARK IV Practiques Handbook (SM-7306-P04A)

MARK IV Special Features Manual (SM-7408-M47A)

MARK IV Quick Reference Folder

Technical Information Bulletin (SM-7501-T43F)

SYSTEM 2000 (Data-base Management System)

System 2000 is a general purpose database management system from MRI Systems Corp. It provides a user-oriented language containing a full complement of database management capabilities, including database definition, creation, update, retrieval, report generation, transaction logging, and interfaces with FORTRAN and PL/I.

System 2000 retains archival copies of databases and records an audit trail of changes made to a database. It is capable of reconstructing a database by applying the audit trail to an archival copy of that database. Data may be manipulated by the programming languages, FORTRAN and PL/I, or a user-oriented language for non-programmers.

The system contains functions for finding the maximum, minimum, average, sum, standard deviation and counts when accessed through the user-oriented language. There is also a report-generation capability designed for people with limited computer experience.

Three cataloged procedures are available: S2KFTH (for FORTRAN), S2KPLO (for the PL/I optimizer), and S2KNAT (for the Natural Language).

System 2000 may be invoked through TSO. You must logon in the 200K region, using the SYS2K or S2KLOGON procedures: for example, logon proc(sys2k). SYS2K is the preferred logon procedure, since it allows larger scratch files to be built. System 2000 file definitions and loader-strings must

be non-numbered card-image files. Since System 2000 files (1-6) have direct organization, they must be created through batch mode. Because of TSO local system queue attribute limits, space extents should be kept to a minimum.

A card-image file called S2KPARMS is used for the System 2000 Report Writer.. There are two data cards in this file: AREAS=6 and RW=YES. In TSO mode, AREAS=6; in batch mode, AREAS=10, if the Report Writer is desired.

System 2000 reference guides and manuals are available from the Documentation Center.

LIBRARIAN

The LIBRARIAN source program retrieval and maintenance system is available to store, on disk or tape, the source programs, test data, job control statements, or any other information normally stored on cards.

Storage on tape or disk is in compressed "master" file format, by the automatic elimination of strings of contiguous like characters from card images. Back-up speed and reduced storage are obvious benefits of data compression. Also, because LIBRARIAN re-uses its library space automatically, you need not be constantly concerned with file reorganization. If you update individual modules on tape, you must generate a whole new master. You may find it helpful to move individual modules from tape to disk if they become very active.

Library updating is achieved by six basic commands: ADD (add a new module to a master file), DLM (delete a module from the master), SEL (select a module to be updated and/or compiled, listed or processed in other ways), INS (insert new cards after a selected card), REP (replace old cards with new), and DEL (delete existing cards).

In addition, the EDIT command invokes a search of a module (or portion of a module) for a certain string of characters and replaces it with a new string. SCAN performs only the search without making a replacement. FILL puts a specified string of characters into chosen card positions without checking for existing contents. All changes in a run may be specified as temporary. The Group Processing Option provides for accessing groups of modules for multiple updates. Password, copy and date options are also available.

LIBRARIAN is available in TSO. You may add a new module to a master file, retrieve or replace existing modules, and display at a terminal the names and characteristics of any

modules. These activities are controlled by three TSO commands:

LIBSAVE - to add a new module to a LIBRARIAN master file, or to replace one or more existing modules in a master file. In either case, a specified TSO sequential dataset is transferred, in its entirety, to a LIBRARIAN master file.

LIBGET - to retrieve a module from a LIBRARIAN master file. This action transfers the module from the master to a dynamically allocated TSO sequential dataset.

LIBINDEX - to obtain a listing of all modules associated with a particular programmer name on a LIBRARIAN master file.

Prompting and diagnostic messages issued by all commands are self-explanatory; many messages have second- and third-informational levels that you can obtain by entering a question mark at the terminal.

The following reference information is available from the Documentation Center:

LIBRARIAN OS User Routine Abstracts

LIBRARIAN TSO User Reference Manual

LIBRARIAN Release Notice 5.2

LIBRARIAN User Reference Manual

LIBRARIAN Summary of Messages

LIBRARIAN Updates

LIBRARIAN Concepts and Facilities

CONTINUOUS SYSTEMS MODELING PROGRAM III (CSMP III)

CSMP is an IBM program designed for the digital simulation of continuous processes. The Input language enables you to prepare statements describing a physical system, starting from either a block diagram or a differential equation representation of that system.

Since CSMP is an outgrowth of analog computer techniques, you can proceed directly from an analog computer block diagram to the digital solution. As a logical extention of

this fact you can simulate a hybrid (analog/digital) computer.

You describe the system to be simulated to the program by a series of structure, data, and control statements. The program includes a basic set of functional blocks with which the components of a continuous system may be represented, plus means for you to define functions suited to your particular simulation requirements. Therefore, CSMP can take on the characteristics of a language oriented to any particular special purpose field in continuous system simulation. You use application-oriented input statements to describe the connections between the functional blocks.

CSMP also accepts FORTRAN statements, thereby allowing you to handle complex nonlinear and time-variant problems. A translator converts structure statements into a FORTRAN subroutine, which is then compiled and executed alternately with a selected integration routine to accomplish the simulation. Centralized integration ensures that all integrator outputs are computed simultaneously at the end of the iteration cycle. Several different types of routines are available to perform the integration operation, including five fixed step-size routines (rectangular, trapezoidal, Simpson's second-order Adams, and fixed step Runge-Kutta) and four variable-step routines (fourth-order Runge-Kutta), one of which handles stiff equation, and fifth order Milne predictor-corrector). An alternative fourth-order Runge-Kutta routine performs the integration in double precision.

An entire CSMP simulation can be controlled by a sequence of FORTRAN statements executed only at the termination of a simulation run. A user-written FORTRAN program can test run responses, define run control conditions, and supervise both input and output information, providing a method for handling the type of iterative computation involved, for example, in automatic search procedures for parameter optimization.

Format options permit tabulating, print-plotting, or producing contoured or shaded print-plots of several variables or the results of several simulation runs on a single grid.

Two cataloged procedures, CSMPGLG and CSMPGX, will invoke a CSMP III model. Some examples are shown in Table 16.

By means of parameter information in the PARM field of the CSMP EXEC statement, you can specify the CSMP III translator, FORTRAN, linkage editor, and execution phase options used.

TABLE 16

CSMP EXAMPLES

// EXEC CSMPGLG	no plots
// EXEC CSMPGLG,PLOTTER=PRINTER	crude plots
// EXEC CSMPGLG,PLOTTER=S35BW	for FR-80
// EXEC CSMPGLG,PLOTTER=VERS11	for VERSATEC
// EXEC CSMPGLG,PLOTTER=CAL780	for CALCOMP780
// EXEC CSMPGLG,PLOTTER=CAL580	for CALCOMP580

Following is a list of differences between the IBM examples given in their reference manual and the procedures followed at Argonne.

- Temporary space allocations are between 3.5 and 7 times the amount given in the IBM example.
- The PARM.X reference is unnecessary.
- AMDLIB, IMSLIB, DISLIB are available to CSMP. The parameters PRELIB and POSTLIB allow you to add your own choice of load module libraries.
- TRANPGM=DEJTNOV, a nonoverlay translator module, replaces the IBM default DEJCSMPT. TRANMID (110K overlay) is also available.
- LINKMEM=CTLNOV, a nonoverlay execution module, replaces the IBM default, CTLCDS (a 102K overlay). CTLMID (110K overlay) is also available.
- Output files are blocked.
- Temporary files SYSLIN, SYSLMOD, SYSUT1 are given unit separation.
- All CSMP FORTRAN subroutines are recompiled on the H compiler using optimization level 2.

- FORTPGM=IEKAA00, OPTIONS='OPT=2' specifies the H compiler optimization level 2, replacing the IBM default G compiler.

- All system datasets are prefixed by SYS1. The datasets are cataloged, and the VSYS parameter has been removed.

- The maximum number of statements, macros, differential equations, etc. has been raised above the original IBM levels.

The operation of CSMP III requires several datasets. Table 17 gives a list of the datasets, by DDname, and a description of their use:

All of the above datasets (except FT15F001) and GRAPHICS, are assigned to direct-access devices. All may, however, be assigned to tapes, except those referred to as direct access, namely, CSMPLIB, STEPLIB, SYSLIB, SYSLMOD, and SYSUT1.

Several IBM manuals introduce and describe the CSMP III system:

Continuous System Modeling Program III (CSMP III)

(CSMP III Graphic Feature) General Information Manual.

CSMP III Program Reference Manual (SH19-7001)

A Guide to Using CSMP -- A Program for Simulating Physical Systems, Frank Speckhart and Walter Green, Prentiss-Hall, 1976.

TABLE 17

CSMP DATASETS

DDname	Description
COMPRINT	SYSOUT dataset for FORTRAN IV Compiler phase
CSMPLIB	Direct-access partitioned input dataset containing members named in INCLUDE Statements in the CSMP III model input dataset (whose ddname is SYSIN)
FT01F001	The output dataset used as input to the CSMP III translator phase
FT02F001	The output dataset used to punch the symbolic deck if the DECK option is chosen
FT03F001	SYSOUT dataset for translator input in case of error in the input processor
FT05F001	Output dataset containing data records used as input by the CSMP III execution phase
FT06F001	SYSOUT dataset for the execution phase
FT07F001	Output dataset used as input to the FORTRAN IV compiler phase
FT08F001	SYSOUT dataset for the translation phase
FT13F001	Intermediate dataset for output and preparation of output
FT14F001	Intermediate dataset during the translation phase for translated structure statements and during the execution phase for OUTPUT, PAGE, and LABEL statements
FT15F001	Output dataset containing plot information if the PREPARE statement is used in the CSMP III models (Otherwise, a null dataset will suffice.)
GRAPHICS	Sequential output for driving graphics

	devices
STEPLIB	Direct-access partitioned input dataset containing the CSMP III load modules.
SYSLIB	Direct-access partitioned input datasets containing the CSMP III and FORTRAN IV execution load modules, that are automatically called during the linkage editor phase
SYSLIN	The object module output, from the FORTRAN IV compiler, which will be part of the primary input to the linkage editor phase
SYSLINK	Primary input to the linkage editor phase, consisting of the output object modules from the FORTRAN IV compiler (SYSLIN) followed by the dataset containing the linkage editor control cards for the CSMP III execution phase
SYSLMOD	Direct-access partitioned output dataset containing the execution phase load module created by the linkage editor
SYSPRINT	SYSOUT dataset for the linkage editor phase
SYSUT1	Direct-access intermediate scratch dataset used by the linkage editor phase
SYSIN	Input dataset containing the CSMP III model(s)

TRANSIENT HEAT TRANSFER, VERSION B (THTB)

THTB is a proprietary code written originally by General Electric for the IBM 704. It is capable of analyzing general three-dimensional heat transfer systems using a finite difference method. The basic function of the program is to set up the general heat balance equation for each node point, to compute the terms that apply from the input given, and to solve the resulting set of equations by the Gauss-Seidel method for the central temperatures of the nodes. The process is repeated for each time increment using the temperatures from the previous time increment as approximations tc the new temperatures.

The original program was revised at Argonne in the mid 1960's and reprogrammed for the CDC 3600. The version, called RE 322, was later modified to fit on the CDC 1604 at the Idaho Site. The IBM 360 version of THTB is a result of the National Reactor Testing Station - Computer Science Center conversion of this CDC-1604 version to their IBM 360/75.

Seven modes of heat exchange are treated by the program: conduction, convection, grey body diffuse radiation, surface flux, internal generation, non-sink mass flow, and latent heat effects.

Either of two separate load modules may be accessed by the procedure THT. The smaller, or default, version (250 K region) is stored in the cataloged dataset SYS2.THTBL8. Specify:

```
// EXEC THT
//GO.SYSIN DD *
```

The large version (450 K region) is stored in SYS2.THTBL9. Specify:

```
// EXEC THT,VERSION=THTBIG
//GO.SYSIN DD *
```

You may also generate a listing of the user input data by adding:

```
//*FORMAT PR,DDNAME=ASPI0001
```

For further information on running THTB problems and on coding improvements, see Tech. Memo. 281, "THTB at ANL," and "THTB/(GE) Three-Dimensional Transient Heat Transfer," Program Library 2209/kRE 322, revised August 21, 1966.