GX20-1926-3

IBM

Reference Summary

IBM Virtual Machine Facility/370: Quick Guide for Users

IBM Corporation, VM/370 Publications,
New England Executive Pk., Burlington, Mass. 01803

GX20-1926-3

IBM® **Reference Summary**

IBM Virtual Machine Facility/370 Quick Guide for Users

This guide describes some of the essential VM/370 operations for the new user. It also provides a brief description of all VM/370 commands for the experienced user. Only a limited amount of prior VM/370 knowledge is assumed for the section on VM/370 operations. See the "Preface" for prerequisite publications.

The user of the command description section should have a thorough understanding of VM/370 command syntax and usage. The CP, CMS, and RSCS commands are summarized in the *Reference Summary,* GX20-1961. This is part of Bill of Forms Order No. GBOF 3576.

Fourth Edition (May 1975)

This is a major revision of GX20-1926-2 and makes obsolete that edition. This edition, GX20-1926-3, corresponds to Release 2 PLC 13 (Program Level Change) of the IBM Virtual Machine Facility/370, and to all subsequent releases unless otherwise indicated in new editions or Technical Newsletters.

Changes are periodically made to the specifications herein; before using this publication in connection with the operation of IBM systems, refer to the latest IBM System/370 Bibliography, Order No. GC20-0001 for the editions that are applicable and current.

Changes and additions to text and illustrations are indicated by a vertical bar to the left of the change.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A handbook-sized binder, FE part number 453559, may be purchased from IBM. Customers may order it through their IBM marketing representative. IBM personnel should order it as an FE part from Mechanicsburg.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, VM/370 Publications, 24 New England Executive Park, Burlington, Massachussetts, 01803. Comments become the property of IBM.

PREFACE

This publication contains information for both the new user and the more experienced VM/370 user.

The sections, "What You Should Know Before You Start Using the VM/370 System" and "VM/370 System Information", should help the new VM/370 user become acquainted with the system. These sections contain information for getting started and setting up a virtual machine.

The section, "Using CMS", discusses using the CMS facilities to create and update files.

The section, "Summary of VM/370 Commands and Service Aids", is an alphameric listing of all the CP and CMS commands and the VM/370 service aids. It is intended for the experienced VM/370 user.

The section "System/370 General information" contains reference information from the following publications:

IBM System/360 Principles of Operation, GA22-6821

IBM System/370 Principles of Operation, GA22-7000

OS/VS, DOS/VS, and VM/370 Assembler Language, GC33-4010

This publication and the Reference Summary, GX20-1961, are a part of Bill of Forms Order No. GBOF 3576.

The new user should use the following VM/370 manuals in conjunction with the first sections of this publication.

PREREQUISITE PUBLICATIONS

IBM Virtual Machine Facility/370:

Introduction, GC20-1800

Command Language Guide for General Users, GC20-1804

EDIT Guide, GC20-1805

EXEC User's Guide, GC20-1812

COREQUISITE PUBLICATIONS

IBM Virtual Machine Facility/370:

System Messages, GC20-1808

Terminal User's Guide, GC20-1810

Experienced users should be familiar with the content of the following publications:

IBM Virtual Machine Facility/370:

Planning and System Generation Guide, GC20-1801

System Programmer's Guide, GC20-1807

Summary of Amendments for VM/370 Quick Guide for Users, GX20-1926-3 VM/370 Release 2 PLC 13

This publication reflects support for the following additions to the Virtual Machine Facility/370:

- 3270 display terminals
- RSCS (Remote Spooling Communications Subsystems)
- Other minor CP and CMS command and operand modifications to support new functions and devices.

Some technical corrections have also been made.

Summary of Amendments
for VM/370 Quick
Guide for Users, GX20-1926-2
VM/370 Release 2 PLC 4

This publication has been substantially changed and enhanced for this edition, notably with the

- Inclusion of information about terminal operation characteristics
- Inclusion of tables regarding filemodes and filetypes
- Inclusion of CP and CMS commands that were removed in the prior edition. Also, VM/370 service aids have been added to this section
- Inclusion of a summary of the information contained in the Principles of Operation and Assembler manuals listed in the Preface.

Some technical corrections have also been made.

CONTENTS

FIGURES

WHAT YOU SHOULD KNOW BEFORE YOU START USING THE VM/370 SYSTEM

The environment of the IBM Virtual Machine Facility/370 (VM/370) is one of virtual machines. A virtual machine is a functional simulation of a real computer and its I/O devices. VM/370 builds and maintains, for each user, a virtual System/370 machine from a predefined configuration.

The virtual machine configuration includes components corresponding to a real System/370: a virtual operator's console, virtual storage, a virtual CPU, and virtual channels and I/O devices. However, since the virtual machines are simulated, their configurations may differ from each other and from the real machine. For example, the real machine may have 512K bytes of real storage and eight real disk drives, while a virtual machine may have 768K bytes of virtual storage and two virtual disk drives.

Regardless of the configuration, you control your virtual machine from your terminal, which is, effectively, your operator's console. The work to be done by the virtual machine is scheduled and controlled by an operating system that can run under VM/370. An example of a virtual machine operating system is the Conversational Monitor System (CMS), which was specifically designed to run in a virtual machine under control of VM/370. CMS provides, at a remote terminal, a full range of conversational capabilities:

- Creation and management of files

- Compilation, testing, and execution of problem programs

- Execution of application programs

The section "Using CMS" describes how you can use a CMS virtual machine under VM/370.

Before you can start using VM/370 you must have:

- A user identification and password.

- A virtual machine defined for your use. (The virtual machine definition should include all the devices you expect to use. For example, a console, spooled unit record devices, and disk space.)

- Properly formatted disk space. (If you wish, you may format your disk space after you log on.)

GETTING STARTED (IDENTIFICATION AND PASSWORD)

Before you can use VM/370, you must be assigned:

- A user identification (userid) that identifies you to the system, and

- A password that is checked when you log on.

(Examples in this guide use a userid of PUBS.) Assignment of a userid and password is normally handled (and approved) by the VM/370 system operations group.

Once you have your userid and password, you can communicate with the VM/370 system from a remote terminal such as an IBM 2741 Communications Terminal, IBM 1050 Data Communication System (or equivalent) or 3277 display terminal. Depending on your terminal installation, you either dial the central VM/370 computer, or are connected directly (through an appropriate control unit if necessary). For a description of the communication procedures for each type of terminal, see the _VM/370_: _Terminal User's Guide_.

VIRTUAL MACHINE CONFIGURATION

When you have a virtual machine defined for you, an entry is made in the control program directory. The systems operation group usually sets up your directory for you. This directory lists the devices and device addresses available to your virtual machine. The following is an example of a typical CMS virtual machine configuration.

Virtual Device	Virtual _Device Address_
Console	009
Card reader	00C
Card punch	00D
Printer	00E
CMS system disk	190
Primary disk for user files	191
CMS system disk extension	19E

FORMATTING DISK SPACE

All disk space must be formatted for use with CMS. The systems operations group usually makes sure your disk space is formatted. The CMS Format program is executed under CMS via the FORMAT command. An example of the CMS FORMAT command is in the section "Using CMS."

VM/370 SYSTEM INFORMATION

TYPING CONVENTIONS

Because certain special characters can be assigned logical editing functions to enter input data via VM/370 terminals, the following typing conventions should be observed. Input data may be entered in either uppercase or lowercase. The examples in this book show the lines you might enter in shading. System responses may be in uppercase or lowercase.

Note: The logical line edit characters shown below are line edit default values. Your installation or your terminal may require other characters to fulfill the line edit function.

Character Delete symbol (@): The character delete symbol deletes the preceding character in the input line. A string of "n" character delete symbols delete the preceding "n" characters in the input line and itself.

Line Delete symbol (¢): The line delete symbol deletes all characters in the current logical line and itself. A line delete symbol cannot be deleted by a character delete symbol.

Line End symbol (#): The line end symbol indicates the end of a logical input line. Use of this character permits more than one logical input line to be entered in the same real input string.

Logical Escape symbol ("): The logical escape symbol causes the character following it to be interpreted as a data character (that is, ignored as an input line editing character). This allows any of the line editing characters to be interpreted literally. For example, consider how the following line must be entered into the system:

```
1 gross #2 pencils @ 92¢ per dozen
```

Under the VM/370 input conventions, this line could not be entered as shown, since it would be affected by the #, @, and ¢ line editing symbols. For example, the # symbol would end the line. However, the line is correctly interpreted if entered as follows:

```
1 gross "#2 pencils "@ 92"¢ per dozen
```

The logical escape characters (") are not put in the file.

Line Length: For CP console functions and CMS commands, input line length is restricted to the physical line entry limitations imposed by the terminal device, or the default record length. Lines exceeding the maximun number of characters, (including blanks, backspaces, underscores, the line editing characters, and the tab character), are truncated to that line length value.

Line Termination: An input data line from an IBM 2741 Communications Terminal is transmitted to the computer by pressing the Return key. The same function performed on the 3277 display terminal is accomplished by pressing the Enter key. Other terminals have similar line termination keys.

Note: For some terminals, such as the IBM 1050, you have to press an Alternate Coding key and some other multiple-function key at the same time.

TERMINAL OPERATING PROCEDURES

For a description of the various terminal operating procedures, see the *VM/370: Terminal User's Guide*.

LOGGING ON

When you have established communication with the VM/370 computer, the system sends a VM/370 ON LINE message to the terminal.

On certain teletypewriter terminals this message may appear adjacent to 12 meaningless characters. Ignore these characters.

On 3277 display terminals this message is apparent by the display of VM/370 logo in conjunction with the System Available lamp being lit.

Press the Attention (ATTN) key, PA1 key (or equivalent) and identify yourself by entering your user identification (userid) as follows:

```
   logon pubs
```

Then press the Return key, the carriage return key, the Enter key (or your terminal's equivalent). If the userid entered is not found in the CP directory, the following message is sent to the terminal:

```
   DMKLOG053E  userid NOT IN CP DIRECTORY
```

Assume, however, the userid entered is found in the CP directory, the VM/370 system responds with:

```
ENTER PASSWORD:
```

At this point, you should enter your password, and then press the Return key or its equivalent.

Note: For security purposes many supported VM/370 terminals provide a masking technique so that the password is not displayed or printed. Depending upon the entry technique the password may or may not be displayed on these terminals. For more information on using the Print Inhibit feature, see the *VM/370: Terminal User's Guide*.

The system then waits for you to enter your password. If the password entered is incorrect, the message

```
DMKLOG050E  PASSWORD INCORRECT
```

is sent to the terminal. You must start the logon procedure from the beginning by entering your userid again. If you do not do this, you are prompted by the message:

```
RESTART
```

If the userid and password entered are valid, but someone else has already logged on with this userid, the VM/370 system issues the message:

```
DMKLOG054E ALREADY LOGGED ON LINE nnn
```

where nnn indicates the line on which the user is logged. If you want to find out why the userid you just entered is in use, issue the MSG command to send a message to the operator or to the other user. You should either log on with another userid (if another userid is reserved for your use) or try again later.

Once you have successfully logged on, the VM/370 system replies with a log message, such as:

```
LOGON AT 11:24:35 EST THURSDAY mm/dd/yy
```

A logon message from the VM/370 operator (if any) also prints at this time.

Once you have successfully logged on the VM/370 system, you can start using the virtual machine that you have set up for your userid.

LOGGING OFF

When you are finished using the system and want to end your terminal session, you do so by logging off of the VM/370 control program (CP). Even if you are in CMS mode, you need only enter the command:

```
logoff
```

and press the Return key (or its equivalent). The system responds with:

```
CONNECT= 00:11:43  VIRTCPU= 000:05.21
   TOTCPU= 000:21.03
LOGOFF AT 11:34:44 EST THURSDAY 11/30/72
```

and the connection with the VM/370 system is terminated. The connect time is in hours, minutes, and seconds. The use of the virtual CPU and total CPU is given in minutes, seconds, and hundredths of a second. Only when the logoff procedure is completed, should you turn terminal power off.

Note: If you logged on over a dialed line, you could specify that the communication line be left connected, by issuing

```
logoff hold
```

When you issue LOGOFF HOLD, you do not have to dial the line before logging on again.

VM/370 ENVIRONMENT

Each input line entered at the terminal by a user is transmitted to the VM/370 system, where it is processed (examined, and accepted or rejected) by a given routine. The portion of VM/370 that has control at the time a particular input line is entered determines which routine processes the input. Each portion of the VM/370 system that can accept input constitutes a unique environment, and only a subset of all possible input is acceptable to any given environment.

Four input processing environments exist:

- Control Program
- Central CMS service routines
- CMS command environments (DEBUG, EDIT, EXEC, or a user-written command)
- RSCS (Remote Spooling Communications Subsystem)

Input lines that are acceptable to the CP, CMS, and RSCS environments are referred to as commands.

Certain CMS commands cause CMS subenvironments (modes) to be entered. Examples of these are the DEBUG and EDIT commands. Lines acceptable to the EDIT environment are referred to as subcommands, or input, depending on the particular mode that is entered when the command is issued and subsequent user action.

When the EDIT command is keyed in, the edit mode is entered, regardless of the status of the file. File status is indicated by the system response to the EDIT command. A response of

```
    NEW FILE:
```

indicates that a file corresponding to the entered file identifiers does not exist, therefore any further action on the user's part involves creating data for the new file. To do this, the user must enter the input mode. This is accomplished by entering the edit subcommand, INPUT. To return to edit mode, enter a null line. (A null line is defined by a carriage return (or equivalent) that is not preceded by any line entered data. For more information on the CMS EDIT environment see VM/370: EDIT Guide.

The ECHO environment is entered when the CP command ECHO is keyed in. All data lines entered in the ECHO environment are transmitted unchanged back to the terminal from which they were received.

The RSCS component differs from other VM/370 components in that one virtual machine has supervisory control of resources. Other virtual machines interface with the RSCS virtual machine via CP SPOOL and TAG commands. These commands and RSCS virtual machine commands (and control information from HASP/ASP Batch Processors, remote MULTI-LEAVING[1] programmable stations and remote nonprogrammable terminals) control the traffic and destination of spool files.

Press the Return key (or its equivalent) with no characters entered, to determine which mode you are in.

You can take various actions to pass control from one mode to another. Figures 1 and 2 indicate the various effects of an attention interruption for your virtual console.

For display terminals the Enter key serves the functions of ATTN signaling, and Return key function (command line end signal) In addition, screen control is provided by the Enter key and the Cancel key. Figure 3 shows this screen control function. For more information on screen image control, see the _VM/370_: _Terminal_ _Users_ _Guide_ and the _VM/370_ _EDIT_ _Guide_.

----

[1] IBM Unregistered Trademark

Figure 1. Attention Handling in VM Mode
(Part 1 of 2)

State	Press ATTN Key	Action
Terminal idle; keyboard entry blocked; virtual machine running	1	Attention interruption pending; virtual machine running
	>1	Keyboard activated for CP input
Terminal receiving output from virtual machine	1	Attention interruption pending; virtual machine running
	>1	Keyboard activated for CP input at completion of console I/O
Keyboard activated for input to virtual machine; no data entered or all data deleted	1	Device end (DE) and attention status pending; virtual machine running
	>1	Unit exception (UE) status pending; keyboard activated for CP input
Keyboard activated for input to virtual machine; some data entered	1	Unit exception (UE) status pending; virtual machine running
	>1	Device end (DE) status pending; keyboard activated for CP input
Keyboard entry blocked; executing CP command	1 or >1	Attention ignored

State	Press ATTN Key	Action
Keyboard entry blocked; in SLEEP mode entered via command	1 or >1	Keyboard activated for CP input
Keyboard entry blocked; in SLEEP mode entered via Diagnose instruction	1 or >1	Virtual machine resumes execution
Terminal receiving output from CP but not from user command	1	Attention interruption pending; virtual machine running
	>1	Keyboard activated for CP input
Terminal receiving output in response to CP command	1 or >1	Output line cancelled and in some cases command output cancelled
Keyboard activated for CP input; no data entered or all data cancelled	1 or >1	Attention interruption made pending; virtual machine running
Keyboard activated for CP input; some data entered	1 or >1	Input line cancelled; keyboard activated for CP input

Figure 1. Attention Handling in VM Mode (Part 2 of 2)

State	Action
Terminal idle; keyboard entry blocked; virtual machine running	Keyboard activated for CP input
Terminal receiving output from virtual machine	Keyboard activated for CP input
Keyboard activated for input to to virtual machine; no data entered or all data deleted	Unit exception (UE) status pending; keyboard activated for CP input
Keyboard activated for input to virtual machine; some data entered	Device end (DE) status pending; keyboard activated for CP input
Keyboard entry blocked; executing CP command	Attention ignored
Keyboard entry blocked; in SLEEP mode entered via command	Keyboard activated for CP input
Keyboard entry blocked; in SLEEP mode entered via Diagnose instruction	Virtual machine resumes execution

Figure 2. Attention Handling in CP Mode (Part 1 of 2)

State	Action
Terminal receiving output from CP but not from user command	Keyboard activated for CP input
Terminal receiving output in response to CP command	Output line cancelled and in some cases command output cancelled
Keyboard activated for CP input; no data entered or all data cancelled	Attention interruption made pending; virtual machine running
Keyboard activated for CP input; some data entered	Input line cancelled; keyboard activated for CP input

Figure 2. Attention Handling in CP Mode (Part 2 of 2)

Figure 3. Summary of Screen Status Action While Executing CMS and CP (Part 1 of 2)

Initial Status	Mode	Key Pressed	Data	Action	Resulting Status
RUNNING	CP	ENTER	NONE	Enters console function mode	CPREAD
			DATA	Executes console function	RUNNING
		CNCL	N/A	Clears output area	RUNNING
	VM	ENTER	NONE	"Attn" interruption pending	RUNNING[1]
			DATA	"Attn" interruption pending, stack data	RUNNING[2]
		CNCL	N/A	Clears output area	RUNNING
MORE	CP/VM	ENTER	NONE	Holds screen output	HOLDING
			DATA	"Attn" interruption pending, stack data	MORE
		CNCL	N/A	Clears output area, continues output	RUNNING
HOLDING	CP/VM	ENTER	NONE	Allows screen output to continue	MORE
			DATA	"Attn" interruption pending, stack data	HOLDING
		CNCL	N/A	Clears output area, continues output	RUNNING[3]

Initial Status	Mode	Key Pressed	Data	Action	Resulting Status
CPREAD	CP/VM	ENTER	NONE	"Null" line return	RUNNING[4]
			DATA	Executes console function	CPREAD
		CNCL	N/A	Clears output area	CPREAD
VMREAD	CP/VM	ENTER	NONE	"Null" line return	RUNNING
			DATA	Processes data	RUNNING
		CNCL	N/A	Clears output area	VMREAD
NOT ACCEPTED	A previously stacked input buffer is still pending				Returns to former status

[1] The status shown is RUNNING, however, the virtual machine should respond to the Attn with a read, whereupon the status goes to VMREAD.

[2] If a data buffer is already stacked for a virtual machine, the terminal displays NOT ACCEPTED status before returning to the RUNNING status.

[3] If you are running with TERMINAL MODE CP (the default for the primary system operator) then an attention return is also made, causing cancellation of the function. Operators at the System/370 Model 158 console use this function to terminate certain QUERY or DISPLAY functions because the System/370 Model 158 console does not have a PA1 key.

[4] Unless you are the VM/370 primary system operator or are using the SET RUN ON option, the status returns to CP READ for another console function if the previous read was for a console function.

Figure 3. Summary of Screen Status Action While Executing CMS and CP (Part 2 of 2)

USING CMS

Before you can use CMS, you must do the following:

- Log on with a valid user identification and password. The user identification should have a directory entry with the devices needed for a CMS user.

- IPL (initial program load) the CMS system by specifying the name of the CMS system or the device address of the CMS system disk.

- Have disk space available that is formatted for use by CMS.

The logging on procedure is discussed in the "VM/370 System Information" section. The IPL and disk formatting procedures are described in this section.

HOW TO IPL CMS

After you have logged on the VM/370 system, you can IPL an operating system.

Assume that CMS is the systemname of your CMS operating system and that 190 is the CMS system disk. You can IPL this CMS system with either of the following commands:

```
ipl cms
ipl 190
```

FORMATTING YOUR VIRTUAL DISK SPACE

Before you can use CMS in your virtual machine, you must have disk space that has been formatted for use by CMS. Usually, the system operator provides formatted disk space for a new user. However, you can format your own disk space. This formatting procedure is performed only when new disk space is being initialized for your virtual machine; it should not be done each time you log on to the system. Formatting a disk destroys the contents of that disk.

The disk space for userid PUBS is defined in the CP directory as virtual disk 191. This virtual disk space is the PUBS A-disk (or primary user disk). If you attempt to use CMS before formatting your A-disk, an error message is issued. For example, assume that you (with userid PUBS) have logged on and now want to IPL CMS in your virtual machine, but your A-disk (virtual address 191) was never formatted. The terminal output looks like this:

```
ipl cms
CMS..VERSION n.n mm/dd/yy

(Press the Return key.)

Y (19E) R/O.
DMSACC112S 'A (191) ' DEVICE ERROR.
R; T=0.01/0.07 11:25:17
```

The "Y (19E) R/O." message tells you that the CMS system you just loaded (via IPL) has a Y-disk at address 19E which is a read-only extension of the system disk (S-disk).

The "DEVICE ERROR" message indicates that your A-disk (in this example, 191) was not correctly formatted prior to use.

To format your disk, issue the CMS FORMAT command

```
format 191 a
```

where 191 indicates the virtual disk address, and "a" indicates that it is the A-disk. The FORMAT command then issues prompting messages to which you must reply:

```
DMSFOR603R FORMAT WILL ERASE ALL FILES
   ON DISK 'A(191)'. DO YOU WISH TO
   CONTINUE? (YES|NO):
yes
DMSFOR605R ENTER DISK LABEL:
pubs01
FORMATTING DISK 'A'.
'3' CYLINDERS FORMATTED ON 'A(191)'.
R; T=0.15/1.60 11:26:03
```

If any files existed, they are erased. The disk space, which contains three cylinders, is labeled PUBS01. When your PUBS A-disk is formatted and the CMS virtual machine is operating, you can use CMS to do some further setup work.

If you know your disk is not formatted at the time you IPL, enter the commands:

```
ipl cms
access (nodisk
```

before pressing the Return key. The error message, DMKACC112S, does not appear. You should then issue the command

```
format 191 a
```

to format your A-disk.

WRITING A PROFILE EXEC

Although you can use CMS without a PROFILE EXEC, it is often convenient to use one. The PROFILE EXEC is a special EXEC procedure that is executed as the first command after you IPL CMS. If you want to use the assembler to assemble programs under CMS, it is a good idea to include the CMS and OS macro library in your PROFILE EXEC definition. You can do this by putting the appropriate GLOBAL command in your PROFILE EXEC. Other additions for your PROFILE EXEC might be:

- The short form of the "Ready" message (R;).

- A blip character of "*" to indicate seconds of virtual CPU time.

You create your PROFILE EXEC by using the CMS EDIT command. The EDIT command is fully described in the _VM/370: EDIT Guide_.

Only the EDIT subcommands used to create your PROFILE EXEC are included here. Your PROFILE EXEC for userid PUBS may be created by issuing the EDIT command with the filename and filetype of "PROFILE EXEC". If the edit program does not find the file you specified, it then issues the message "NEW FILE:" and enters the edit mode. You should type "input". When the edit program responds with "INPUT:", you can start entering the statements of your PROFILE EXEC file. For a description of these subcommands, see the _VM/370: EDIT Guide_. The entire terminal listing would appear as follows:

```
edit profile exec
NEW FILE:
EDIT:
input
INPUT:
&control off
set rdymsg smsg
global maclib cmslib osmacro
set blip * (1)

 (Press the Return key to leave INPUT
  mode.)

EDIT:
file
R; T=0.21/0.84 11:31:37
```

Now that your PROFILE EXEC has been created and filed, you can verify that it contains the desired commands by requesting a copy of it at the terminal:

```
type profile exec

&CONTROL OFF
SET RDYMSG SMSG
GLOBAL MACLIB CMSLIB OSMACRO
SET BLIP * (1)

R; T=0.12/0.58 11:32:58
```

Note: The PROFILE EXEC does not execute immediately (the Ready message is still the long message). The PROFILE EXEC is not executed until the next time you issue IPL CMS or the next time you type "profile" during your terminal session.

For a more detailed discussion about EXEC, see the VM/370: EXEC User's Guide.

EXAMPLE OF CMS PROGRAM DEVELOPMENT FACILITIES

This section illustrates several CMS functions that are useful in creating and manipulating CMS files.

First IPL CMS. Note that if you have followed the preceding instructions, the disk space is already formatted and no error message appears. Also, the short form of the Ready message types because your PROFILE EXEC file is in effect.

CREATING AN ASSEMBLER LANGUAGE SOURCE FILE

The program shown in Figure 4 in this section is an Assembler Language program that reads data from one CMS file and writes it to another CMS file. After you have logged on the system and issued IPL CMS, you can create the program using the CMS EDIT facility.

```
edit manip assemble
NEW FILE:
EDIT:
input
INPUT:
manip     csect
          print nogen
          save  (14,12),,*
          balr  12,0
          using *,12       establish addressability
          la    2,8(,1)    r2=addr of input file in plist
          la    3,32(,1)   r3=addr of output file in plist
* determine if input file exists
          fsstate (2),error=err1
* read a record from input file and write on output file
rd        fsread (2),error=eof,buffer=buff1,bsize=80
          fswrite (3),error=err2,buffer=buff1,bsize=80
          b     rd         loop back for next record
* come here if error reading input file
eof       equ   *
          la    15,7       test code for read error
```

Figure 4. Sample Assembler Language Program Used for Creating a Source File (Part 1 of 2)

```
          c     15,=f'12'  end of file?
          bne   err3       error if not
          return (14,12),rc=0
* if input file does not exist
err1      wrterm 'file not found',edit=yes
          b     erret
* if error writing file
err2      linedit text='error code ..... in writing file', sub=(dec,(15))
          b     errt
* if reading error was not normal end of file
err3      linedit text='error code ..... in reading file', sub=(dec,(15))
erret     return (14,12),rc=1  return to caller
buff1     ds    cl80
          end   manip
(Press the RETURN key to leave Input mode.)

EDIT:
file
R;
```

Figure 4. Sample Assembler Language Program Used for Creating a Source File (Part 2 of 2)

The Editor (the term applied to the edit program that is used by the EDIT command), did not find a file with the filename and filetype of MANIP ASSEMBLE, so it created the file for you. Enter the INPUT subcommand so that you can enter your program code into the file. You must issue the FILE subcommand in order to save your program.

This program (MANIP CSECT) uses several CMS macros; when it is assembled, this program requires the CMS macro library. However, your PROFILE EXEC (for the userid PUBS) has specified that the CMS macros be included; no further action is necessary to include the CMS macros.

The Load Address (LA) instruction following EOF (end-of-file) is inserted only for testing; it is deleted after the function is tested.

ASSEMBLING A SOURCE FILE

To assemble the MANIP program, you enter the "ASSEMBLE MANIP" command, then wait for the assembler to complete processing:

```
assemble manip
*****i
 ASSEMBLER (F) DONE


 MAN00331            B     ERRT
 IEU024 NEAR OPERAND COLUMN 1-UNDEF SYMBOL

      1 STATEMENT FLAGGED IN THIS ASSEMBLY
      8 WAS HIGHEST SEVERITY CODE
R(00008);
```

Each asterisk (*) on the second line indicates two seconds of virtual CPU time.

The message IEU024 indicates an error in your program. The line in your program containing the error has a sequence number of MAN00331. Print your listing file to find this line.

At this point, three files are associated with your program. First, the file, MANIP ASSEMBLE, contains the source statements of your program. This file was the input used by the Assembler Language program. The output from the assembler is two permanent files. One of these files, MANIP TEXT, contains the object module. The other file, MANIP LISTING, contains a listing of the source statements, assembled machine code, and other associated information based on the options selected for the ASSEMBLE command.

Correcting Errors

Since the assembler has detected an error in the source code, you must correct the error before attempting to execute the program. Just as you used the Editor to create the assembler file, you also use the Editor to change or correct the assembler file. When you issue the EDIT MANIP ASSEMBLE command this time, the Editor finds your file and enters edit mode. Then issue the LOCATE subcommand to find the line in error. Issue the CHANGE subcommand to correct the error and then issue FILE to save the corrected program. The terminal output as follows:

```
edit manip assemble
EDIT:
locate /errt/
          B     ERRT
change /errt/erret/
          B     ERRET
```

```
file
R;
```

Now that the error has been corrected, you can assemble the file again:

```
assemble manip
*****i
 ASSEMBLER (F) DONE


 NO STATEMENTS FLAGGED IN THIS ASSEMBLY
R;
```

This time, the program assembled without any assembler-detected errors. The TEXT and LISTING files from the previous assembly are erased automatically and replaced by the new ones from the current assembly.

CREATING A LOAD MODULE

You can now create a load module from the TEXT file that was created by the assembler. The resulting MODULE file can then be executed.

```
load manip
R;
```

```
genmod manip
R;
```

Now, a fourth file, MANIP MODULE, exists. This file is in executable form.

Testing and Correcting a Program

Once the MODULE file has been created, you can begin testing. To execute the MANIP MODULE file, issue the MANIP command name, plus the file identifiers for the input and output files. The input file (MANIP ASSEMBLE A1) is to be copied and the resulting file is to be called MANIP1 ASSEMBLE A1. The first test should take the branch on the FSREAD error. The following error message appears on the terminal:

```
manip manip assemble a1 manip1 assemble a1
ERROR CODE 7 IN READING FILE.
R(00001);
```

You should then use the Editor to correct the program so that this branch is no longer taken.

```
edit manip assemble
EDIT:
find eof
EOF     EQU *
next
        LA 15,7 TEST CODE FOR READ ERROR
delete
file
R;
```

After the corrected version of the program is filed, assemble and execute the program again.

```
assemble manip
*****i
 ASSEMBLER (F) DONE


 NO STATEMENTS FLAGGED IN THIS ASSEMBLY
R;
```

```
load manip
R;
```

```
genmod manip
R;
```

Now that the testing statement has been deleted, and a new MODULE file created, further testing of the program can begin. First, attempt to copy a file that does not exist. The file is not found.

```
manip file1 assemble a1 file2 assemble a1
FILE NOT FOUND
R(00001);
```

Then, attempt to copy a file to itself. Your program is not equipped to do this; an error occurs.

```
manip manip assemble a1 manip assemble a1
ERROR CODE 9 IN WRITING FILE.
R(00001);
```

Finally, create a new file (MANIP1) from your MANIP file.

```
manip manip assemble a1 manip1 assemble a1
R;
```

ERASING UNWANTED FILES

Once testing is complete, display the beginning of MANIP1 to make sure that it was copied correctly, then delete the MANIP1 file:

```
type manip1 assemble 1 5

MANIP  CSECT
       PRINT NOGEN
       SAVE     (14,12),,*
       BALR     12,0
       USING *,12 ESTABLISH ADDRESSABILITY

R;

erase manip1 assemble
R;
```

The LISTFILE command can then be issued to make sure the file was erased:

```
listfile * assemble
MANIP    ASSEMBLE  A1
R;
```

PRINTING, PUNCHING, AND READING FILES

PRINTING

When you want to print your program listing, you should first check the output status of your virtual printer by entering:

```
query 00e
PRT  00E CLS A          COPY 01
R;
```

Since output class A is acceptable for program listings, print the LISTING file:

```
print manip listing
R;
```

You can also print the LISTING file by specifying the PRINT option when you issue the ASSEMBLE command. Once the LISTING file is printed, it can be erased. Also, you may want to erase the TEXT file from which the MODULE file was generated:

```
erase manip listing
R;
```

```
erase manip text
R;
```

PUNCHING

If other users want to use your MANIP program, send it to them by changing the destination of your virtual punch, then punch the MANIP TEXT file. Use the CMS COPYFILE or MOVEFILE commands to transfer the MANIP MODULE file to another user. For example, suppose the user PAYROLL wanted to use the MANIP program. You could send PAYROLL a copy of the TEXT file by entering:

```
spool 00d to payroll
R;
```

```
punch manip text
PUN FILE 029 TO  PAYROLL
R;
```

READING

When the user PAYROLL logs on the VM/370 system, the following message types during the logon procedure:

```
FILES: 001 RDR,  NO PRT,  NO PUN
```

To read in this file, the PAYROLL user must IPL CMS and issue the command:

```
read *
:READ  MANIP TEXT A1 PUBS mm/dd/yy 13:29:03
R;
```

Note, however, that the PAYROLL user can decide whether or not he wants the file before he reads it by invoking the command:

```
query reader all
FILE CLS RECDS ORIGIN DATE     TIME     NAME   TYPE
029  A   00051 PUBS   11/30/72 13:29:03 MANIP  TEXT
```

If the PAYROLL user does not want the file, he can purge it from his reader, as follows:

```
purge reader        (or purge reader 029)
0001 FILE  PURGED
```

CMS can be used for many other purposes. Those functions illustrated in the previous discussion are intended to help the new VM/370 user become acquainted with the system and its capabilities. Once you are familiar with these commands and functions, you have a sound base upon which to build a more thorough understanding of the VM/370 system.

DISK DETERMINATION (FILEMODE MANAGEMENT)

Figure 5 relates CMS commands, method of specifying filemode, and criteria used in choosing a disk directory for reading and writing.

The Filemode column indicates how to specify the filemode on the command line. The symbols used are:

Symbol	Meaning
command	CMS command name
fm	Explicit filemode letter can be specified
=	Write disk to Read disk
*	Refer to all disks in a set search order
d	Default mode: let system determine where to go
-	Null mode; unable to specify filemode letter in this command

The Reading column indicates the disks that CMS searches when looking for the file to be read. The symbols used are:

Symbol	Meaning
N/A	Not applicable, command does not cause any reading to be done
fm	Read from the specified disk
*R	Refer to all disks in the standard search order
A	Read only from the primary disk

Symbol	Meaning
1R	Read from the primary disk and its extensions
*cuu	All occurrences of the address

The Writing column indicates the disks that CMS attempts to write the file to. The symbols used are:

Symbol	Meaning
N/A	Not applicable, command does not cause any writing to be done
fm	Write onto the specified disk
R	Write onto the disk from which a file was read (or its parent)
*W	Choose any read/write disk in the standard search pattern
1W	Attempt to write onto the primary disk
cuu	Write to the specified address
*WS	First read/write disk with enough space
*1	First disk where file is found if disk is in read/write status
cw	Attempts to write on the disk from which the file was read. If the disk is read-only, no writing is done.

Command	Filemode	Reading	Writing
ACCESS	mode	fm	N/A
	d	1R	N/A
ASM3705	—	*R	R,1W,*W
ASSEMBLE	—	*R	R,1W,*W
COBOL[1]	—	*R	R,1W,*W
COMPARE	fm	fm	N/A
	*	*R	N/A
CONVERT[1]	fm	fm	fm
COPYFILE	fm	fm	fm
	=	N/A	R
	*	*R	CW
CP	—	N/A	N/A
DEBUG	—	N/A	N/A
DIRECT			
DISK DUMP	fm	fm	N/A
	*	*R	N/A
	d	A	N/A
DISK LOAD	—	N/A	1W
EDIT	fm	fm	fm
	*	*R	R
	d	1R	R

[1] IBM Program Product

Figure 5. Disk Determination (Part 1 of 5)

Command	Filemode	Reading	Writing
ERASE	fm	N/A	fm
	*	N/A	*W
	d	N/A	1W
EXEC	-	*R	N/A
FILEDEF	fm	fm	fm
	d	1R	1W
	*	*R	N/A
FORMAT	mode	N/A	fm
FORTGI[1]	-	*R	R,*W
FORTHX[1]	-	*R	R,*W
GENDIRT	-	N/A	N/A
GENMOD	fm	N/A	fm
	*	N/A	1W
	d	N/A	1W
GEN3705	-	*R	1W
GLOBAL	-	N/A	N/A
GOFORT[1]	-	*R	R,*W
INCLUDE	-	*R	1W
LISTDS			
LISTFILE	fm	fm	1W
	*	*R	1W
	d	A	1W
LKED	-	*R	R,1W,*W
LOAD	-	*R	1W

[1] IBM Program Product

Figure 5. Disk Determination (Part 2 of 5)

Command	Filemode	Reading	Writing
LOADMOD	–	*R	N/A
	fm	fm	N/A
MACLIB	–	*R	R,1W
MODMAP	–	*R	N/A
MOVEFILE	–	N/A	N/A
NCPDUMP	–	*R	1W
PLIC[1]	–	*R	R,1W,*W
PLICR[1]	–	*R	R,1W,*W
PLIOPT[1]	–	*R	R,1W,*W
PRINT	fm	fm	N/A
	d	1R	N/A
	*	*R	N/A
PUNCH	fm	fm	N/A
	d	1R	N/A
	*	*R	N/A
QUERY	–	N/A	N/A
READCARD	fm	N/A	fm
	d	N/A	1W
	*	N/A	1W
RELEASE	–	*cuu	cuu
	mode	fm	fm
RENAME	fm	fm	fm
	*	*R	N/A
	=	N/A	R

[1] IBM Program Product

Figure 5. Disk Determination (Part 3 of 5)

Command	Filemode	Reading	Writing
RUN	fm	fm	N/A
	*	*R	N/A
	d	1R	N/A
SAVENCP	–	*R	N/A
SCRIPT[2]	–	*R	1W
SET	–	N/A	N/A
SORT	fm	fm	fm
	*	*R	R,1W
START	–	N/A	1W
STATE	fm	N/A	N/A
	*	N/A	N/A
	d	N/A	N/A
SVCTRACE	–	N/A	N/A
SYNONYM	fm	fm	N/A
	*	*R	N/A
	d	1R	N/A
TAPE DUMP	fm	fm	N/A
	*	*R	N/A
	d	1R	N/A
TAPE LOAD	fm	N/A	fm
	d	N/A	1W
TAPE SCAN	–	N/A	N/A
TAPE SKIP	–	N/A	N/A
TAPPDS	fm	N/A	fm
	d	N/A	1W

[1] IBM Program Product

[2] IBM User Installed Program (IUP)

Figure 5. Disk Determination (Part 4 of 5)

Command	Filemode	Reading	Writing
TESTCOB[1]	—	*R	R, 1W, *W
TESTFORT[1]	—	*R	*1, *WS
TXTLIB	—	*R	R, 1W
TYPE	fm	fm	N/A
	*	*R	N/A
	d	1R	N/A
UPDATE	—	*R	R, 1W
	fm	fm	fm, R, 1W
	d	1R	1W
	*	*R	R, 1W
VSBASIC[1]	—	*R	R, 1W
ZAP	—	*R	R

[1] IBM Program Product

Figure 5. Disk Determination (Part 5 of 5)

RESERVED FILETYPE DESCRIPTIONS

Figure 6 shows filetypes that have special uses in CMS.

Filetype	Command	Usage	Filename
ASSEMBLE	ASSEMBLE	Input	fn
ASM3705	ASM3705	Input	fn
	GEN3705	Output	fn(nn)
AUXxxxx	UPDATE	Input	fn
BASDATA	BASIC execution	Execution time files	fn
BASIC	BASIC	Input	fn
CMSUT1	READCARD COPYFILE DISK TAPE UPDATE INCLUDE LOAD MACLIB	Intermediate work file	READCARD COPYFILE DISK TAPE fn DMSLDR DMSLDR DMSLBM
CNTRL	UPDATE	Input	fn
COBOL	COBOL[1]	Input	fn
COPY	MACLIB	Input	fn

[1] IBM Program Product

Figure 6. Reserved Filetypes (Part 1 of 4)

Format		
RECFM	LRECL	Contents
F	80	Assembler Language source statements
F F	80 80	3704/3705 assembler source statements
F	80	Auxiliary update file
U	≤3440	User input and output files
F	≤256	BASIC language source statements
F	80	
F	80	Control file update
F	80	COBOL source statements
F	80	COPY control cards and macro definitions

Filetype	Command	Usage	Filename
DIRECT	DIRECT	Input	fn
EXEC	EXEC	Input	fn
	LISTFILE	Output	CMS
	GEN3705	Output	fn
FREEFORT	GOFORT[1]	Input	fn
FORTRAN	FORTGI[1]	Input	fn
	FORTHX[1]		
	GOFORT[1]		
	TESTFORT[1]		
FTnnF001	FORTRAN execution	Input/Output	fn
LISTING	ASSEMBLE	Output	fn
	ASM3705	Output	fn
	GOFORT[1]		
	FORTGI[1]		
	FORTHX[1]		
	COBOL[1]	Output	fn
	PLIC[1]		
	PLICR[1]		
	PLIOPT[1]	Output	fn
	TESTCOB[1]	Input	fn
LKEDIT	LKED	Output	fn
LOADLIB	LKED	Output	fn
	ZAP	Input	fn
MACLIB	GLOBAL	Library	fn
	MACLIB	Input/Output	fn

[1] IBM Program Product

Figure 6. Reserved Filetypes (Part 2 of 4)

Format		Contents
RECFM	LRECL	
F	80	User Directory entries
F	80	EXEC statements
V	≤81	FREEFORM FORTRAN source statements
F	80	FORTRAN source statements
		User input and output files
F	121	Processor printed output
F	121	COBOL processor output used as input to SOURCE subcommand of TESTCOB
F	121	Listing
V	≤260	3704/3705 control program load modules
Library contains dictionary and members		Macro definitions Macro definitions

Filetype	Command	Usage	Filename
MACRO	MACLIB	Input	fn
MAP	INCLUDE	Output	LOAD
	LOAD	Output	LOAD
	MACLIB	Output	fn
	TXTLIB	Output	fn
MEMO			
MODULE	GENMOD	Output	fn
	LOADMOD	Input	fn
	MODMAP	Input	fn
PLI or	PLIOPT[1]	Input	fn
PLIOPT	PLIC	Input	fn
	PLICR	Input	fn
SCRIPT	SCRIPT[2]	Input	fn
SYNONYM	SYNONYM	Reference	fn
SYSUT1	ASM3705	Work	fn
SYSUT2	ASSEMBLE	Work	fn
SYSUT3	COBOL[1]	Work	fn
	LKED	Work	fn
	PLIOPT[1]	Work	fn
SUSUT4	COBOL[1]	Work	fn
	LKED		
	PLIC		
	PLICR		
	TESTCOB[1]	Input	
TESTFORT	TESTFORT[1]	Output	fn

[1] IBM Program Product

[2] IBM User Installed Program (IUP)

Figure 6. Reserved Filetypes (Part 3 of 4)

Format		
RECFM	LRECL	Contents
F	80	Macro definitions
		Module map
		Module map
		Library map
		Library map
F	80	
V		Nonrelocatable object file
F		PL/I source statements
V	≤133	Input to SCRIPT processor
F	80	Command name synonyms
F	80	
	512	Used as input to TESTCOB
VB	125	Processor printed output

Filetype	Command	Usage	Filename
TEXT	ASSEMBLE	Output	fn
	ASM3705	Output	fn
	COBOL[1]	Output	fn
	GEN3705	Output	fn(Ln)
	INCLUDE	Input	fn
	LKED	Input	fn
	LOAD	Input	fn
	PLIOPT[1]	Output	fn
	TXTLIB	Input	fn
	GOFORT[1]	Output	fn
	FORTGI[1]	Output	
	FORTHX[1]	Output	
	TEXTFORT[1]	Input	
TXTLIB	GLOBAL	Library	fn
	TXTLIB	Output	fn
UPDATE	UPDATE	Input	fn
UPDLOG	UPDATE	Output	fn
VSBASIC	VSBASIC[1]	Input	fn
VSBDATA	VSBDATA	Execution time files	fn

[1] IBM Program Products

Figure 6. Reserved Filetypes (Part 4 of 4)

Format		
RECFM	LRECL	Contents
F	80	Object code
F	80	3704/3705 source code and job control language statements
		Object code
F	80	Linkage editor control statements for 3704/3705 control programs
		Object code
		Object code and LKED control cards
		Object code
		Object code
		Object code
		Object file
Library contains dictionary and members		Object decks
F	80	UPDATE control cards
F		UPDATE log
F	≤256	VSBASIC language source statements
V	≤140	VSBASIC user input/output files

CMS RETURN CODES

If a condition arises during execution of a CMS command that results in the display of a Warning, Error, Severe Error, or Terminal Error message, the CMS command passes a nonzero return code to register 15. CMS return codes (RC) are assigned as follows:

RC	Meaning
4	The user did not specify all the conditions necessary to execute the command as intended. Execution of the command continues, however the result may or may not be as the user intended.
8	Device errors occurred for which a Warning message is issued, or Errors have been introduced into the output file.
12	Errors have been found in the input file.
20	An invalid character is in the fileid. Valid characters are: 0-9, A-Z, *, @, #, a-z.
24	The user did not specify the command line correctly.
28	Error occurred while trying to access, or manipulate, a user's files; for example, file not found.
32	The user's file is not in the expected format, or The user's file does not contain the expected information.
36	Error occurred to the user's devices for which he is responsible. For example, a disk is in read-only status, and needs to be in write

status in order to write out a file.

40 A functional error occurred during execution of the command for which the user is responsible, or
The user failed to supply all the necessary conditions for executing the command, or
End-of-file, end-of-tape (where applicable).

88 A CMS system restriction prevented execution of the command, or
The function requested is an unsupported feature, or
The device requested is an unsupported device.

100 Input/output device errors.

104 A functional error occurred during execution of the command for which the system is responsible.

256 All unexpected errors for which the system is responsible; that is, Terminal Error messages.

If no Warning, Error, Severe Error, or Terminal Error messages are generated during execution of the command, the return code passed to register 15 is zero.

Commands that invoke Program Products pass the return code set by the program in register 15 to the user. This code may have the same number as a CMS code described above; however, it will have been redefined by the Program Product or compiler in operation.

Return Codes Produced by the CMS DIRECT Command

RC	Meaning
1	Invalid filename, or file not found.
2	Error loading the directory.
3	Invalid option from CMS.
4	Directory not swapped, user not privilege class A, B, or C.
5	Directory not swapped, system (old) directory locked.
6	Directory not swapped, the directory in use by the system is not the updated directory.
1xx	Error in the CMS RDBUF routine.
2xx	Error in the CMS TYPLIN routine.

where:

xx is the CMS routine return code.

Return Codes Produced by the CMS DDR Command

RC	Meaning
1	Invalid filename, or file not found.
2	Error in executing the program.
3	Flagged DASD (Direct Access Storage Device) track.
4	Permanent tape or DASD I/O error.

RC	Meaning
1xx	Error in the PRINTIO routine.
2xx	Error in the CONREAD routine.
3xx	Error in the RDBUF routine.
4xx	Error in the TYPLIN routine.

where:

xx is the CMS routine return code.

Example of a Return Code from a CP Command

Commands or functions of commands passed to CP pass the return code sent back by CP to register 15. For example, if the user is in CMS mode and invokes the CP command LINK:

```
ipl cms
CMS VERSION 1.0 mm/dd/yy
----------------------------------
----------------------------------
cp link to * vaddr1 as vaddr2 r
```

The user has entered the CP command LINK to userid *. That means the user's own directory is searched for device vaddr1. Vaddr2 is the virtual address to be assigned to the device for this virtual machine. Read-only access is requested. No password is required because the user has linked to one of his own disks.

The result may be

R;   which indicates successful execution.
R(nnnnn);   which indicates an error.

If nnnnn contains a CMS return code (as described previously), the error occurred in CMS.

If nnnnn contains a CP message identification (see "CP Error Message Numbers,"), the error occurred in CP.

The return code may be used by a systems programmer in the DEBUG subcommand and also in EXEC procedures. See the *VM/370: EXEC User's Guide* for a description of the &RETCODE special variable.

In this publication, the terms "return code" and "completion code" are synonymous.

SUMMARY OF VM/370 COMMANDS AND SERVICE AIDS

The following list contains all the CP, RSCS, and CMS commands and the VM/370 service aids; a brief description precedes a syntactic representation of each command. The commands and subcommands are shown in uppercase and lowercase; the uppercase represents the minimum truncation of the command or keyword operand that the system accepts. An all lowercase operand indicates a user or system supplied variable value. Examples: raddr (real address)=00E, fn (filename)=HISTORY1. Where operands are between braces ({ }) only one must be selected. Where operands are between brackets ([ ]) only one or none can be selected. The underscored operand (_) between brackets is the system selected default value if another operand is not selected. In this text a vertical bar (|) indicates the separation of operands between brackets and braces. See the *VM/370: Command Language Guide for General Users*, for an explanation of other notational conventions.

Description	Format
* CP Class Any Permits comments	* any comment
* CMS Permits comments	* any comment
#CP Any Executes a CP command	#CP [commandline1 [#commandline2# ...]]
ACCESS CMS Defines direct access space to a CMS virtual machine and relates it to a logical directory.	ACcess [cuu mode [/ext [fn [ft [fm]]]]] [(options...[)]] options: [NOPROF|ERASE][NODISK]
ACNT CP Class A Creates accounting records.	ACNT { ALL|userid1 userid2 . . .}
ADSTOP CP Class G Halts the virtual machine's execution.	ADSTOP { hexloc|OFF }

```
|ASM3705                       CMS |ASM3705 fn [(options...[)]]                                     |
|  Invokes 3705 assembler.          |                                                                |
|                                   | options:                                                       |
|                                   |   [XREF|NOXREF]      [LOAD|NOLOAD]                             |
|                                   |   [RENT|NORENT]      [LIST|NOLIST]                             |
|                                   |   [DECK|NODECK]      [PRINT|DISK|NOPRINT]                      |
|                                   |   [LINECNT 55|LINECNT nn]                                      |
|                                   |                                                                |
|ASSEMBLE                      CMS |Assemble  fn  [(options...[)]]                                  |
|  Invokes the system assembler.    |                                                                |
|                                   |Listing control options:                                        |
|                                   |[ALOGIC|NOALOGIC]        [MLOGIC|NOMLOGIC  ]                    |
|                                   |[ESD|NOESD]              [RLD|NORLD        ]                    |
|                                   |[LIST|NOLIST]            [LIBMAC|NOLIBMAC  ]                    |
|                                   |[MCALL|NOMCALL]          [FLAG (0)|FLAG nnn]                    |
|                                   |[LINECOUN (55)|LINECOUN (nn)]                                   |
|                                   |[DISK|PRINT|NOPRINT]                                            |
|                                   |[XREF (FULL)|XREF (SHORT)|NOXREF]                               |
|                                   |                                                                |
|                                   |output control options:                                         |
|                                   |[DECK|NODECK]            [OBJECT|NOOBJECT]                      |
|                                   |[TEST|NOTEST]                                                   |
|                                   |                                                                |
|                                   |SYSTERM options:                                                |
|                                   |[NUMBER|NONUM]           [STMT|NOSTMT]                          |
|                                   |[TERMINAL|NOTERM]                                               |
|                                   |                                                                |
|                                   |other options:                                                  |
|                                   |[ALIGN|NOALIGN]          [BUFSIZE (STD)|BUFSIZE (MIN)]          |
|                                   |[RENT|NORENT]  [SYSPARM (string)|SYSPARM (?)|SYSPARM ()]        |
```

Description	Format
ATTACH CP Class B Attaches a real device to a specified user or to the system.	ATTach raddr [To] {userid [As] vaddr [R[/O]] SYSTEM [As] volid}
ATTACH CHANNEL CP Class B Attaches a channel to a designated user.	ATTach CHANnel c [To] [userid|*]
ATTN CP Class G Makes attention interruption pending.	ATTN
BACKSPAC CP Class D Restarts a current spool file.	Printer Format: BAckspac raddr [File pages 1] Punch Format: BAckspac raddr [File]
BACKSPAC RSCS Restarts spool file processing at the beginning or at the back-spaced point.	BAckspac [linkid][FILE|nnn]
BEGIN CP Class G Starts the execution of a virtual machine.	Begin [hexloc]

```
|CHANGE                  CP CLASS D |CHange  ┌      ┐ ⎛                                                 ⎞
|  Alters the attributes of a closed|        |userid| ⎜ Reader    ⎧CLass c1⎫ CLass c2                    ⎟
|  spool file.                      |        |SYSTEM| ⎜           ⎨spoolid ⎬                             ⎟
|                                   |        └      ┘ ⎜           ⎩ALL     ⎭                             ⎟ ┌                  ┐
|                                   |                 ⎜                                                 ⎟ |NAme⎧fn [ft]⎫ |
|                                   |                 ⎜                                                 ⎟ |    ⎩ dsname⎭ |
|                                   |                 ⎜                                                 ⎟ └                  ┘
|                                   |                 ⎜⎧Printer⎫⎧CLass c1⎫⎛CLass c2 ⎞                   ⎟
|                                   |                 ⎨⎩PUnch  ⎭⎨ spoolid⎬⎜COpy nn  ⎟                   ⎬
|                                   |                 ⎜          ⎩ALL     ⎭⎜┌      ┐ ⎟                   ⎟
|                                   |                 ⎜                    ⎜|HOld  | ⎟                   ⎟
|                                   |                 ⎜                    ⎜|NOHold| ⎟                   ⎟
|                                   |                 ⎜                    ⎜└      ┘ ⎟                   ⎟
|                                   |                 ⎜                    ⎜DIst dist⎟                   ⎟
|                                   |                 ⎜                    ⎜┌     ┐  ⎟                   ⎟
|                                   |                 ⎜                    ⎜|SYS  |  ⎟                   ⎟
|                                   |                 ⎜                    ⎜|NOSYS|  ⎟                   ⎟
|                                   |                 ⎝                    ⎝└     ┘  ⎠                   ⎠
|CHANGE                  CP CLASS G |CHange ⎛ Reader ⎧CLass c1⎫⎛CLass c2 ⎞⎞ ┌                  ┐
|  Alters the attributes of a closed|       ⎜ Printer⎨spoolid ⎬⎜COpy nn  ⎟⎟ |NAme⎧fn [ft]⎫  |
|  spool file.                      |       ⎜ PUnch  ⎩ALL     ⎭⎜┌      ┐ ⎟⎟ |    ⎩ dsname⎭  |
|                                   |       ⎨                   ⎨|HOld  | ⎬⎬ └                  ┘
|                                   |       ⎜                   ⎜|NOHold| ⎟⎟
|                                   |       ⎜                   ⎜└      ┘ ⎟⎟
|                                   |       ⎝                   ⎝DIst dist⎠⎠
```

Description	Format
CHANGE RSCS Alters the attributes of a closed spool file.	CHange [linkid] spoolid {PRIority nn CLass c COpy nn HOld NOHold DIst distcode Name [fn [ft] | dsname]}
CLOSE CP Class G Terminates spooling operations on a virtual reader, printer, or punch.	CLose {Reader [HOld|NOHold] vaddr Printer PUnch vaddr CONsole [PUrge | [HOld | NOHold] [DIst distcode] [NAme {fn [ft] | dsame}]]}
CMD RSCS Controls the logging of I/O activity on a specified RSCS link or passes control information to remote batch processor.	CMS linkid {text|LOG|NOLOG}
CMSBATCH CMS Invokes the CMS Batch Facility, creating a virtual machine running in batch mode.	CMSBATCH [sysname]

```
COMPARE                            CMS  COMpare  fileid1 fileid2 [(COL mm-nn[) ]]
  Compares all or part of records
  in two existing files.

COPYFILE                           CMS  COPYfile  fileid1 fileid2...][fileido] [(options...[) ]]
  Copies files according to
  operand specifications.               options:
                                        [Type  ] [OLDDate] [RECfm F] [NOPRompt] [TRAns]
                                        [NOType] [NEWDate] [RECfm V] [PRompt   ]

                                        ┌       ┐ ┌                ┐ ┌                ┐
                                        |UPcase | |FRom recno      | |FOR recno       |
                                        |LOwcase| |FRLabel xxxxxxxx| |TOLabel xxxxxxxx|
                                        └       ┘ └                ┘ └                ┘

                                        ┌       ┐ ┌       ┐            ┌      ┐
                                        |REPlace| |Fill c | [TRUnc  ]  |PAck  | [EBcdic]
                                        |OVly   | |Fill hh| [NOTRunc]  |UNPack|
                                        |APpend | |Fill 40|            └      ┘
                                        |NEWFile| └       ┘             ┌       ┐
                                        └       ┘           [LRecl nn] |SPecs  |
                                                                       |NOSPecs|
                                                                       └       ┘

COUPLE                     CP Class G
  Connects virtual channel to           COUPLE  vaddr1 [To] userid vaddr2
  channel adapters.

CP                       CP Class Any   CP [commandline]
  Permits execution of CP commands
  within your privilege class.
```

Description	Format
CP CMS Permits entry of CP console functions from the CMS environment.	CP [commandline]
CPEREP CP Class C, E, and F Invokes CPEREP service aid to process VM/370 error recordings.	┌ ┐ ┌ ┐ CPEREP │IO │ [HIST] [TAPIN] │CLEARALL│ │MC │ │CLEARIO │ │ALL│ │CLEARMC │ └ ┘ └ ┘ [HELP]
DCP CP Class E Displays real processor storage on the terminal.	┌ ┐┌┌ ┐ ┐ DCP │Lhexloc1││{-}│ hexloc2 │ │ │Thexloc1││{:}│ END │ │ │ hexloc1││ └ ┘ │ │ 0 ││ ┌ ┐ │ └ ┘│{.}│bytecount│ │ │ │ END │ │ └ └ ┘ ┘

DDR CMS

Dumps, restores, or copies data between DASD devices and tape devices.

```
DDR  [fn [ft [fm|*]]]

I/O Definition Statements:
                          ┌      ┐
{ INput  }  cuu type      |volser| [ (options...[) ]]
{ OUTput }                |altape|
                          └      ┘
options:
┌         ┐ ┌       ┐ ┌          ┐
|SKip nn  | |REWind | |MOde 6250 |
|MOde den | |LEave  | |MOde 1600 |
└         ┘ |UNload | |MOde  800 |
            └       ┘ └          ┘
SYSPRINT Control Statement:
SYsprint cuu

Function Control Statements:
             ┌                                             ┐
{ DUmp     } |cyl1 [To] [cyl2[Reorder] [To] [cyl3]]        |
{ COpy     } |CPvol                                        |
{ REstore  } |NUcleus                                      |
             |ALL                                          |
             └                                             ┘
TYPE and PRINT Function Statements:
                  ┌    ┌    ┐┐
{ PRint } cyl1    |hh1 |rr1 ||[To cyl2 [hh2[rr2]]][(options...[) ]]
{ TYpe  }         | 0  | 0  ||
                  └    └    ┘┘
options:
[Hex] [Graphic] [Count]
```

Description	Format
DEBUG CMS	DEBUG
Enters the DEBUG environment to perform program analysis and repair.	DEBUG environment entered; the formats of each DEBUG subcommand are as follows:
Stops program execution.	BReak id {symbol|hexloc}
Types the Channel Address Word (CAW).	CAW
Types the Channel Status Word (CSW).	CSW
Assigns a symbolic name to a specific storage address.	DEFine symbol hexloc [bytecount 4]
Dumps the contents of storage locations to the virtual printer.	DUmp [symbol1 hexloc1 0] [symbol2 hexloc2 * 32] [ident]
Exits from the DEBUG environment.	GO [symbol hexloc]

Types the contents of the specified general registers.	GPR reg1 [reg2]
Returns to CMS environment.	HX
Sets a base address.	ORigin [symbol|hexloc]
Displays old PSW.	PSW
Returns to CMS environment.	RETurn

Changes the contents of the specified register or location.	SET { CAW hexinfo
	CSW hexinfo [hexinfo]
	PSW hexinfo [hexinfo]
	GPR reg hexinfo [hexinfo] }

Stores information in the specified virtual location.	STore { symbol hexloc } hexinfo
Examines virtual storage locations.	X { symbol } [n|length] { hexloc } [n|4]

Description	Format
DEFINE CP Class G Reconfigures the user's virtual machine.	DEFine { Reader Printer PUnch [As] vaddr CONsole CTCa TIMer 1403 3211 CHANnels[As] {SEL | BMX} LIne [As] vaddr [TEL[E2] | IBM[1]] vaddr1 [As] vaddr2 GRAF cuu [3270] [3158] T2314 | T2318 | T3330 | T3340 | T2305 [As] vaddr [CYL] nnn STORage [As] {nnnnK | nnM} }

DEFINE RSCS Adds a link and its attributes to the existing link table.	DEFine linkid {CLass c|KEEP holdslot|LINE vaddr TASK name|TYPE driverid}
DELETE RSCS Undefines a previously defined RSCS link.	DELete linkid
DETACH CP Class B Removes a real device from the system or from a specific user.	DETach raddr [From] {userid SYSTEM}
DETACH CP Class G Removes a virtual device or channel from the virtual machine.	DETach {vaddr CHANnel c}
DETACH CHANNEL CP Class B Removes the specified channel and all its related devices from the specified user.	DETach CHANnel c [From] userid
DIAL CP Class Any Attaches a terminal device to a multiple access system.	Dial userid [vaddr]

<table>
<tr><th>Description</th><th>Format</th></tr>
<tr><td><u>DIRECT</u> CMS
Allows creation, editing and swapping of VM/370 user directory.</td><td>DIRECT [fn [ft [fm|*]]] [(EDIT[ )]]

<u>Control Statements</u>:
Account number [distribution]
Console cuu devtype [class]
Dedicate cuu {rdev|[VOLID] volser}[R/O]
DIRectory cuu devtype volser
Ipl iplsys
Link userid ldev [cuu 2 [mode]]
Mdisk cuu devtype {cylr|T-DISK} cyls
volser [mode [pr [pw [pm]]]]}
Option Realtimer Ecmode Virt=real Acct Svcoff BMX
SPEcial cuu devtype [IBM|Tele]
Spool cuu devtype [class]
User userid pass [stor [mstor [cl [pri
┌ ┌ ┌ ┌ ┐┐┐┐
│le │ld │cd │es ││││]]]]
│ON │ON │ON │ON ││││
│OFF │OFF │OFF │OFF││││
└ └ └ └ ┘┘┘┘</td></tr>
</table>

DISABLE CP Classes A,B Inhibits the use of communication lines.	DISAble { raddr... ALL }
DISCONN CP Class Any Disconnects the terminal from virtual machine operation.	DISConn [HOld]
DISCONN RSCS Disconnects the RSCS operator's terminal from the RSCS virtual machine.	DISConn [userid]
DISK CMS Dumps and restores disk files.	DISK { DUMP fn ft [fm] LOAD }

```
┌───────────────────────────────────────┬──────────────────────────────────────────────────┐
|          Description                  |                     Format                       |
|───────────────────────────────────────|──────────────────────────────────────────────────|
|                                       |                                                  |
|DISPLAY                   CP Class G   |Display / ┌        ┐ ┌┌   ┌          ┐┐ \          |
|  Displays storage locations and       |        | | hexloc1| |{-}|hexloc2   || |          |
|  registers within the virtual         |        | |Khexloc1| |{:}|END       || |          |
|  machine.                             |        | |Lhexloc1| |   └          ┘| |          |
|                                       |        | |Thexloc1| |   ┌          ┐| |          |
|                                       |        | |   0    | |{.}|bytecount || |          |
|                                       |        | └        ┘ |   |END       || |          |
|                                       |        |            └   └          ┘┘ |          |
|                                       |        | Greg1 ┌ {-}┌reg2┐       ┐    |          |
|                                       |        < Yreg1 | {:}|END |       |    >          |
|                                       |        | Xreg1 |    └    ┘       |    |          |
|                                       |        |       |    ┌        ┐   |    |          |
|                                       |        |       | {.}|regcount|   |    |          |
|                                       |        |       |    |END     |   |    |          |
|                                       |        |       └    └        ┘   ┘    |          |
|                                       |        | Psw                          |          |
|                                       |        | CAW                          |          |
|                                       |        \ CSW                          /          |
|                                       |                                                  |
└───────────────────────────────────────┴──────────────────────────────────────────────────┘
```

```
┌──────────────────────────────────────────────────────────────────────────────────────────────────┐
|                                    |                                                             |
|DMCP                     CP Class E |DMCP  ┌          ┐ ┌     ┌            ┐┐[*dumpid]           |
|  Dumps any area of System/370 real|      |Lhexloc1| |{:}  |  hexloc2   ||                     |
|  storage to a spool device.        |      |Thexloc1| |{-}  |  END       ||                     |
|                                    |      | hexloc1| |     └            ┘|                     |
|                                    |      |   0    | |                   |                     |
|                                    |      └        ┘ |                   |                     |
|                                    |                 |     ┌          ┐  |                     |
|                                    |                 | {.} |bytecount|  |                     |
|                                    |                 |     |  END     |  |                     |
|                                    |                 └     └          ┘  ┘                     |
|                                    |                                                             |
|DRAIN                    CP Class D |DRain   ┌          ┐                                      |
|  Stops spooling activity on the    |        |Reader    |                                      |
|  specific device after the         |        |Printer   |                                      |
|  current file is finished          |        |PUnch     |                                      |
|  spooling.                         |        |raddr...  |                                      |
|                                    |        |ALL       |                                      |
|                                    |        └          ┘                                      |
|DRAIN                          RSCS |DRain [linkid]                                             |
|  Deactivates the specified link    |                                                             |
|  after the current file process is|                                                             |
|  completed.                        |                                                             |
|                                    |                                                             |
|DUMP                     CP Class G |DUMP ( ┌         ┐┌    ┌           ┐┐ )                    |
|  Dumps virtual machine registers   |      ( |Lhexloc1||{-}|hexloc2     || )                    |
|  and storage to the virtual        |      ( |Thexloc1||{:}|END         || )[*dumpid]           |
|  printer.                          |      ( | hexloc1||   └           ┘| )                    |
|                                    |      ( |   0     ||   ┌          ┐ | )                    |
|                                    |      ( └         ┘|{.}|bytecount| | )                    |
|                                    |      (           |   |END       | | )                    |
|                                    |                  └   └          ┘ ┘                      |
└──────────────────────────────────────────────────────────────────────────────────────────────────┘
```

Description	Format
ECHO CP Class G Returns data directly to the terminal.	ECho [nn 1]
EDIT CMS Provides access to the EDIT environment.	Edit fn ft [fm] [(options...[)]] options: [LRECL nnn][NODISP] The EDIT subcommands are as follows:
Scans records, altering the specified character.	ALter {parm1} {parm2} [1|n|*[G|*]]
Save file after indicated number of lines.	AUTOsave {n|OFF}
Reposition current line pointer backward.	BAckward [1|n]
Points to the last line of the file.	Bottom
Translates to uppercase.	CASE [U|M]

Changes string1 to string2.	Change /string1/string2 [/ [n|*|1 [G|*]]]
Enters CMS subset command mode.	CMS
Deletes n lines or to EOF.	DELete [n|1|*]
Points to the nth line down from the current line.	DOwn [n|1]
Deletes lines from the current line to (but not including) the line that contains the designated string.	DString /string[/]
Saves the file edited on disk and returns to CMS.	FILE fn [ft [fm]]
Searches the file for the given line.	Find [line]

Description	Format
EDIT (cont.)	
Resets or types the filemode.	FMode [fm]
Resets or types the filename.	FName [fn]
Changes the mode of displaying data on a 3270 terminal from typewriter style to display style or vice versa.	FORMat {DISPLAY|LINE}
Reposition current line pointer forward.	FOrward [1|n]
Inserts some or all of the given file.	Getfile fn [ft|* [fm|* [m|1 [n|*]]]]
Expands text into line images or displays current settings.	IMAGE {ON|OFF|CANON}
Inserts 'line' in the file or enters input mode.	Input [line]
Sets or types line numbering.	LINEmode [Left | Right | OFF]
Scans the file for the first occurrence of 'string'.	Locate /string[/]

Enters LONG error message mode.	LONG
Points to the nth line down from the current line.	Next [n|1]
Replaces all or part of the current line.	Overlay line
Saves current mode settings.	PREserve
Sets the line increment.	PROMPT [incr]
Terminates the EDIT session.	QUIT
Sets or displays record format.	RECfm [F|V]
Recomputes line numbers.	RENum [strtno|10 [incrno|strtno]]
Executes the following OVERLAY request n times.	REPEAT [n|1|*]
Replaces the current line with 'line' or deletes the line and enters input mode.	Replace [line]

Description	Format
EDIT (cont.)	
Restores mode settings.	REStore
Returns to EDIT environment.	RETURN
Stacks (LIFO) the last EDIT subcommand.	{REUSE} [edit subcommand] { = }
Saves the file on disk.	SAVE [fn [ft [fm]]]
Displays a number of lines above or below the current line.	S[croll][Up] [*|n|1]
Turns serialization on or off in columns 73-80.	SERial {OFF|ON|ALL|seq} [incr|10]}
Enters SHORT error message mode	SHORT
Stacks n lines in the terminal input buffer.	STACK [n|1|edit subcommand]
Sets the given tabs	TABSet {n1 n2 ...nx}
Points to the beginning of the file.	TOP

<table>
<tr><td>Sets or displays the column of truncation.</td><td>TRUNC [n|*]</td></tr>
<tr><td>Types the specified number of lines beginning with the current line.</td><td>Type [1 |m|* [n|*]]</td></tr>
<tr><td>Points to the line n lines above the current line.</td><td>Up [n|1]</td></tr>
<tr><td>Sets, displays, or resets verify mode.</td><td>Verify [ON|OFF] [[startcol|1]endcol|*]</td></tr>
<tr><td>Assigns to X or Y the given EDIT subcommand.</td><td>{X|Y} [edit subcommand|n|1]</td></tr>
<tr><td>Sets or displays the columns to be edited.</td><td>Zone [m|*|1[n|*]]</td></tr>
<tr><td>Types the last EDIT subcommand.</td><td>?</td></tr>
<tr><td>Locates the line.</td><td>nnnnn [text]</td></tr>
<tr><td>Duplicates the current line.</td><td>$DUP [1|n]</td></tr>
<tr><td>Moves n lines up or m lines down.</td><td>$MOVE n {Up m|Down m|To label}</td></tr>
</table>

Description	Format

ENABLE CP Classes A,B
Activates communication lines.
	ENable { raddr... }
	{ ALL }

ERASE CMS Deletes files from a user's disk	ERASE { fn ft [fm] } [(options...[)]] { * * } Options: [Type|Notype]
EXEC CMS Invokes EXEC files.	EXec fn [args...] The formats of the EXEC control statements and built-in functions are as follows:
Assigns variable.	&variable = ae
Defines or redefines arguments.	&ARGS [arg1 [arg2 ...]]
Punches the following lines into cards.	&BEGPUNCH [ALL]
Stacks the following lines into the terminal input buffer.	&BEGSTACK [LIFO|FIFO] [ALL]
Types the following lines at the terminal.	&BEGTYPE [ALL]

Combines token1 and token2.	&CONCAT tok1 [tok2 ...]
Provides a branching address for EXEC branch statements.	&CONTINUE
Supplies the parameters for the execution phase of the EXEC file.	&CONTROL [OFF | ERROR | ALL | CMS] [TIME | NOTIME] [NOPACK | PACK] [NOMSG | MSG]
Allows the defined token to be known from this point on by its composition, that is, numeric or character data.	&DATATYPE tok
END statement for action started by &BEGPUNCH, &BEGSTACK or &BEGTYPE.	&END
Provides error return processing.	&ERROR action
Exits from the EXEC file with a given return code.	&EXIT [returncode|0]

Description	Format
EXEC (cont.)	
Transfers control to a defined location.	&GOTO {TOP|linenumber|label}

Allows statement execution if the comparison is satisfied.	&IF {tok1 } {EQ} {tok2 } executable
	{&$ } {NE} {&$ } statement
	{* } {LT} {&* }
	{LE}
	{GT}
	{GE}

Indicates number of nonblank characters in next token.	&LENGTH tok
Allows the use of the literal value of the token.	&LITERAL tok

Repetitively executes a sequence of statements.	&LOOP {n } {m }
	{label} {condition}

Punches a card with the defined tokens.	&PUNCH tok1 [tok2 ...]
Reads the next line (or lines) from the terminal.	&READ [n ARGS VARS var1 [var2 ...] 1]

<table>
<tr><th></th><th></th></tr>
<tr><td>Skips subsequent statements.</td><td>&SKIP [n | 1]</td></tr>
<tr><td>Types blank lines at the terminal.</td><td>&SPACE [n | 1]</td></tr>
<tr><td>Places a line of tokens in the console stack.</td><td>&STACK [LIFO|FIFO] [tok1 [tok2 ...]]</td></tr>
<tr><td>Extracts the desired string from the given token.</td><td>&SUBSTR tok i [j]</td></tr>
<tr><td>Types time information on the terminal.</td><td>&TIME [ON|OFF|RESET|TYPE]</td></tr>
<tr><td>Prints the tokens at the terminal.</td><td>&TYPE tok1 [tok2 ...]</td></tr>
<tr><td>EXTERNAL CP Class G
Creates an external interruption condition on the virtual machine.</td><td>EXTernal [code | 40]</td></tr>
</table>

Description	Format
FILEDEF CMS Simulates OS JCL (Job Control Language) data definition (DD) statements.	FIledef {ddname | nn | *} Terminal [(optA optD()]] PRint | PUnch | Reader [(optD()]] DISK [fn ft | FILE ddname] [fm | A1] [(optionB optionD()]] [[DISK fn ft | FILE ddname] [fm | A1]] {DSN ? | DSN qual1 qual2...} [(option B optionD()]] DUMMY [(optionD()]] TAPn [optionC optionD()]] CLEAR optA: [UPCASE|LOWCASE] optB: [KEYLEN n] [XTENT {n|50}] [CONCAT] [LIMCT nn] [OPTCD {A|E|F|R}] [DISP MOD] [MEMBER membername] [DSORG{PS|PO|DA|IS}]

```
                                    optC: [7TRACK|9TRACK][TRTCH {0|0C|OT|E|ET}]
                                          [DEN {200|556|800|1600|6250}]

                                    optD: [PERM] [CHANGE|NOCHANGE]
                                          [RECFM{F|FB|V|VB|U|VS|VBS|FS|FBS|A|M}]
                                          [LRECL nn] [{BLOCK|BLKSIZE} nn]

FLUSH                    CP Class D  FLush  raddr [ALL] [HOld]
  Halts and immediately purges or
  holds the current spool file.

FLUSH                          RSCS  FLush [linkid][spoolid|*][ALL|HOld]
Stop and delete the current
file from further processing

FORCE                    CP Class A  FORCE  userid
  Forces logout of the named user.

FORMAT                          CMS  FORMAT cuu mode [nocyl][(options ...[)]]
  Formats a disk for use by CMS.     options:
                                     ┌      ┐
                                     |LABEL |
                                     |RECOMP|
                                     └      ┘
```

Description	Format
FORMAT Format disk service aid	Format Service Aid Control Statements FORMAT,devadr,devtype,volser,startcyladr,endcyladr ALLOCATE,devadr,devtype,volser TEMP,startcyladr,endcyladr PERM,startcyladr,endcyladr TDSK,startcyladr,endcyladr DRCT,startcyladr,endcyladr END FORMAT,devadr,devtype,volser,LABEL
FREE CP Class D Releases previously held user spool files.	FRee userid [Printer] [PUnch] [ALL]
FREE RSCS Causes I/O transmission on a particular link to resume.	FRee [linkid]
FWDSPACE RSCS Causes the current file being processed to be repositioned in a forward direction.	FWdspace [linkid][nnn]

GENDIRT CMS Creates auxiliary module directories.	GENDIRT directoryname [targetmode]
GENMOD CMS Generates absolute nonrelocatable files (MODULE files).	Genmod [fn [ft [fm]]] [(options...[)]] options: ┌ ┐ ┌ ┐ ┌ ┐ │NOMAP│ │STR │ │FROM ent1│ [SYSTEM] │MAP │ │NOSTR│ │TO ent2│ └ ┘ └ ┘ └ ┘
GEN3705 CMS Invokes 3705 Stage 2 service aid.	GEN3705 fn ft [fm] [(options...[)]] options: [RUN|NORUN][SAVE|NOSAVE]
GLOBAL CMS Defines CMS libraries to be searched for macros and subroutines.	GLobal ⎧MACLIB⎫ [libname... libname8] ⎩TXTLIB⎭
HALT CP Class A Stops any active channel program on the real device specified.	HALT raddr

Description	Format
HB CMS Immediate Command Halts the execution of CMS batch virtual machine at the end of the current job.	HB
HO CMS Immediate Command Halts the current CMS tracing operation.	HO
HOLD CP Class D Defers processing of specified spool output.	HOld userid [Printer PUnch ALL]
HOLD RSCS Suspends file transmission temporarily for a particular link.	HOld [linkid] IMMed
HT CMS Immediate Command Halts typing at the terminal.	HT
HX CMS Immediate Command Halts the execution of the current CMS operation.	HX

IBCDASDI
Initialize disk service aid to format disk for OS or DOS use.

IBCDASDI Control Statements:

```
JOB [user information]


MSG TODEV=xxxx,TOADDR=cuu

DADEF TODEV=xxxx,TOADDR=cuu[,IPL=YES]
 {,VOLID={serial|SCRATCH}}
 [,MODEL=n][,CYLNO=nnn]
 [,FLAGTEST=NO][,PASSES=n][,BYPASS=YES]

VLD NEWVOLID=serial{,VOLPASS={0|1}}
 [,OWNERID=xxxxxxxxxx][,ADDLABEL=n]

VTOCD STRTADR=nnnnn,EXTENT=nnnn

END [user information]

GETALT TODEVxxxx,TOADDR=cuu
 ,TRACK=cccchhhh,VOLID=serial
 [,FLAGTEST=NO][,PASSES=n]
 [,BYPASS=YES][,MODEL=n]
```

Description	Format
INCLUDE CMS Brings additional TEXT files into storage.	INclude fn... [(options...[)] options: CLEAR RESET {entry} NOMAP NOINV NOREP NOCLEAR {*} MAP INV REP TYPE NOAUTO NOLIBE NODUP NOTYPE AUTO LIBE DUP [START] [SAME] [ORIGIN hexloc]
INDICATE CP Class G Displays the use of and contention for major system resources.	INDicate LOAD USER
INDICATE CP Class E Displays the use of and contention for major system resources.	INDicate LOAD USER [* userid] Queues I/O PAGing {WAIT ALL}

```
IPL                           CP Class G  Ipl { vaddr [cylno]  [CLear   ][STOP]  [PARM{p1 p2...}] }
  Initiates a program load on the             { systemname     [NOClear]                           }
  virtual machine.

LINK                          CP Class G  LINK  [To] userid vaddr1 [As] vaddr2
  Permits one user to access mini-              [mode] [[PASS=  ] password]
  disks belonging to another user.

LISTDS                               CMS  LISTDS [?|dsname]{fm|*}[(options...[)]]
  Display information about               options: [FO[RMAT]] [PDS]
  data set or file.

                                                     ┌    ┌    ┌   ┐┐┐
LISTFILE                             CMS  Listfile   |fn  |ft  |fm||| [(options...[)]
  Lists information about CMS                        | *  | *  | *|||
  files.                                  options:   └    └    └   ┘┘┘
                                           [Header|NOHeader]  [EXec|APpend]

                                           [FName|FType|FMode|FOrmat|ALloc|Date|Label]

LKED                                 CMS  LKED fn [(options...[)]]
  Creates 3705 load module.                options:
                                            [NCAL] [LET] [ALIGN2] [NE] [OL] [RENT]
                                            [REUS] [REFR] [OVLY] [XCAL] [TERM|NOTERM]
                                            [NAME membername] [LIBE libraryname]
                                            [XREF|MAP|LIST] [PRINT|DISK|NOPRINT]
```

Description	Format
LOAD CMS Brings TEXT files into storage and establishes links.	LOAD fn ... [(options...[)]] options: [CLEAR / NOCLEAR] [RESET {entry / *}] [NOMAP / MAP] [NOINV / INV] [NOREP / REP] [START] [TYPE / NOTYPE] [NOAUTO / AUTO] [NOLIBE / LIBE] [ORIGIN {hexloc / TRANS}] [NODUP / DUP]
LOADBUF CP Class D Loads UCS (Universal Character Set) buffer on the real printer device.	LOADBUF raddr {UCS name [Fold] [Ver] / FCB name [Index [nn]]}
LOADMOD CMS Brings a single MODULE file into storage.	LOADMod fn [ft fm]
LOADVFCB CP Class G Loads a forms control image for a virtual 3211 printer.	LOADFVCB vaddr FCB name [Index [nn]]

<table>
<tr><td>LOCATE CP Class E
Provides the addresses of CP control blocks related to a specified user, virtual device, or real device.</td><td>LOCate {raddr|userid [vaddr]}</td></tr>
<tr><td>LOCK CP Class A
Locks specified pages in processor storage.</td><td>LOCK {userid|SYSTEM} firstpage lastpage [MAP]</td></tr>
<tr><td>LOGOFF CP Class Any
Terminates a terminal session.</td><td>LOG[off|out] [HOld]</td></tr>
<tr><td>LOGON CP Class Any
Initiates all virtual machine operation.</td><td>Log[on|in] userid [password] [Mask] [Noipl]</td></tr>
<tr><td>MACLIB CMS
Performs maintenance on macro libraries.</td><td>MAClib { GEN
ADD libname fn1 [fn2...]
REP
DEL libname memname1 [memname2...]
COMP libname
MAP libname } [(options...[)]]

options:
[TERM|PRINT|DISK]</td></tr>
</table>

Description	Format
MODMAP CMS Types a MODULE file load map.	MODmap fn

MONITOR CP Classes A and E
Starts or stops the recording of interruptions and other events that occur in the real machine.
	MONitor
	DIsplay{PERform|RESPonse|SCHedule|USER}
	ENable {INSTsim|DAStap|SEEKs|SYSprof }
	INTerval nnnnn[SEC|MIN]
	STArt {CPTRACE
	TAPE vaddr [MODE{800|1600|6250 }]}
	STOP {CPTRACE
	TAPE}

MOVEFILE CMS Moves data from one device to another device.	MOVEfile [inputddname [outputddname]] INMOVE OUTMOVE [(PDS[)]]
MESSAGE CP Classes A and B Sends text messages to other users, system operator or self.	Message {ALL|userid|*|OPerator} msgtext MSG

<table>
<tr><td>MESSAGE CP Class Any
Sends text messages to other users, system operator or self.</td><td>Message {userid|*|OPerator} msgtext
MSG</td></tr>
<tr><td>MSG RSCS
Allows message test to be sent via RSCS remote or local stations to any RSCS facility or VM/370 user.</td><td>Msg linkid msgtext</td></tr>
<tr><td>NCPDUMP CMS
Service aid spool files created by 3705 dumping operations.</td><td>NCPDUMP [DUMPxx] [(options...[)]]
options:
[ERASE][NOFORM][MNEMONIC]</td></tr>
</table>

<table>
<tr><th>Description</th><th>Format</th></tr>
<tr><td>NETWORK CP Class A
Stops communications to 3705 controllers or resources or 3270 remote equipment.</td><td>NETWORK HALT resource
SHUTDOWN [raddr|ALL]</td></tr>
<tr><td>NETWORK CP Class A and B
Provides controls for utilizing and controlling 3705 and its resources. Also provides a means of altering binary synchronous line poll delay interval.</td><td>NETWORK {LOAD raddr ncpname
DUMP raddr [IMMED|OFF|AUTO]
ENable [ALL|resources...]
DISABLE [ALL|resources...]
Query [OFFline|FREe|ALL|resources...|ACTive]
Display raddr hexloc1 [{-|:} [hexloc2|END] | {.} [bytecount|END]]
SHUTDOWN [ALL|raddr]
POLLdlay nnnn [ALL raddr]
VARY {ONline|OFFline|EP|NCP}[resources...]}</td></tr>
<tr><td>NETWORK CP Class F
Provides trace data on 3704 and 3705 resources.</td><td>NETWORK TRACE {BTU raddr|resource|END}</td></tr>
</table>

```
NOTREADY                 CP Class G |NOTReady  vaddr
  Simulates loss of ready status on|
  virtual spooling device.         |
                                   |
ORDER                    CP Class D |ORDer  r        ┐ (Reader  ) (CLass c1 CLass c2...)
  Provides a technique for ordering|       |userid| < Printer  >< spoolid1 spoolid2...>
  closed spool files.              |       |SYSTEM| ( PUnch   )
                                   |       └       ┘
                                   |
ORDER                    CP Class G |ORDer  (Reader  ) (CLass c1 CLass c2...)
  Provides a technique for ordering|       < Printer >< spoolid1 spoolid2... >
  closed spool files.              |       ( PUnch   )
                                   |
ORDER                        RSCS |[linkid] {spoolid1[spoolid2....]}
  Reorders file queue for a link.  |
                                   |
PRINT                         CMS |PRint  fn ft [fm|*][(options...[)]]
  Directs a specified spool file   |options:
  to the virtual printer.          | ┌    ┐ ┌              ┐  [UPCASE][HEX]
                                   | |CC  | |MEMBER  { *   }|  [LInecoun[nn|55]]
                                   | |NOCC| |        { name}|
                                   | └    ┘ └              ┘
                                   |
PUNCH                         CMS |PUnch fn ft [fm|*] (options...[)]]
  Directs a specified spool file to| options:
  the virtual punch.               | ┌        ┐┌              ┐
                                   | |HEADER  ||MEMBER{name } |
                                   | |NOHEADER||      {*    } |
                                   | └        ┘└              ┘
```

Description	Format
PURGE CP Class D Deletes a spooled file before reading, printing, or punching occurs.	PURge [userid | SYSTEM] { Reader | Printer | PUnch | ALL } [ALL | CLass c1 CLass c2 ... | spoolid1 spoolid2 ...]
PURGE CP Class G Deletes a spooled file before reading, printing, or punching occurs.	PURge { Reader | Printer | PUnch | ALL } [CLass c1 CLass c2 ... | spoolid1 spoolid2 ... | ALL]
PURGE RSCS Removes inactive file queued on a specified link.	PURge [linkid]{ALL|spoolid1[spoolid2...]}
QUERY Classes A,B,C,D,E,F, and G Provides operational status.	Query { LOGmsg | Names | Users [userid] | userid }
QUERY CP Classes A,E Provides the paging activity index or specified user priority or status of the Virtual Machine Assist feature.	Query { PAGing | PRIORity userid | SASsist }

```
QUERY                      CP Class B |Query
  Provides the current status of      |      ⎧ ⎧ DAsd     ┌       ┐ ⎫ ⎫
  all system devices.                 |      ⎪ ⎪ TApes    |ACTive | ⎪ ⎪
                                      |      ⎪ ⎨ LINES    |OFFline| ⎬ ⎪
                                      |      ⎪ ⎪ UR       |FREe   | ⎪ ⎪
                                      |      ⎪ ⎪ GRaf     |ATTach | ⎪ ⎪
                                      |      ⎪ ⎩ ALL      |ALL    | ⎭ ⎪
                                      |      ⎨            └       ┘   ⎬
                                      |      ⎪   DAsd volid           ⎪
                                      |      ⎪   TDsk                 ⎪
                                      |      ⎪   STORage              ⎪
                                      |      ⎪   raddr                ⎪
                                      |      ⎪   SYStem raddr         ⎪
                                      |      ⎩   DUMP                 ⎭
                                      |
QUERY                      CP Class D |Query ⎧ Files [CLass c] [userid]            ⎫
  Provides data on spooling           |      ⎪                                     ⎪
  operations.                         |      ⎪ Reader   ┌┌       ┐           ┐     ⎪
                                      |      ⎪ Printer  ||ALL    | [userid]|     ⎪
                                      |      ⎨ PUnch    ||CLass c|         |     ⎬
                                      |      ⎪          |└       ┘         |     ⎪
                                      |      ⎪          |  spoolid         |     ⎪
                                      |      ⎪          └                  ┘     ⎪
                                      |      ⎩ HOld                              ⎭
```

```
 ------------------------------------------------------------------------------------------
|         Description                 |                 Format                             |
|-------------------------------------|----------------------------------------------------|
|QUERY                        Class G |Query  / Time                                  \    |
|  In CP mode, use the QUERY command  |      |  Set                                    |    |
|  to request system status and       |      |  TERMinal                               |    |
|  machine configuration information  |      |                                         |    |
|                                     |      |  Files   [CLass c]                      |    |
|                                     |      |                                         |    |
|                                     |      |               / CHANNels \              |    |
|                                     |      |              |  GRaf      |             |    |
|                                     |      |              |  CONsole   |             |    |
|                                     |      |              |  Dasd      |             |    |
|                                     |      |              |  TApes     |             |    |
|                                     |      |              |  LINES     |             |    |
|                                     |     <  [Virtual]   <   UR         >           >    |
|                                     |      |              |  STORage   |             |    |
|                                     |      |              |  ALL       |             |    |
|                                     |      |               \ vaddr    /              |    |
|                                     |      |                                         |    |
|                                     |      |  Links         vaddr                    |    |
|                                     |      |                                         |    |
|                                     |      |  Reader       |spoolid|                 |    |
|                                     |      |  Printer      |ALL    |                 |    |
|                                     |      |  PUnch        |CLass c|                 |    |
|                                     |      |                                         |    |
|                                     |       \ PF[nn]                                /    |
 ------------------------------------------------------------------------------------------
```

QUERY CMS

In CMS mode, use the QUERY command to gather certain information about the virtual machine environment.

```
Query  ( BLIP                             )
       | RDYMSG                           |
       | LDRTBLS                          |
       | RELPAGE                          |
       | IMPCP                            |
       | IMPEX                            |
       | ABBREV                           |
       | REDTYPE                          |
       | PROTECT                          |
       | SEARCH                           |
       | DISK     (mode)                  |
       <          ( *  )                  >
       |                                  |
       |          (SYSTEM)                |
       | SYNONYM <USER   >                |
       |          (ALL   )                |
       |                                  |
       | FILEDEF                          |
       | MACLIB                           |
       | TXTLIB                           |
       | LIBRARY                          |
       | INPUT                            |
       ( OUTPUT                           )
```

QUERY RSCS

Queries RSCS linkid, file or status information.

```
Query  (linkid [Stat|Def|Queue]        )
       <File spoolid [Stat|RSCS|VM]    >
       (SYstem [Active]                )
```

Description	Format
READCARD CMS Reads data from the spooled card input device.	READcard { fn ft [fm|A] * [* [fm|A]] }
READY CP Class G Makes a device end interruption pending for the specified device.	READY vaddr
RELEASE CMS Makes a disk and its directory inaccessible to a virtual machine.	RELease { cuu|mode }
RENAME CMS Changes the name of a CMS file or files.	Rename fileid1 fileid2 [(options...[)]] options: [TYPE|NOTYPE] [NOUPDIRT|UPDIRT]
REPEAT CP Class D Increases the copies of, or holds, an output spool file.	REPeat raddr [[nn|1]|[nn]HOld]

REQUEST CP Class G Makes attention interruption pending.	REQuest
RESET CP Class G Clears all pending interruptions resets error conditions on the device specified.	RESET vaddr
REWIND CP Class G Rewinds a real tape drive.	REWIND vaddr
RO CMS Immediate Command Resumes recording of trace information previously suspended by the SO immediate command.	RO
RT CMS Immediate Command Resumes terminal typing.	RT
RUN CMS Initiates a series of functions for a file.	RUN fn [ft [fm]] [(args...)]

<table>
<tr><th>Description</th><th>Format</th></tr>
<tr><td>SAVENCP CMS
Reads/Loads 3705 control program load module.</td><td>SAVENCP fn [(options... [)]]
options:
[ENTRY symbol |CXFINIT]
[NAME ncpname|fn] [LIBE libname|fn]</td></tr>
<tr><td>SAVESYS CP Class E
Creates a copy of virtual machine contents as they currently exist.</td><td>SAVESYS systemname</td></tr>
<tr><td>SET CP Class A
Sets special CP preferred options.</td><td>SET { FAVored userid [xx]
REServe userid [OFF]
PRIORity userid nn
SASsist {ON|OFF} }</td></tr>
</table>

SET CP Class B

Establishes disposition for log messages and dumps.

```
SET   (LOGmsg   [nn|NULL]            )
      {                              }
      (DUmp{AUTO|raddr}[ALL|CP]      )
```

SET CP Class F

Sets recording mode for a device, or enables/disables soft machine check interrupts.

```
SET RECord(OFF                                                                   )
          )ON raddr LIMIT nn BYTE nn BIT n[ (AND)BYTE nn BIT n]                  {
          )                                [ (OR )             ]                 }
          (                                                                      )
     MODE (RETRY) (Quiet )
          (MAIN ) (Record)
```

SET CP Class G

The SET command controls various functions within your virtual machine.

```
Set  ( ACNT      (    )                          )
     ( MSG       (    )                          )
     ( WNG       ( ON )                          )
     ( IMSG      ( OFF)                          )
     ( RUN       (    )                          )
     { LINEDit   {    }                          }
     ( NOTRans   (    )                          )
     ( ECmode    (    )                          )
     ( ISAM      (    )                          )
     ( PAGEX     (    )                          )
     (                                           )
     ( EMSG {ON|OFF|CODE|TEXT}                   )
```

Description	Format
SET (cont.)	TIMER {ON|OFF|REAL} ASsist {OFF|[ON][SVC|NOSVC]} PFnn [IMMed|DElayed][pfdata#...] PFnn [TAB n1 n2...nn] PFnn COPY [resid]
SET CMS Controls various functions within your virtual machine.	SET [BLIP string[(count)] | BLIP ON | BLIP OFF] [INPUT a xx | xx yy] [LDRTBLS nn] [OUTPUT [xx a]] [RDYMSG SMSG | RDYMSG LMSM] [RELPAGE OFF | RELPAGE ON] [ABBREV OFF | ABBREV ON]

	IMPEX OFF IMPCP OFF REDTYPE OFF IMPEX ON IMPCP ON REDTYPE ON PROTECT OFF AUTOREAD ON PROTECT ON AUTOREAD OFF
SHUTDOWN CP Class A Checkpoints and terminates the current VM/370 operation.	SHUTDOWN
SLEEP CP Class Any Places the virtual machine in a dormant state with the terminal keyboard entry blocked.	SLeep [nn[SEC|MIN|HRs]]
SO CMS Immediate Command Suspends the recording of trace information during the execution command or program.	SO
SORT CMS Rearranges records within a file.	SORT fileid1 fileid2
SPACE CP Class D Forces single spacing on the printer.	SPAce raddr

```
|          Description                    |                      Format
|-----------------------------------------|---------------------------------------------------------------------
|SPOOL                       CP Class G   |SPool {Reader}   [CLass T     ] [CONT   ][CLOSE] [EOF  ][HOLD  ]
|   Changes spooling control options.     |      {vaddr }   [CLass{*|c}  ] [NOCONT ][PURGE] [NOEof][NOHOLD]
|                                         |
|                                         |      {Printer}  ( [[To ][userid ]] [HOld   ] [CONt   ] )
|                                         |      {PUnch  }  ( [[For][  *    ]] [NOHold ] [NOCont ] )
|                                         |      {vaddr  }  ( [    [SYSTEM ]]                     )
|                                         |                 ( [             ] [CLass A] [COpy nn] )
|                                         |                 ( [ OFF         ] [CLass c] [COpy 01] )
|                                         |                 (                                     )
|                                         |                 ( [CLOSE]                             )
|                                         |                 ( [PURGE]                             )
|                                         |
|                                         |      {CONsole}  ( [STArt][HOld  ] [CONt  ] [TErm  ] )
|                                         |      {vaddr  }  ( [STOp ][NOHold] [NOCont] [NOTErm] )
|                                         |                 (                                   )
|                                         |                 ( [To userid][CLass T][COpy nn][CLOSE] )
|                                         |                 ( [OFF      ][CLass c][COpy 01][PURGE] )
```

```
START                        CP Class D  STArt   ┌                                  ┐
  Restarts a drained device or                   │ Reader                           │
  changes its output spooling                    │ Printer                          │
  class.                                         │ PUnch                            │
                                                 │ ALL                              │
                                                 │                                  │
                                                 │[raddr[CLass c...][NOsep]]...     │
                                                 └                                  ┘

START                              CMS   START   ┌                  ┐
  Begins program execution.                      │{ entry } [args...]│
                                                 │{   *   }          │
                                                 └                  ┘

START                             RSCS   STArt   [linkid] ┌              ┐ [parm...]
  Activates an RSCS link that is                          │CLass c       │
  in the deactive state to start                          │LINE vaddr    │
  processing files.                                       │TASK name     │
                                                          │TYPE driverid │
                                                          └              ┘

STATE                              CMS   STATE   fn ft [fm]
  Verifies the existence of a file.

STCP                         CP Class C  STCP   {{ hexloc } hexwd1 [hexwd2...]}
  Alters real storage locations.                {{Lhexloc }                   }
                                                { Shexloc   hexdata           }
```

Description	Format
STORE CP Class G Alters virtual machine storage, PSW, and registers.	STore { hexloc Lhexloc hexwd1 [hexwd2...] Shexloc hexdata {Greg | Yreg | Xreg} hexwd1 [hexwd2...] Psw [hexwd1] hexwd2 STATUS }
SVCTRACE CMS Records information about supervisor call instructions.	SVCTrace {ON | OFF}
SYNONYM CMS Specifies alternate names for invoking CMS commands.	SYNonym [fn [ft | SYNONYM [fm | A1]]] [(options...[)]] Options: [NOSTD|STD][CLEAR]
SYSTEM CP Class G Simulates virtual machine console functions.	SYStem {CLEAR | RESET | RESTART}

```
|TAG                    CP Class G |TAG   DEV {Printer|PUnch|CONsole|vaddr}[text]
|  Appends or queries the TAG text |      FILE  spoolid [text]
|  to a VM/370 spool file utilized |      QUERY {DEV{Printer|PUnch|CONsole|vaddr}}
|  by subsystems (such as          |            {FILE spoolid                    }
|  RSCS).                          |
|                                  |
|                                  |
|TAPE                          CMS |TAPE  ( DUMP {fn} {ft} [fm] [(optA optB optD()]]
|  Performs tape to disk or disk to|      (      {* } {* } [* ]
|  tape operations for CMS data    |      (
|  sets.                           |      ( LOAD [fn [ft [fm]]] [(optB optC optD()]]
|                                  |      (      [*  [*  [A ]]]
|                                  |      (
|                                  |      ( SCAN[fn|*[ft|*]]    [(optB optC optD()]]
|                                  |      (
|                                  |      ( SKIP[fn|*[ft|*]]    [(optB optC optD()]]
|                                  |      (
|                                  |      ( MODEset [(optD()]]
|                                  |      (
|                                  |      ( tapcmd [n] [(optD()]]
|                                  |      (        [1]
|                                  |
|                                  |optA:  [WTM  ]  optB:  [NOPRint]  optC:  [EOF n]
|                                  |       [NOWTM]         [PRint  ]         [EOT  ]
|                                  |                       [DISK   ]         [EOF 1]
|                                  |                       [Term   ]
```

<table>
<tr><th>Description</th><th>Format</th></tr>
<tr><td>TAPE (cont.)</td><td>optD: |TAPi|ccu| [TRTCH {O|OC|OT|E|ET}] |7TRACK|
|TAP1|181| |9TRACK|
[DEN{200|556|800|1600|6250}]
tapcmd: [BSF|BSR|ERG|FSF|FSR|REW|RUN|WTM]</td></tr>
<tr><td>TAPPDS CMS
Loads an OS partitioned data set (PDS) file or card-image records from tape to disk.</td><td>TAPPDS |fn |ft |fm||| [(options...[)]]
|* |* |* |||
|A1|||
options:
|UPDATE| |COL1 | |TAPn| |END | |MAXTEN |
|NOPDS | |NOCOL1| |TAP1| |NOEND| |NOMAXTEN|
|PDS |</td></tr>
</table>

TERMINAL CP Class G

Changes parameters for terminal operations.

```
TERMinal  { CHardel   { ON   } }
          { LINEDel   { OFf  } }
          { LINENd    { char } }
          { EScape             }
          {                    }
          { Mask      { ON   } }
          { APL       { OFf  } }
          { ATtn               }
          {                    }
          { MODE {CP|VM}       }
          {                    }
          { LINESize  {nnn }   }
```

TRACE CP Class G

Traces and records program execution.

```
TRace  { { SVC       }                                  }
       { { I/O       }                                  }
       { { PROgram   } [ Printer                    ]   }
       { { EXTernal  } [                            ]   }
       { { PRIV      } [ [BOTH    ]  [RUN  ]        ]   }
       { { SIO       } [ [TERMinal]  [NORun]        ]   }
       { { CCW       } [                            ]   }
       { { BRanch    } [ OFf                        ]   }
       { { INSTruct  }                                  }
       { { ALL       }                                  }
       { { CSW       }                                  }
       {                                                }
       {   END                                          }
```

<table>
<tr><th>Description</th><th>Format</th></tr>
<tr><td>TRACE RSCS
Traces certain line and error activity for a specified link.</td><td>TRace linkid[ALL|ERRors|END]</td></tr>
<tr><td>TRANSFER CP Class D
Transfers closed reader spool files.</td><td>TRANsfer [userid|SYSTEM] {spoolid|CLass c|ALL} [To|FROM] {userid|ALL}</td></tr>
<tr><td>TRANSFER CP Class G
Transfers closed reader spool files.</td><td>TRANsfer {spoolid|CLass c|ALL} [[FROM|TO] {userid|ALL}]</td></tr>
<tr><td>TXTLIB CMS
Performs maintenance on a library of TEXT files (object modules).</td><td>TXTlib {
GEN libn fn ...
ADD libn fn ...
DEL libn membername...
MAP libn [(TERM)|(PRINT)|(DISK)]
}</td></tr>
</table>

TYPE CMS

Types all or part of a file at a terminal.

```
Type  fn ft [fm] ┌┌rec1┐┌recn┐┐ [(options...[)]]
                 ││ *  ││ *  ││
                 ││ 1  ││    ││
                 └└    ┘└    ┘┘
options:
┌                             ┐       ┌                ┐
│COL {xxxxx} - {yyyyy}│ [HEX] │MEMBER]{*    }│
│    {  1  }   {lrec1}│       │       {name }│
└                             ┘       └                ┘
```

UNLOCK CP Class A

Releases storage.

```
UNLOCK  {{userid|SYSTEM} firstpage lastpage}
        {VIRT=REAL                          }
```

UPDATE CMS

Makes changes in a file as defined by control cards in a record file.

```
              ┌                                   ┐
Update fn1│ft1       │fm1[fn2[ft2[fm2]]]││[(option...[)]]
          │ASSEMBLE  │A1                ││
          └          └                  ┘┘
options:
┌      ┐┌      ┐┌      ┐┌      ┐┌      ┐┌      ┐┌     ┐┌      ┐
│REP   ││NOSEQ8││INC   ││CTL   ││STK   ││NOTERM││PRINT││STORNO│
│NOREP ││SEQ8  ││NOINC ││NOCTL ││NOSTK ││TERM  ││DISK ││NOSTOR│
└      ┘└      ┘└      ┘└      ┘└      ┘└      ┘└     ┘└      ┘

Control Statements:
  ./ S [segstrt[segincr[label]]]
  ./ I segno [$][segstrt[segincr]]]
  ./ D segno1 [segno2][$]
  ./ R segno1 [segno2][$[segstrt[segincr]]]
  ./ * comment
```

Description	Format
<u>VARY</u> CP Class B Varies the availability of a device.	VARY {ONLine } raddr... {OFfline}
<u>VMFDUMP</u> Service aid command that is the processor of system dumps.	VMFDUMP [DUMPxx] [ERASE] [NOMAP] [NOHEX] [NOFORM] [NOVIRT]

<u>WARNING</u> CP Classes A,B
Sends high priority messages.
	Warning {userid } msgtext
	WNG {OPerator}
	{ALL }

ZAP CMS

Modifies or dumps MODULE, LOADLIB, or TXTLIB files.

```
    (MODULE )
ZAP < LOADLIB >[ libname¹ ... libname³][ (options...[) ]]
    (TXTLIB )
options:
┌                ┐┌       ┐
|TERM            ||PRINT  |
|INPUT filename  ||NOPRINT|
└                ┘└       ┘

Control Statements:
 BASE   address

                        ┌                                          ┐
 DUMP  (membername)  |csectname [startaddress [endaddress]]|
       (modulename)  |ALL                                   |
                        └                                          ┘

 NAME  (membername)  [csectname]
       (modulename)

 (VERIFY)  disp   data
 (VER   )

 REP   disp  data

 * comment

 END
```

Dec.	Hex	Instruction (RR)	Graphics and Controls BCDIC	EBCDIC(1)	ASCII	7-Track Tape BCDIC(2)	Card Code	Binary
0	00			NUL	NUL		12-0-1-8-9	0000 0000
1	01			SOH	SOH		12-1-9	0000 0001
2	02			STX	STX		12-2-9	0000 0010
3	03			ETX	ETX		12-3-9	0000 0011
4	04	SPM		PF	EOT		12-4-9	0000 0100
5	05	BALR		HT	ENQ		12-5-9	0000 0101
6	06	BCTR		LC	ACK		12-6-9	0000 0110
7	07	BCR		DEL	BEL		12-7-9	0000 0111
8	08	SSK			BS		12-8-9	0000 1000
9	09	ISK			HT		12-1-8-9	0000 1001
10	0A	SVC		SMM	LF		12-2-8-9	0000 1010
11	0B			VT	VT		12-3-8-9	0000 1011
12	0C			FF	FF		12-4-8-9	0000 1100
13	0D			CR	CR		12-5-8-9	0000 1101
14	0E	MVCL		SO	SO		12-6-8-9	0000 1110
15	0F	CLCL		SI	SI		12-7-8-9	0000 1111
16	10	LPR		DLE	DLE		12-11-1-8-9	0001 0000
17	11	LNR		DC1	DC1		11-1-9	0001 0001
18	12	LTR		DC2	DC2		11-2-9	0001 0010
19	13	LCR		TM	DC3		11-3-9	0001 0011
20	14	NR		RES	DC4		11-4-9	0001 0100
21	15	CLR		NL	NAK		11-5-9	0001 0101
22	16	OR		BS	SYN		11-6-9	0001 0110
23	17	XR		IL	ETB		11-7-9	0001 0111
24	18	LR		CAN	CAN		11-8-9	0001 1000
25	19	CR		EM	EM		11-1-8-9	0001 1001
26	1A	AR		CC	SUB		11-2-8-9	0001 1010
27	1B	SR		CU1	ESC		11-3-8-9	0001 1011
28	1C	MR		IFS	FS		11-4-8-9	0001 1100
29	1D	DR		IGS	GS		11-5-8-9	0001 1101
30	1E	ALR		IRS	RS		11-6-8-9	0001 1110
31	1F	SLR		IUS	US		11-7-8-9	0001 1111
32	20	LPDR		DS	SP		11-0-1-8-9	0010 0000
33	21	LNDR		SOS	| !		0-1-9	0010 0001
34	22	LTDR		FS	"		0-2-9	0010 0010
35	23	LCDR			#		0-3-9	0010 0011
36	24	HDR		BYP	$		0-4-9	0010 0100
37	25	LRDR		LF	%		0-5-9	0010 0101
38	26	MXR		ETB	&		0-6-9	0010 0110
39	27	MXDR		ESC	'		0-7-9	0010 0111
40	28	LDR			(		0-8-9	0010 1000
41	29	CDR			)		0-1-8-9	0010 1001
42	2A	ADR		SM	*		0-2-8-9	0010 1010
43	2B	SDR		CU2	+		0-3-8-9	0010 1011
44	2C	MDR			,		0-4-8-9	0010 1100
45	2D	DDR		ENQ	-		0-5-8-9	0010 1101
46	2E	AWR		ACK	.		0-6-8-9	0010 1110
47	2F	SWR		BEL	/		0-7-8-9	0010 1111
48	30	LPER			0		12-11-0-1-8-9	0011 0000
49	31	LNER			1		1-9	0011 0001
50	32	LTER		SYN	2		2-9	0011 0010
51	33	LCER			3		3-9	0011 0011
52	34	HER		PN	4		4-9	0011 0100
53	35	LRER		RS	5		5-9	0011 0101
54	36	AXR		UC	6		6-9	0011 0110
55	37	SXR		EOT	7		7-9	0011 0111
56	38	LER			8		8-9	0011 1000
57	39	CER			9		1-8-9	0011 1001
58	3A	AER			:		2-8-9	0011 1010
59	3B	SER		CU3	;		3-8-9	0011 1011
60	3C	MER		DC4	<		4-8-9	0011 1100
61	3D	DER		NAK	=		5-8-9	0011 1101
62	3E	AUR			>		6-8-9	0011 1110
63	3F	SUR		SUB	?		7-8-9	0011 1111

Figure 7. Code Translate Table
(Part 1 of 5)

Dec.	Hex	Instruction (RX)	Graphics and Controls			7-Track Tape	Card Code	Binary
			BCDIC	EBCDIC(1)	ASCII	BCDIC(2)		
64	40	STH		Sp Sp	@	(3)	no punches	0100 0000
65	41	LA			A		12-0-1-9	0100 0001
66	42	STC			B		12-0-2-9	0100 0010
67	43	IC			C		12-0-3-9	0100 0011
68	44	EX			D		12-0-4-9	0100 0100
69	45	BAL			E		12-0-5-9	0100 0101
70	46	BCT			F		12-0-6-9	0100 0110
71	47	BC			G		12-0-7-9	0100 0111
72	48	LH			H		12-0-8-9	0100 1000
73	49	CH			I		12-1-8	0100 1001
74	4A	AH		¢ ¢	J		12-2-8	0100 1010
75	4B	SH	.	. .	K	B A 8 2 1	12-3-8	0100 1011
76	4C	MH	⌑)	< <	L	B A 8 4	12-4-8	0100 1100
77	4D		[	((	M	B A 8 4 1	12-5-8	0100 1101
78	4E	CVD	<	+ +	N	B A 8 4 2	12-6-8	0100 1110
79	4F	CVB	‡	| |	O	B A 8 4 2 1	12-7-8	0100 1111
80	50	ST	& +	& &	P	B A	12	0101 0000
81	51				Q		12-11-1-9	0101 0001
82	52				R		12-11-2-9	0101 0010
83	53				S		12-11-3-9	0101 0011
84	54	N			T		12-11-4-9	0101 0100
85	55	CL			U		12-11-5-9	0101 0101
86	56	O			V		12-11-6-9	0101 0110
87	57	X			W		12-11-7-9	0101 0111
88	58	L			X		12-11-8-9	0101 1000
89	59	C			Y		11-1-8	0101 1001
90	5A	A		! !	Z		11-2-8	0101 1010
91	5B	S	$	$ $	[	B 8 2 1	11-3-8	0101 1011
92	5C	M	*	* *	\	B 8 4	11-4-8	0101 1100
93	5D	D	]	))	]	B 8 4 1	11-5-8	0101 1101
94	5E	AL	;	; ;	¬ ^	B 8 4 2	11-6-8	0101 1110
95	5F	SL	Δ	¬ ¬	_	B 8 4 2 1	11-7-8	0101 1111
96	60	STD	-	- -	`	B	11	0110 0000
97	61		/	/ /	a	A 1	0-1	0110 0001
98	62				b		11-0-2-9	0110 0010
99	63				c		11-0-3-9	0110 0011
100	64				d		11-0-4-9	0110 0100
101	65				e		11-0-5-9	0110 0101
102	66				f		11-0-6-9	0110 0110
103	67	MXD			g		11-0-7-9	0110 0111
104	68	LD			h		11-0-8-9	0110 1000
105	69	CD			i		0-1-8	0110 1001
106	6A	AD		¦	j		12-11	0110 1010
107	6B	SD	,	, ,	k	A 8 2 1	0-3-8	0110 1011
108	6C	MD	% (	% %	l	A 8 4	0-4-8	0110 1100
109	6D	DD	Y	_ _	m	A 8 4 1	0-5-8	0110 1101
110	6E	AW	\	> >	n	A 8 4 2	0-6-8	0110 1110
111	6F	SW	⧻	? ?	o	A 8 4 2 1	0-7-8	0110 1111
112	70	STE			p		12-11-0	0111 0000
113	71				q		12-11-0-1-9	0111 0001
114	72				r		12-11-0-2-9	0111 0010
115	73				s		12-11-0-3-9	0111 0011
116	74				t		12-11-0-4-9	0111 0100
117	75				u		12-11-0-5-9	0111 0101
118	76				v		12-11-0-6-9	0111 0110
119	77				w		12-11-0-7-9	0111 0111
120	78	LE			x		12-11-0-8-9	0111 1000
121	79	CE		`	y		1-8	0111 1001
122	7A	AE	ƀ	: :	z	A	2-8	0111 1010
123	7B	SE	# ▪	# #	{	8 2 1	3-8	0111 1011
124	7C	ME	@ '	@ @	¦	8 4	4-8	0111 1100
125	7D	DE	:	' '	}	8 4 1	5-8	0111 1101
126	7E	AU	>	= =	~	8 4 2	6-8	0111 1110
127	7F	SU	√	" "	DEL	8 4 2 1	7-8	0111 1111

Figure 7. Code Translate Table (Part 2 of 5)

Dec.	Hex	Instruction and Format	Graphics and Controls BCDIC	EBCDIC(1)		ASCII	7-Track Tape BCDIC(2)	Card Code	Binary
128	80	SSM -S						12-0-1-8	1000 0000
129	81			a	a			12-0-1	1000 0001
130	82	LPSW -S		b	b			12-0-2	1000 0010
131	83	Diagnose		c	c			12-0-3	1000 0011
132	84	WRD SI		d	d			12-0-4	1000 0100
133	85	RDD		e	e			12-0-5	1000 0101
134	86	BXH RS		f	f			12-0-6	1000 0110
135	87	BXLE		g	g			12-0-7	1000 0111
136	88	SRL		h	h			12-0-8	1000 1000
137	89	SLL		i	i			12-0-9	1000 1001
138	8A	SRA						12-0-2-8	1000 1010
139	8B	SLA			{			12-0-3-8	1000 1011
140	8C	SRDL			≤			12-0-4-8	1000 1100
141	8D	SLDL			(			12-0-5-8	1000 1101
142	8E	SRDA			+			12-0-6-8	1000 1110
143	8F	SLDA			+			12-0-7-8	1000 1111
144	90	STM						12-11-1-8	1001 0000
145	91	TM SI		j	j			12-11-1	1001 0001
146	92	MVI		k	k			12-11-2	1001 0010
147	93	TS -S		l	l			12-11-3	1001 0011
148	94	NI SI		m	m			12-11-4	1001 0100
149	95	CLI		n	n			12-11-5	1001 0101
150	96	OI		o	o			12-11-6	1001 0110
151	97	XI		p	p			12-11-7	1001 0111
152	98	LM -RS		q	q			12-11-8	1001 1000
153	99			r	r			12-11-9	1001 1001
154	9A							12-11-2-8	1001 1010
155	9B				}			12-11-3-8	1001 1011
156	9C	SIO, SIOF S			⌑			12-11-4-8	1001 1100
157	9D	TIO, CLRIO			)			12-11-5-8	1001 1101
158	9E	HIO, HDV			±			12-11-6-8	1001 1110
159	9F	TCH			■			12-11-7-8	1001 1111
160	A0				-			11-0-1-8	1010 0000
161	A1			~	°			11-0-1	1010 0001
162	A2			s	s			11-0-2	1010 0010
163	A3			t	t			11-0-3	1010 0011
164	A4			u	u			11-0-4	1010 0100
165	A5			v	v			11-0-5	1010 0101
166	A6			w	w			11-0-6	1010 0110
167	A7			x	x			11-0-7	1010 0111
168	A8			y	y			11-0-8	1010 1000
169	A9			z	z			11-0-9	1010 1001
170	AA							11-0-2-8	1010 1010
171	AB				└			11-0-3-8	1010 1011
172	AC	STNSM SI			┌			11-0-4-8	1010 1100
173	AD	STOSM			[			11-0-5-8	1010 1101
174	AE	SIGP -RS			≥			11-0-6-8	1010 1110
175	AF	MC -SI			●			11-0-7-8	1010 1111
176	B0				^{0}			12-11-0-1-8	1011 0000
177	B1	LRA -RX			^{1}			12-11-0-1	1011 0001
178	B2	See below			^{2}			12-11-0-2	1011 0010
179	B3				^{3}			12-11-0-3	1011 0011
180	B4				^{4}			12-11-0-4	1011 0100
181	B5				^{5}			12-11-0-5	1011 0101
182	B6	STCTL RS			^{6}			12-11-0-6	1011 0110
183	B7	LCTL			^{7}			12-11-0-7	1011 0111
184	B8				^{8}			12-11-0-8	1011 1000
185	B9				^{9}			12-11-0-9	1011 1001
186	BA	CS RS						12-11-0-2-8	1011 1010
187	BB	CDS			┘			12-11-0-3-8	1011 1011
188	BC				┐			12-11-0-4-8	1011 1100
189	BD	CLM RS			]			12-11-0-5-8	1011 1101
190	BE	STCM			≠			12-11-0-6-8	1011 1110
191	BF	ICM			—			12-11-0-7-8	1011 1111

Figure 7. Code Translate Table (Part 3 of 5)

Dec.	Hex	Instruction (SS)	Graphics and Controls BCDIC	EBCDIC(1)	ASCII	7-Track Tape BCDIC(2)	Card Code	Binary
192	C0		?	{		B A 8 2	12-0	1100 0000
193	C1		A	A	A	B A 1	12-1	1100 0001
194	C2		B	B	B	B A 2	12-2	1100 0010
195	C3		C	C	C	B A 2 1	12-3	1100 0011
196	C4		D	D	D	B A 4	12-4	1100 0100
197	C5		E	E	E	B A 4 1	12-5	1100 0101
198	C6		F	F	F	B A 4 2	12-6	1100 0110
199	C7		G	G	G	B A 4 2 1	12-7	1100 0111
200	C8		H	H	H	B A 8	12-8	1100 1000
201	C9		I	I	I	B A 8 1	12-9	1100 1001
202	CA						12-0-2-8-9	1100 1010
203	CB						12-0-3-8-9	1100 1011
204	CC			ʃ			12-0-4-8-9	1100 1100
205	CD						12-0-5-8-9	1100 1101
206	CE			⑂			12-0-6-8-9	1100 1110
207	CF						12-0-7-8-9	1100 1111
208	D0		!	}		B 8 2	11-0	1101 0000
209	D1	MVN	J	J	J	B 1	11-1	1101 0001
210	D2	MVC	K	K	K	B 2	11-2	1101 0010
211	D3	MVZ	L	L	L	B 2 1	11-3	1101 0011
212	D4	NC	M	M	M	B 4	11-4	1101 0100
213	D5	CLC	N	N	N	B 4 1	11-5	1101 0101
214	D6	OC	O	O	O	B 4 2	11-6	1101 0110
215	D7	XC	P	P	P	B 4 2 1	11-7	1101 0111
216	D8		Q	Q	Q	B 8	11-8	1101 1000
217	D9		R	R	R	B 8 1	11-9	1101 1001
218	DA						12-11-2-8-9	1101 1010
219	DB						12-11-3-8-9	1101 1011
220	DC	TR					12-11-4-8-9	1101 1100
221	DD	TRT					12-11-5-8-9	1101 1101
222	DE	ED					12-11-6-8-9	1101 1110
223	DF	EDMK					12-11-7-8-9	1101 1111
224	E0		‡	\		A 8 2	0-2-8	1110 0000
225	E1						11-0-1-9	1110 0001
226	E2		S	S	S	A 2	0-2	1110 0010
227	E3		T	T	T	A 2 1	0-3	1110 0011
228	E4		U	U	U	A 4	0-4	1110 0100
229	E5		V	V	V	A 4 1	0-5	1110 0101
230	E6		W	W	W	A 4 2	0-6	1110 0110
231	E7		X	X	X	A 4 2 1	0-7	1110 0111
232	E8		Y	Y	Y	A 8	0-8	1110 1000
233	E9		Z	Z	Z	A 8 1	0-9	1110 1001
234	EA						11-0-2-8-9	1110 1010
235	EB						11-0-3-8-9	1110 1011
236	EC			⑁			11-0-4-8-9	1110 1100
237	ED						11-0-5-8-9	1110 1101
238	EE						11-0-6-8-9	1110 1110
239	EF						11-0-7-8-9	1110 1111
240	F0	SRP	0	0	0	8 2	0	1111 0000
241	F1	MVO	1	1	1	1	1	1111 0001
242	F2	PACK	2	2	2	2	2	1111 0010
243	F3	UNPK	3	3	3	2 1	3	1111 0011
244	F4		4	4	4	4	4	1111 0100
245	F5		5	5	5	4 1	5	1111 0101
246	F6		6	6	6	4 2	6	1111 0110
247	F7		7	7	7	4 2 1	7	1111 0111
248	F8	ZAP	8	8	8	8	8	1111 1000
249	F9	CP	9	9	9	8 1	9	1111 1001
250	FA	AP		|			12-11-0-2-8-9	1111 1010
251	FB	SP					12-11-0-3-8-9	1111 1011

Figure 7. Code Translate Table (Part 4 of 5)

Dec.	Hex	Instruction (SS)	Graphics and Controls			7-Track Tape	Card Code	Binary
			BCDIC	EBCDIC(1)	ASCII	BCDIC(2)		
252	FC	MP					12-11-0-4-8-9	1111 1100
253	FD	DP					12-11-0-5-8-9	1111 1101
254	FE						12-11-0-6-8-9	1111 1110
255	FF						12-11-0-7-8-9	1111 1111

1. Two columns of EBCDIC graphics are shown. The first gives standard bit pattern assignments. The second shows the T-11 and TN text printing chains (120 graphics).
2. Add C (check bit) for odd or even parity as needed, except as noted.
3. For even parity use CA.

Op code (S format)

B202 - STIDP
B203 - STIDC
B204 - SCK
B205 - STCK
B206 - SCKC

B207 - STCKC
B208 - SPT
B209 - STPT
B20A - SPKA
B20B - IPK

B20D - PTLB
B210 - SPX
B211 - STPX
B212 - STAP
B213 - RRB

Figure 7. Code Translate Table (Part 5 of 5)

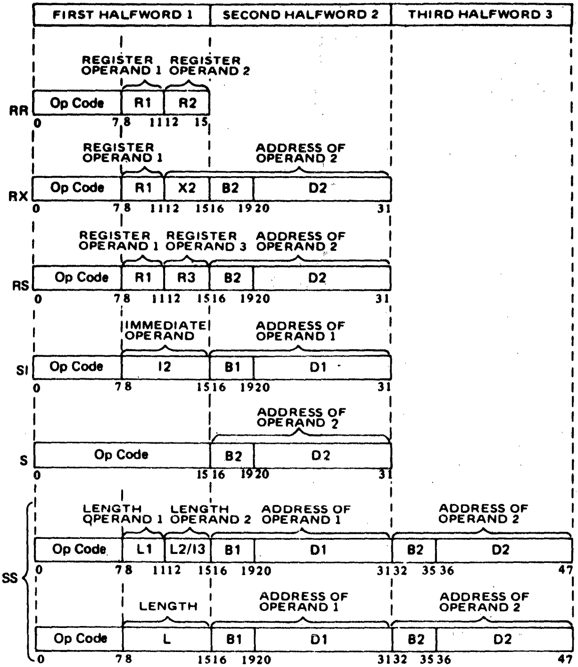


Figure 8. Machine Instruction Formats

CR	Bits	Name of field	Associated with	Init.
0	0	Block - multiplex'g control	Block - multiplex'g	0
	1	SSM suppression control	SSM instruction	0
	2	TOD clock sync control	Multiprocessing	0
	8 - 9	Page size control	Dynamic addr. transl.	0
	10	Unassigned (must be zero)		0
	11 - 12	Segment size control		0
	16	Malfunction alert mask	Multiprocessing	0
	17	Emergency signal mask		0
	18	External call mask		0
	19	TOD clock sync check mask		0
	20	Clock comparator mask	Clock comparator	0
	21	CPU timer mask	CPU timer	0
	24	Interval timer mask	Interval timer	1
	25	Interrupt key mask	Interrupt key	1
	26	External signal mask	External signal	1
1	0 - 7	Segment table length	Dynamic addr. transl.	0
	8 - 25	Segment table address		0
2	0 - 31	Channel masks	Channels	1
8	16 - 31	Monitor masks	Monitoring	0
9	0	Successful branching event mask	Program - event record'g	0
	1	Instruction fetching event mask		0
	2	Storage alteration event mask		0
	3	GR alteration event mask		
	16 - 31	PER general register masks		0
10	8 - 31	PER starting address	Program - event record'g	0
11	8 - 31	PER ending address	Program - event record'g	0
14	0	Check - stop control	Machine - check handling	1
	1	Synch. MCEL control		1
	2	I/O extended logout control	I/O extended logout	0
	4	Recovery report mask	Machine - check handling	0
	5	Degradation report mask		0
	6	Ext. damage report mask		1
	7	Warning mask		0
	8	Asynch. MCEL control		0
	9	Asynch. fixed log control		0
15	8 - 28	MCEL address	Machine - check handling	512

Figure 9. Control Registers

Condition Code Setting	**0**	**1**	**2**	**3**
Mask Bit Value	**8**	**4**	**2**	**1**
General Instructions				
Add, Add Halfword	zero	< zero	> zero	overflow
Add Logical	zero, no carry	not zero, no carry	zero, carry	not zero, carry
AND	zero	not zero	—	—
Compare, Compare Halfword	equal	1st op low	1st op high	—
Compare and Swap/Double	equal	not equal	—	—
Compare Logical	equal	1st op low	1st op high	—
Exclusive OR	zero	not zero	—	—
Insert Characters under Mask	all zero	1st bit one	1st bit zero	—
Load and Test	zero	< zero	> zero	—
Load Complement	zero	< zero	> zero	overflow
Load Negative	zero	< zero	—	—
Load Positive	zero	—	> zero	overflow
Move Long	count equal	count low	count high	overlap
OR	zero	not zero	—	—
Shift Left Double/Single	zero	< zero	> zero	overflow
Shift Right Double/Single	zero	< zero	> zero	—
Store Clock	set	not set	error	not oper
Subtract, Subtract Halfword	zero	< zero	> zero	overflow
Subtract Logical	—	not zero, no carry	zero, carry	not zero, carry
Test and Set	zero	one	—	—
Test under Mask	zero	mixed	—	ones
Translate and Test	zero	incomplete	complete	—
Decimal Instructions				
Add Decimal	zero	< zero	> zero	overflow
Compare Decimal	equal	1st op low	1st op high	—
Edit, Edit and Mark	zero	< zero	> zero	—
Shift and Round Decimal	zero	< zero	> zero	overflow
Subtract Decimal	zero	< zero	> zero	overflow
Zero and Add	zero	< zero	> zero	overflow
Floating-Point Instructions				
Add Normalized	zero	< zero	> zero	—
Add Unnormalized	zero	< zero	> zero	—
Compare	equal	1st op low	1st op high	—
Load and Test	zero	< zero	> zero	—
Load Complement	zero	< zero	> zero	—
Load Negative	zero	< zero	—	—
Load Positive	zero	—	> zero	—
Subtract Normalized	zero	< zero	> zero	—
Subtract Unnormalized	zero	< zero	> zero	—
Input/Output Instructions				
Clear I/O	no oper in progress	CSW stored	chan busy	not oper
Halt Device	interruption pending	CSW stored	channel working	not oper
Halt I/O	interruption pending	CSW stored	burst op stopped	not oper
Start I/O, SIOF	successful	CSW stored	busy	not oper
Store Channel ID	ID stored	CSW stored	busy	not oper
Test Channel	available	interruption pending	burst mode	not oper
Test I/O	available	CSW stored	busy	not oper
System Control Instructions				
Load Real Address	translation available	ST entry invalid	PT entry invalid	length violation
Reset Reference Bit	R=0, C=0	R=0, C=1	R=1, C=0	R=1, C=1
Set Clock	set	secure	—	not oper
Signal Processor	accepted	stat stored	busy	not oper

Figure 10. Condition Codes

PROGRAM INTERRUPTION CODES

<table>
<tr><th colspan="2">Interruption Code</th><th rowspan="2">Program Interruption Cause</th><th colspan="2">Interruption Code</th><th rowspan="2">Program Interruption Cause</th></tr>
<tr><th>Dec</th><th>Hex</th><th>Dec</th><th>Hex</th></tr>
<tr><td>1</td><td>0001</td><td>Operation</td><td>12</td><td>000C</td><td>Exponent overflow</td></tr>
<tr><td>2</td><td>0002</td><td>Privileged operation</td><td>13</td><td>000D</td><td>Exponent underflow</td></tr>
<tr><td>3</td><td>0003</td><td>Execute</td><td>14</td><td>000E</td><td>Significance</td></tr>
<tr><td>4</td><td>0004</td><td>Protection</td><td>15</td><td>000F</td><td>Floating-point divide</td></tr>
<tr><td>5</td><td>0005</td><td>Addressing</td><td>16</td><td>0010</td><td>Segment translation</td></tr>
<tr><td>6</td><td>0006</td><td>Specification</td><td>17</td><td>0011</td><td>Page translation</td></tr>
<tr><td>7</td><td>0007</td><td>Data</td><td>18</td><td>0012</td><td>Translation specification</td></tr>
<tr><td>8</td><td>0008</td><td>Fixed-point overflow</td><td>19</td><td>0013</td><td>Special operation</td></tr>
<tr><td>9</td><td>0009</td><td>Fixed-point divide</td><td>64</td><td>0040</td><td>Monitor event</td></tr>
<tr><td>10</td><td>000A</td><td>Decimal overflow</td><td>128</td><td>0080</td><td>Program event (code may be combined with another code)</td></tr>
<tr><td>11</td><td>000B</td><td>Decimal divide</td><td></td><td></td><td></td></tr>
</table>

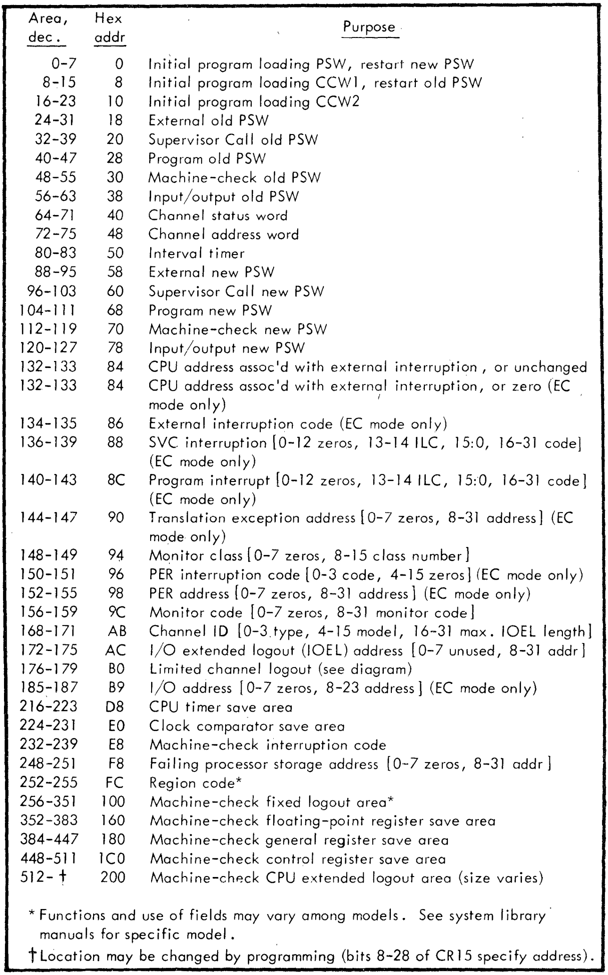
CNOP ALIGNMENT

<table>
<tr><td colspan="8">Double Word</td></tr>
<tr><td colspan="4">Word</td><td colspan="4">Word</td></tr>
<tr><td colspan="2">Half Word</td><td colspan="2">Half Word</td><td colspan="2">Half Word</td><td colspan="2">Half Word</td></tr>
<tr><td>Byte</td><td>Byte</td><td>Byte</td><td>Byte</td><td>Byte</td><td>Byte</td><td>Byte</td><td>Byte</td></tr>
<tr><td>0,4
0,8</td><td></td><td>2,4
2,8</td><td></td><td>0,4
4,8</td><td></td><td>2,4
6,8</td><td></td></tr>
</table>

EDIT AND EDMK PATTERN CHARACTERS (in hex)

20 - digit selector	40 - blank	5C - asterisk
21 - start of significance	4B - period	6B - comma
22 - field separator	5B - dollar sign	C3D9 - CR

Figure 11. Program Interrupt Codes, CNOP Alignment, and Edit and EDMK Pattern Characters



Area, dec.	Hex addr	Purpose
0-7	0	Initial program loading PSW, restart new PSW
8-15	8	Initial program loading CCW1, restart old PSW
16-23	10	Initial program loading CCW2
24-31	18	External old PSW
32-39	20	Supervisor Call old PSW
40-47	28	Program old PSW
48-55	30	Machine-check old PSW
56-63	38	Input/output old PSW
64-71	40	Channel status word
72-75	48	Channel address word
80-83	50	Interval timer
88-95	58	External new PSW
96-103	60	Supervisor Call new PSW
104-111	68	Program new PSW
112-119	70	Machine-check new PSW
120-127	78	Input/output new PSW
132-133	84	CPU address assoc'd with external interruption , or unchanged
132-133	84	CPU address assoc'd with external interruption, or zero (EC mode only)
134-135	86	External interruption code (EC mode only)
136-139	88	SVC interruption [0-12 zeros, 13-14 ILC, 15:0, 16-31 code] (EC mode only)
140-143	8C	Program interrupt [0-12 zeros, 13-14 ILC, 15:0, 16-31 code] (EC mode only)
144-147	90	Translation exception address [0-7 zeros, 8-31 address] (EC mode only)
148-149	94	Monitor class [0-7 zeros, 8-15 class number]
150-151	96	PER interruption code [0-3 code, 4-15 zeros] (EC mode only)
152-155	98	PER address [0-7 zeros, 8-31 address] (EC mode only)
156-159	9C	Monitor code [0-7 zeros, 8-31 monitor code]
168-171	AB	Channel ID [0-3 type, 4-15 model, 16-31 max. IOEL length]
172-175	AC	I/O extended logout (IOEL) address [0-7 unused, 8-31 addr]
176-179	B0	Limited channel logout (see diagram)
185-187	B9	I/O address [0-7 zeros, 8-23 address] (EC mode only)
216-223	D8	CPU timer save area
224-231	E0	Clock comparator save area
232-239	E8	Machine-check interruption code
248-251	F8	Failing processor storage address [0-7 zeros, 8-31 addr]
252-255	FC	Region code*
256-351	100	Machine-check fixed logout area*
352-383	160	Machine-check floating-point register save area
384-447	180	Machine-check general register save area
448-511	1C0	Machine-check control register save area
512- †	200	Machine-check CPU extended logout area (size varies)

* Functions and use of fields may vary among models. See system library manuals for specific model.

† Location may be changed by programming (bits 8-28 of CR15 specify address).

Figure 12. Fixed Storage Locations

PROGRAM STATUS WORD (BC Mode)

Channel Masks	E	Protect'n Key	CMWP	Interruption Code
0 6	7	8 11	12 15	16 23 24 31

ILC	CC	Program Mask	Instruction Address
32	34	36 39	40 47 48 55 56 63

0-5 Channel 0 to 5 masks
6 Mask for channel 6 and up
7 (E) External mask
12 (C=0) Basic control mode
13 (M) Machine-check mask
14 (W=1) Wait state
15 (P=1) Problem state

32-33 (ILC) Instruction length code
34-35 (CC) Condition code
36 Fixed-point overflow mask
37 Decimal overflow mask
38 Exponent underflow mask
39 Significance mask

PROGRAM STATUS WORD (EC Mode)

0R00 0TIE	Protect'n Key	CMWP	00	CC	Program Mask	0000 0000
0 7	8 11	12 15	16	18	20 23	24 31

0000 0000	Instruction Address
32 39	40 47 48 55 56 63

1 (R) Program event recording mask
5 (T=1) Translation mode
6 (I) Input/output mask
7 (E) External mask
12 (C=1) Extended control mode
13 (M) Machine-check mask
14 (W=1) Wait state

15 (P=1) Problem state
18-19 (CC) Condition code
20 Fixed-point overflow mask
21 Decimal overflow mask
22 Exponent underflow mask
23 Significance mask

Figure 13. Program Status Words (PSW), BC and EC Modes

CHANNEL ADDRESS WORD (hex 48)

Key	0000	Command Address
0 3	4 7	8 15 16 23 24 31

CHANNEL COMMAND WORD

Command Code	Data Address
0 7	8 15 16 23 24 31

Flags	00	▨	Byte Count
32 37	38 39	40 47	48 55 56 63

CD—bit 32 (80) causes use of address portion of next CCW.
CC—bit 33 (40) causes use of command code and data address of next CCW.
SLI—bit 34 (20) causes suppression of possible incorrect length indication.
Skip—bit 35 (10) suppresses transfer of information to main storage.
PCI—bit 36 (08) causes a channel program controlled interruption.
IDA—bit 37 (04) causes bits 8-31 of CCW to specify location of first IDAW.

CHANNEL STATUS WORD (hex 40)

Key	0	L	CC	CCW Address
0 3	4	5	6 7	8 15 16 23 24 31

Unit Status	Channel Status	Byte Count
32 39	40 47	48 55 56 63

5 Logout pending
6-7 Deferred condition code
32 (8000) Attention
33 (4000) Status modifier
34 (2000) Control unit end
35 (1000) Busy
36 (0800) Channel end
37 (0400) Device end
38 (0200) Unit check
39 (0100) Unit exception

40 (0080) Program control interruption
41 (0040) Incorrect length
42 (0020) Program check
43 (0010) Protection check
44 (0008) Channel data check
45 (0004) Channel control check
46 (0002) Interface control check
47 (0001) Chaining check
48-63 Residual byte count for the last CCW used

Figure 14. Channel Address Word (CAW), Channel Command Word (CCW), and Channel Status Word (CSW)

LIMITED CHANNEL LOGOUT (hex B0)

0	SCU id	Detect	Source	000	Field validity flags	TT	00	A	Seq.
0	1 3	4 7	8 12	13 15	16 23	24	26	28	29 31

Detect field
4 CPU
5 Channel
6 Storage control unit
7 Storage unit

Source field
8 CPU
9 Channel
10 Storage control unit
11 Storage unit
12 Control unit

16-23 Field validity flags
16 Interface address
17-18 Reserved (00)
19 Sequence code
20 Unit status
21 Command address and key
22 ,Channel address
23 Device address

24-25 (TT) Type of termination
Code 00 Interface disconnect
01 Stop, stack, or normal
10 Selective reset
11 System reset

28 (A) I/O error alert

29-31 Sequence code

Figure 15. Limited Channel Logout

MACHINE-CHECK INTERRUPTION CODE (hex E8)

MC conditions	000 00	Time	Stg. error	0	Validity indicators
0 8	9 13	14	16 18	19	20 31

0000 0000	0000 00	Val.	MCEL length
32 39	40 45	46	48 55 56 63

0 System damage
1 Instr. proc'g damage
2 System recovery
3 Timer damage
4 Timing facil. damage
5 External damage
6 Not assigned (0)
7 Degradation
8 Warning

14 Backed-up
15 Delayed
16 Uncorrected
17 Corrected
18 Key uncorrected
20 PSW bits 12-15
21 PSW masks and key
22 Prog. mask and CC
23 Instruction address

24 Failing stg. address
25 Region code
27 Floating-pt registers
28 General registers
29 Control registers
30 CPU ext'd logout
31 Storage logical
46 CPU timer
47 Clock comparator

Figure 16. Machine Check Interrupt Code

Standard Command Code Assignments (CCW bits 0-7)

xxxx	0000	Invalid	††††	††01	Write
††††	0100	Sense	††††	††10	Read
xxxx	1000	Transfer in Channel	††††	††11	Control
††††	1100	Read Backward	0000	0011	Control No Operation

x -Bit ignored.   †Modifier bit for specific type of I/O device

CONSOLE PRINTERS

Write, No Carrier Return	01	Sense	04
Write, Auto Carrier Return	09	Audible Alarm	0B
Read Inquiry	0A		

3504, 3505 CARD READER/3525 CARD PUNCH   (GA21-9124)

Command	Binary	Hex	Bit Meanings	
Sense	0000 0100	04	SS	Stacker
Feed, Select Stacker	SS10 F011		00	1
Read Only*	11D0 F010		01	2
Diagnostic Read	1101 0010	D2	10	2
Read, Feed, Select Stacker*	SSD0 F010		F	Format Mode
Write RCE Format*	0001 0001	11	0	Unformatted
3504, 3505 only			1	Formatted
Write OMR Format†	0011 0001	31	D	Data Mode
			0	1-EBCDIC
3525 only			1	2-Card image
Write, Feed, Select Stacker	SSD0 0001			
Print Line*	LLLL L101		L	Line Position
			5-bit binary value	

*Special feature on 3525   †Special feature

PRINTERS: 3211/3811 (GA24-3543), 3203/IPA, 1403/2821* (GA24-3312)

	After			
	Write	Immed		
Space 1 Line	09	0B	Write without spacing	01
Space 2 Lines	11	13	Sense	04
Space 3 Lines	19	1B	Load UCSB without folding	FB
Skip to Channel 0†	-	83	Fold†	43
Skip to Channel 1	89	8B	Unfold†	23
Skip to Channel 2	91	93	Load UCSB and Fold (exc. 3211)	F3
Skip to Channel 3	99	9B	UCS Gate Load (1403 only)	EB
Skip to Channel 4	A1	A3	Load FCB†	63
Skip to Channel 5	A9	AB	Block Data Check	73
Skip to Channel 6	B1	B3	Allow Data Check	7B
Skip to Channel 7	B9	BB	Read PLB†	02
Skip to Channel 8	C1	C3	Read UCSB†	0A
Skip to Channel 9	C9	CB	Read FCB†	12
Skip to Channel 10	D1	D3	Diag. Check Read (exc. 3203)	06
Skip to Channel 11	D9	DB	Diagnostic Write†	05
Skip to Channel 12	E1	E3	Raise Cover†	6B
			Diagnostic Gate†	07
			Diagnostic Read (1403 only)	02

*1403/IPA diagnostics are model-dependent;   †3211 only
UCS special feature on 1403

Figure 17. I/O Command Code (Part 1 of 4)

3420/3803, 3410/3411 MAGNETIC TAPE
See GA32-0020, -0021, -0022 for function of specific models and special features required.

Write	01
Read Forward	02
Read Backward	0C
Sense	04
Sense Reserve*†	F4
Sense Release*†	D4
Request Track-in-Error	1B
Loop Write-to-Read†	8B
See Diagnose†	4B
Rewind	07
Rewind Unload	0F
Erase Gap	17
Write Tape Mark	1F
Backspace Block	27
Backspace File	2F
Forward Space Block	37
Forward Space File	3F
Data Security Erase†	97
Diagnostic Mode Set†	0B

	Density	Parity	DC	Trans	Cmd
Mode Set 1 (7-track)	200	odd	on	off	13
			off	off	33
				on	3B
		even	off	off	23
				on	2B
	556	odd	on	off	53
			off	off	73
				on	7B
		even	off	off	63
				on	6B
	800	odd	on	off	93
			off	off	B3
				on	BB
		even	off	off	A3
				on	AB
Mode Set 2 (9 track)					
800 bpi					CB
1600 bpi					C3
6250 bpi†					D3

*Two-channel switch required †3420 only

DIRECT ACCESS STORAGE DEVICES:
3330-3340 SERIES (GA26-1592, -1617, -1619, -1620);
2305/2835 (GA26-1589); 2314, 2319 (GA26-3599, -1606)

Command		MT Off	MT On*	Count
Control	Orient (c)	2B		Nonzero
	Recalibrate	13		Nonzero
	Seek	07		6
	Seek Cylinder	0B		6
	Seek Head	1B		6
	Space Count	0F		3 (a); nonzero (d)
	Set File Mask	1F		1
	Set Sector (a,f)	23		1
	Restore (executes as a no-op)	17		Nonzero
	Vary Sensing (c)	27		1
	Diagnostic Load (a)	53		1
	Diagnostic Write (a)	73		512
Search	Home Address Equal	39	B9	4
	Identifier Equal	31	B1	5
	Identifier High	51	D1	5
	Identifier Equal or High	71	F1	5
	Key Equal	29	A9	KL
	Key High	49	C9	KL
	Key Equal or High	69	E9	KL
	Key and Data Equal (d)	2D	AD	Number of bytes (including mask bytes) in search argument
	Key and Data High (d)	4D	CD	
	Key and Data Eq. or Hi (d)	6D	ED	
Continue Scan	Search Equal (d)	25	A5	
	Search High (d)	45	C5	
	Search High or Equal (d)	65	E5	
	Set Compare (d)	35	B5	
	Set Compare (d)	75	F5	
	No Compare (d)	55	D5	

* Code same as MT Off except as listed.
a. Except 2314, 2319
b. 3330-3340 Series only; manual reset on 3340.
c. 2304/2835 only.
d. 2314, 2319 only.
e. String switch or 2-channel switch feature required; standard on 2314 and 2844.
f. Special feature required on 3340.

Figure 17. I/O Command Code (Part 2 of 4)

DIRECT ACCESS STORAGE DEVICES: (cont'd)
3330-3340 SERIES (GA26-1592, -1617, -1619, 1620);
2305/2835 (GA26-1589); 2314, 2319 (GA26-3599, -1606)

Command		MT Off	MT On*	Count
Read	Home Address	1A	9A	5
	Count	12	92	8
	Record 0	16	96	Number of bytes to be transferred
	Data	06	86	
	Key and Data	0E	8E	
	Count, Key and Data	1E	9E	
	IPL	02		
	Sector (a,f)	22		1
Sense	Sense I/O	04		24 (a); 6 (d)
	Read, Reset Buffered Log (b)	A4		24
	Read Buffered Log (c)	24		128
	Device Release (e)	94		24 (a); 6 (d)
	Device Reserve (e)	B4		24 (a); 6 (d)
	Read Diagnostic Status 1 (a)	44		16 or 512
Write	Home Address	19		5 (exc. 7 on 3340)
	Record 0	15		8+KL+DL of R0
	Erase	11		8+KL+DL
	Count, Key and Data	1D		8+KL+DL
	Special Count, Key and Data	01		8+KL+DL
	Data	05		DL
	Key and Data	0D		KL+DL

* Code same as MT Off except as listed.

a. Except 2314, 2319.

b. 3330-3340 Series only; manual reset on 3340.

c. 2304/2835 only.

d. 2314, 2319 only.

e. String switch or 2-channel switch feature required; standard on 2314 and 2844.

f. Special feature required on 3340.

Figure 17. I/O Command Code (Part 3 of 4)

Device	Command for CCW		8-Bit Code							Hex	Dec
		0	1	2	3	4	5	6	7		
1052	Read Inquiry BCD	0	0	0	0	1	0	1	0	0A	10
	Read Reader 2 BCD	0	0	0	0	0	0	1	0	02	02
	Write BDC, Auto Carriage Return	0	0	0	0	1	0	0	1	09	09
	Write BDC, No Auto Carriage Return	0	0	0	0	0	0	0	1	01	01
	No Op	0	0	0	0	0	0	1	1	03	03
	Sense	0	0	0	0	0	1	0	0	04	04
	Alarm	0	0	0	0	1	0	1	1	0B	11
2540	Read, Feed, Select Stacker SS Type AA	S	S	D	0	0	0	1	0		
	Read Type AB	1	1	D	0	0	0	1	0		
	Read, Feed (1400 compatability mode only)	1	1	D	1	0	0	1	0		
	Feed, Select Stacker SS Type BA	S	S	1	0	0	0	1	1		
	PFR Punch, Feed, Select Stacker SS Type BA	S	S	D	0	1	0	0	1		
	Punch, Feed, Select Stacker SS Type BB	S	S	D	0	0	0	0	1		
2400 Tape*	Read Backward (Overrides Data Converter On)	0	0	0	0	1	1	0	0	0C	12
	Sense	0	0	0	0	0	1	0	0	04	04
	Write	0	0	0	0	0	0	0	1	01	01
	Read	0	0	0	0	0	0	1	0	02	02
	Control	0	0	C	C	C	1	1	1		
		D	D	M	M	M	0	1	1		
		1	1	N	N	N	0	1	1		

SS	Stacker
00	R1
01	R2
10	RP3

D	Data Mode
0	EBCDIC
1	Column Binary

N	N	N	
0	0	0	1600 bpi P.E.**
0	0	1	800 bpi NRZ1

C	C	C	Control Codes	Hex	Dec
0	0	0	REW	7	7
0	0	1	RUN	0F	15
0	1	0	ERG	17	23
0	1	1	WTM	1F	31
1	0	0	BSR	27	39
1	0	1	BSF	2F	47
1	1	0	FSR	37	55
1	1	1	FSF	3F	63

D	D	7 Track Density	
0	0	200	7 Track
0	1	556	
1	0	800**	
1	1	***	

M	M	M	(Mode Modifiers)	Set Density	Set Odd Parity	Set Even Parity	Data Converter On	Data Converter Off	Translator On	Translator Off	Request TIE (Track in Error)
0	0	0	No Op								
0	0	1	Not Used								
0	1	0	Reset Condition	X	X		X			X	
0	1	1	Nine-track only								X
1	0	0		X		X		X		X	
1	0	1		X		X		X	X		
1	1	0	Reset Condition	X	X			X		X	
1	1	1		X	X			X	X		

* 9 track op. forces 800 BPI and odd parity; also, it overrides 7 track but does not reset 7 track. Load/Sys Reset forces 7 track to 800 BPI, odd parity, data converter on, translator off.

** Reset condition

*** Set 9 Track mode, Models 4-6

Figure 17. I/O Command Code (Part 4 of 4)

Code	Action Before Printing a Line
b̸	Space one line (blank code)
0	Space two lines
-	Space three lines
+	Suppress space
1	Skip to channel 1
2	Skip to channel 2
3	Skip to channel 3
4	Skip to channel 4
5	Skip to channel 5
6	Skip to channel 6
7	Skip to channel 7
8	Skip to channel 8
9	Skip to channel 9
A	Skip to channel 10
B	Skip to channel 11
C	Skip to channel 12

Code	Action After Punching a Card
V	Select punch pocket 1
W	Select punch pocket 2

Figure 18. ANSI Control Characters

Figure 19. System/370 Instructions (Part 1 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Add	A	5A	RX	R1, D2(X2,B2)	Add opr 2 to opr 1 (Sto) (Reg)	Addr Specif Fxpt Oflo	0 Sum = 0 1 Sum < 0 2 Sum > 0 3 Overflow
Add	AR	1A	RR	R1, R2	Add opr 2 to opr 1 (GPR) (Reg)	Fxpt Oflo	0 Sum = 0 1 Sum < 0 2 Sum > 0 3 Overflow
Add Decimal	AP	FA	SS	D1(L1,B1), D2(L2,B2)	Add dec opr 2 to opr 1 (Sto) (Sto) (Right to left byte by byte). (Opr 1 and 2 must be in packed) (Fields can overlap if low-order bytes coincide) (If opr 1 and opr 2 refer to same field, the field is doubled)	Addr Data Dec Oflo Protect Opera	0 Sum = 0 1 Sum < 0 2 Sum > 0 3 Overflow
Add Halfword	AH	4A	RX	R1, D2(X2,B2)	Add opr 2 to opr 1 (Sto) (Reg) (High-order 16 bits expanded) opr 2	Addr Fxpt Oflo Specif	0 Sum = 0 1 Sum < 0 2 Sum > 0 3 Overflow
Add Logical	AL	5E	RX	R1, D2(X2,B2)	Add log opr 2 to opr 1 (Sto) (Reg)	Addr Specif	0 Sum = 0 1 Sum ≠ 0 2 Sum = 0 3 Sum ≠ 0
Add Logical	ALR	1E	RR	R1, R2	Add log opr 2 to opr 1 (Reg) (Reg)	None	0 Sum = 0 1 Sum ≠ 0 2 Sum = 0 3 Sum ≠ 0

Figure 19. System/370 Instructions (Part 2 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Add Normalized (Extended)	AXR	36	RR	R1, R2	FP Add opr 2 to opr 1 (FPR pair) (FPR pair) Extended sum is put in opr 1 (FPR pair) Each operand consists of two FPR Only FPR 0 and FPR 4 may be specified for opr 1 or opr 2.	Specif Exp Oflo Exp Uflo Signif Opera	0 Fract = 0 1 Result < 0 2 Result >0
Add Normalized (Long)	AD	6A	RX	R1, D2(X2,B2)	FP Add opr 2 to opr 1 (Sto) (FPR) S Char Fraction 0 1 7 8 63	Addr Specif Signif Exp Oflo Exp Uflo Opera	0 Fract = 0 1 Result < 0 2 Result > 0
Add Normalized (Long)	ADR	2A	RR	R1, R2	FP Add opr 2 to opr 1 (FPR) (FPR)	Specif Opera Signif Exp Oflo Exp Uflo	0 Fract = 0 1 Result < 0 2 Result > 0
Add Normalized (Short)	AE	7A	RX	R1, D2(X2,B2)	FP Add opr 2 to opr 1 (Sto) (FPR) S Char Fraction 0 1 7 8 31 (Low-order halves of FPR ignored and unchanged)	Addr Specif Signif Exp Oflo Exp Uflo	0 Fract = 0 1 Result < 0 2 Result > 0
Add Normalized (Short)	AER	3A	RR	R1, R2	FP Add opr 2 to opr 1 (FPR) (FPR) (Low-order halves of FPR ignored and unchanged)	Specif Signif Exp Oflo Exp Uflo	0 Fract = 0 1 Result < 0 2 Result > 0

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Add Unnormalized (Long)	AW	6E	RX	R1, D2(X2,B2)	FP Add opr 2 to opr 1 (Sto) (FPR)	Addr Specif Signif Exp Oflo Opera	0 Fract = 0 1 Result < 0 2 Result > 0
Add Unnormalized (Long)	AWR	2E	RR	R1, R2	FP Add opr 2 to opr 1 (FPR) (FPR)	Specif Signif Exp Oflo Opera	0 Fract = 0 1 Result < 0 2 Result > 0
Add Unnormalized (Short)	AU	7E	RX	R1, D2(X2,B2)	FP Add opr 2 to opr 1 (Sto) (FPR) (Low-order halves of FPR ignored and unchanged)	Addr Specif Signif Exp Oflo Opera	0 Fract = 0 1 Result < 0 2 Result > 0
Add Unnormalized (Short)	AUR	3E	RR	R1, R2	FP Add opr 2 to opr 1 (FPR) (FPR) (Low-order halves of FPR ignored and unchanged)	Specif Signif Exp Oflo Opera	0 Fract = 0 1 Result < 0 2 Result > 0
AND	N	54	RX	R1, D2(X2,B2)	Place the product of both opr s into opr 1	Addr Specif	0 Result = 0 1 Result ≠ 0
AND	NC	D4	SS	D1(L,B1), D2(B2)	Place the product of both opr's into opr 1 (Left to right byte by byte) (Max number of bytes ANDed: 256)	Addr Protect	0 Result = 0 1 Result ≠ 0
AND	NR	14	RR	R1, R2	Place the product of both opr's into opr 1	None	0 Result = 0 1 Result ≠ 0
AND	NI	94	SI	D1(B1), I2	AND the 1 byte from the instruction stream (8-15) to opr 1	Addr Protect	0 Result = 0 1 Result ≠ 0

Figure 19. System/370 Instructions (Part 3 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Branch and Link	BAL	45	RX	R1, D2(X2,B2)	Store ILC,CC prog mask, and 24 bits of inst adr in opr 1. Branch to adr of opr 2	None	Unchanged
Branch and Link	BALR	05	RR	R1, R2	Store ILC,CC prog mask, and 24 bits of inst adr in opr 1. Branch to adr of opr 2 (If opr 2 = 0, store, no branch)	None	Unchanged
Branch on Condition	BC	47	RX	M1, D2(X2,B2)	Compare opr 1 with cond code (Mask) 8-11 (Mask = 7) Branch on non-zero cond code (Mask = 15) Uncond branch (Mask = 8) Cond code 00 (Mask = 4) Cond code 01 (Mask = 2) Cond code 10 (Mask = 1) Cond code 11 (NOP if cond not met)	None	Unchanged
Branch on Condition	BCR	07	RR	M1, R2	Compare opr 1 with cond code Branch to opr 2 adr if cond met (If opr 2 = 0, NOP	None	Unchanged
Branch on Count	BCT	46	RX	R1, D2(X2,B2)	Reduce opr 1 by 1 and branch to opr 2 adr (If opr 1 = 1) Reduce, no branch	None	Unchanged
Branch on Count	BCTR	06	RR	R1, R2	Reduce opr 1 by 1 and branch to opr 2 adr (If opr 1 = 1) Reduce, no branch (If opr 2 = 0) Reduce, no branch	None	Unchanged
Branch on Equal	BE	47(BC 8)	RX,Ext.	D2(X2,B2)	Branch if mask = cond code	None	Unchanged
Branch on Equal	BER	07(BCR 8)	RR,Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on High	BH	47(BC 2)	RX,Ext.	D2(X2,B2)	Branch if mask = cond code	None	Unchanged
Branch on High	BHR	07(BCR 2)	RR,Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Index High	BXH	86	RS	R1, R3, D2(B2)	Add opr 3 to opr 1 Sum compared to opr 3 if opr 3 adr is odd Sum compared to opr 3 + 1 if opr 3 addr is even. Branch to opr 2 addr if sum > 3/opr 3 + 1	None	Unchanged

Figure 19. System/370 Instructions (Part 4 of 33)

Figure 19. System/370 Instructions (Part 5 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Branch on Index Low or Equal	BXLE	87	RS	R1, R3, D2(B2)	Same as Branch On Index High Branch to opr 2 adr if sum < or = opr 3+1	None	Unchanged
Branch on Low	BL	47(BC 4)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Low	BLR	07(BCR4)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch if Mixed	BM	47(BC 4)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch if Mixed	BMR	07(BCR 4)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Minus	BM	47(BC 4)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Minus	BMR	07(BCR 4)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Not Equal	BNE	47(BC 7)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Not Equal	BNER	07(BCR 7)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Not High	BNH	47(BC 13)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Not High	BNHR	07(BCR 13)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Not Low	BNL	47(BC 11)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Not Low	BNLR	07(BCR 11)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Not Minus	BNM	47(BC 11)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Not Minus	BNMR	07(BCR 11)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Not Ones	BNO	47(BC 14)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Not Ones	BNOR	07(BCR 14)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Not Plus	BNP	47(BC 13)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Not Plus	BNPR	07(BCR 13)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Not Zeros	BNZ	47(BC 7)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Not Zeros	BNZR	07(BCR 7)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged

Ext = Extended Mnemonic

Figure 19. System/370 Instructions (Part 6 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Branch if Ones	BO	47(BC 1)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch if Ones	BOR	07(BCR 1)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Overflow	BO	47(BC 1)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Overflow	BOR	07(BCR 1)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Plus	BP	47(BC 2)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Plus	BPR	07(BCR 2)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch if Zeros	BZ	47(BC 8)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch if Zeros	BZR	07(BCR8)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch on Zero	BZ	47(BC 8)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch on Zero	BZR	07(BCR 8)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Branch Unconditional	B	47(BC 15)	RX, Ext.	D2 (X2, B2)	Branch if mask = cond code	None	Unchanged
Branch Unconditional	BR	07(BC 15)	RR, Ext.	R2	Branch if mask = cond code	None	Unchanged
Clear I/O	CLRIO	9D01	S	D2 (B2)	Terminate execution of current I/O op at addressed dev.	Priv	0 opr's = 1 CSW stored 2 channel or subchannel busy 3 not oprtnal
Compare	C	59	RX	R1, D2(X2, B2)	Compare opr 1 algebraically to opr 2 (Reg)	Addr Specif	0 opr's = 1 1st < 2 1st >
Compare	CR	19	RR	R1, R2	Compare opr 1 algebraically to opr 2	None	0 opr's = 1 1st < 2 1st >

Ext = Extended Mnemonic

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Compare and Swap	CS	BA	RS	R1,R3,D2(B2)	Compare opr 1 to opr 2. Store opr 3 in opr 2 if =, store opr 2 in opr 1 if ≠.	Addr Specif Protect Opera	0 opr's = 1 1st ≠ 2nd: 2nd replaced by 3rd
Compare Decimal	CP	F9	SS	D1 (L1,B1), D2(L2,B2)	Compare opr 1 to opr 2 (binary right to left) byte by byte (Opr's must be packed) (Fields can overlap if low-order bytes coincide) (The shorter opr is extended with high-order zeros)	Addr Data Opera	0 opr's = 1 1st < 2 1st >
Compare Double and Swap	CDS	BB	RS	R1, R3, D2(B2)	Compare opr 1 to opr 2. Store opr 3 in opr 2 if =, store opr 2 in opr 1 if ≠.	Addr Specif Protect Opera	0 opr's = 1 1st ≠ 2nd 2nd replaced by 3rd
Compare Halfword	CH	49	RX	R1, D2(X2,B2)	Compare opr 1 algebraically to opr 2 (Hi-order 16 bits expanded) opr 2	Addr Specif	0 opr's = 1 1st < 2 1st >
Compare Logical	CL	55	RX	R1, D2(X2,B2)	Compare opr 1 to opr 2 (binary left to right) (Terminates if/when ≠ found)	Addr Specif	0 opr's = 1 1st < 2 1st >
Compare Logical	CLC	D5	SS	D1 (L,B1), D2(B2)	Compare opr 1 to opr 2 (binary left to right) (Terminated if/when ≠ found) (opr length max 256 bytes)	Addr Specif	0 opr's = 1 1st < 2 1st >
Compare Logical Immediate	CLI	95	SI	D1 (B1), I2	Compare opr 1 to opr 2 (Imm) (Sto) (binary left to right) (Terminates if/when ≠ found)	Addr	0 opr's = 1 1st < 2 1st >
Compare Logical	CLR	15	RR	R1, R2	Compare opr 1 to opr 2 (binary left to right) (Terminates if/when ≠ found)	Addr	0 opr's = 1 1st < 2 1st >
Compare Logical Characters Under Mask	CLM	BD	RS	R1, M3,D2(B2)	Compare opr 2 to opr 1 under control of mask (binary left to right)	Addr Protect Opera	0 Selected by bytes = or mask = 0 1 Selected field 1st opr is low 2 Selected Field 1st opr is high

Figure 19. System/370 Instructions (Part 7 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Compare Logical Long	CLCL	0F	RR	R1, R2	Compare opr 1 to opr 2 (opr 1 and 2 indicate even/odd reg. pair)	Addr Specif Opera Protect	0 opr's = 1 1st < 2 1st > 3 --
Compare (Long)	CD	69	RX	R1, D2(X2, B2)	Compare opr 1 algebraically to opr 2 (Equalize and subtract)	Addr Specif Opera	0 opr's = 1 1st < 2 1st >
Compare (Long)	CDR	29	RR	R1, R2	Compare opr 1 algebraically to opr 2 (FPR) (Equalize and subtract)	Specif Addr Opera	0 opr's = 1 1st < 2 1st >
Compare (Short)	CE	79	RX	R1, D2(X2, B2)	Compare opr 1 algebraically to opr 2 (FPR) (Sto) (Low-order halves of FPR ignored and unchanged)	Addr Specif Opera	0 opr's = 1 1st < 2 1st >
Compare (Short)	CER	39	RR	R1, R2	Compare opr 1 algebraically to opr 2 (FPR) (FPR) (Low-order halves of FPR ignored and unchanged)	Specif Opera	0 opr's = 1 1st < 2 1st >
Convert to Binary	CVB	4F	RX	R1, D2(X2, B2)	Convert opr 2 (packed decimal) (Doubleword bounds) to binary and put in opr 1 location	Addr Specif Data Fxpt Div	Unchanged
Convert to Decimal	CVD	4E	RX	R1, D2(X2, B2)	Convert opr 1 (binary) to packed decimal (doubleword bounds) and put in opr 2	Addr Specif Protect	Unchanged
Diagnose	----	83		See IBM System/370 Principles of Operation, GA22-7000	See IBM System/370 Principles of Operation, GA22-7000	Priv Oper Model dependent	Unpredictable

Figure 19. System/370 Instructions (Part 8 of 33)

Figure 19. System/370 Instructions (Part 9 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Divide	D	5D	RX	R1, D2 (X2,B2)	Divide opr 1 by opr 2 (even and odd regs) (Sto) Opr 1 becomes remainder and quotient	Addr Specif Fxpt Div	Unchanged
Divide	DR	1D	RR	R1, R2	Divide opr 1 by opr 2 Dividend: even and odd pair regs Opr 1 becomes remainder and quotient (full word only)	Specif Fxpt Div	Unchanged
Divide Decimal	DP	FD	SS	D1(L1,B1), D2(L2,B2)	Divide opr 1 by opr 2 Opr 1 becomes quotient and remainder (left justified) Dividend: at least 1 leading zero, max size 31 digits and sign Divisor: max size 15 digits and sign, numerically larger than dividend Both opr's packed format Remainder size = divisor size (Fields can overlap if low-order bytes coincide.)	Addr Protect Specif Data Dec Div Opera	Unchanged
Divide (Long)	DD	6D	RX	R1, D2(X2,B2)	FP Divide opr 1 by opr 2 (FPR) (Sto) Opr 1 becomes quotient (prenormalized)	Addr Specif Exp Oflo FP Div Opera Exp Uflo	Unchanged
Divide (Long)	DDR	2D	RR	R1, R2	FP Divideopr 1 by opr 2 Prenormalize (FPR) (FPR) (Dividend) (Divisor) Oper 1 becomes quotient	Specif Opera Exp Oflo Exp Uflo FP Div	Unchanged

Figure 19. System/370 Instructions (Part 10 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Divide (Short)	DE	7D	RX	R1, D2(X2,B2)	FP Divide opr 1 by opr 2 Prenormalize (Dividend) (Divisor) Opr 1 becomes quotient (Low-order halves of FPR ignored and unchanged)	Addr Specif Exp Oflo Exp Uflo FP Div Opera	Unchanged
Divide (Short)	DER	3D	RR	R1, R2	FP Divide opr 1 by 2 Prenormalize (FPR) (FPR) (Dividend) (Divisor) Opr 1 becomes quotient (Low-order halves of FPR ignored and unchanged)	Specif Exp Oflo FP Div Exp Uflo Opera	Unchanged
Edit	ED	DE	SS	D1(L,B1), D2(B2)	Opr 1 = pattern, opr 2 = source Opr 2 is changed from packed to zoned and edited under control of opr 1. Opr's processed left to right (Fill char is 1st char in pattern field unless it is a digit/select/significance-start char.) (Opr 1 terminates operation) See IBM System/370 Principles of Operation, GA22-7000	Addr Data Opera Protect	Source 0 field = 0 1 field < 0 2 field > 0
Edit and Mark	EDMK	DF	SS	D1(L,B1), D2(B2)	Same as Edit (Adr of 1st significant result digit recorded in GPR 1)	Opera Addr Data Protect	Source 0 field = 0 1 field < 0 2 field > 0
Exclusive OR	X	57	RX	R1, D2(X2,B2)	Exclusive-OR opr 2 and opr 1 and the modulo-two sum placed in opr 1	Addr Specif	0 Result = 0 1 Result ≠ 0

Figure 19. System/370 Instructions (Part 11 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Exclusive OR	XC	D7	SS	D1(L,B1), D2(B2)	Exclusive-OR opr 2 and opr 1 and modulo-two sum placed in opr 1.	Addr Protect	0 Result = 0 1 Result ≠ 0
Exclusive OR	XR	17	RR	R1, R2	Exclusive-OR opr 2 and opr 1 and modulo-two sum placed in opr 1.		0 Result = 0 1 Result ≠ 0
Exclusive OR Immediate	XI	97	SI	D1(B1), I2	Exclusive-OR opr 2 and opr 1 and modulo-two sum placed in opr 1.	Addr Protect	0 Result = 0 1 Result ≠ 1
Execute	EX	44	RX	R1, D2(X2,B2)	The instruction addressed by opr 2 is modified by opr 1 and executed.	Addr Exec Specif	May be set by this instruction
Halve, Long	HDR	24	RR	R1, R2	Opr 2 is divided by 2 and placed in opr 1.	Specif Opera	Unchanged
Halve, Short	HER	34	RR	R1, R2	Opr 2 is divided by 2 and placed in opr 1.	Specif Opera	Unchanged
Halt Device	HDV	9E01	S	D1(B1)	Execution of current I/O op at addressed dev is terminated (full op cd - 1001 1110 xxxx xxx1).	Priv	0 Subchan busy with another dev or int pending 1 CSW stored 2 Chan working with another device
Halt I/O	HIO	9E00	S	D1(B1)	Execution of current I/O op at addresses dev, subchan, and chan term (full op cd - 1001 1110 xxxx xxx0).	Priv	0 Chan or sub-chan not working 1 CSW stored 2 Burst oper terminated 3 Not operational
Insert Character	IC	43	RX	R1, D2(X2,B2)	Byte at opr 2 is inserted in bits 24-31 of reg at opr 1.	Addr	Unchanged
Insert Characters Under Mask	ICM	BF	RS	R1, M3, D2(B2)	1 to 4 bytes at opr 2 are inserted in reg at opr 1 under control of mask.	Addr Protect Opera	0 Selected bits or mask = 0 1 Leftmost bit of spec byte = 1 2 Leftmost bit of spec byte = 0

Figure 19. System/370 Instructions (Part 12 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Insert PSW Key	IPK	B208	S		Protection key of current PSW inserted into reg 2 bit pos 24-27. Bits 28-31 set to 0.	Priv	Unchanged
Insert Storage Key	ISK	09	RR	R1, R2	Opr 2, 8-20 fetches 7-bit sto key byte. 7-bit sto key is placed in opr 1, 24-30. Bits 0-23 unchanged, 31 set to zero. (opr 2, 0-7 and 21-27 ignored, 28-31 must = 0)	Priv Addr Specif Opera	Unchanged
Load	L	58	RX	R1, D2(X2,B2)	Load opr 2 into opr 1.	Addr Specif	Unchanged
Load	LR	18	RR	R1, R2	Opr 2 into opr 1.	None	Unchanged
Load Address	LA	41	RX	R1, D2(X2,B2)	Opr 2, 12-31 to opr 1, 8-31. Opr 1, 0-7 set to zero (no storage reference made)	None	Unchanged
Load and Test	LTR	12	RR	R1, R2	Opr 2 into opr 1 (When opr 1 and opr 2 specify same reg result is test without data transfer.)	None	0 Result = 0 1 Result < 0 2 Result > 0
Load and Test (Long)	LTDR	22	RR	R1, R2	Opr 2 into opr 1 (FPR) (FPR) (When opr 1 and opr 2 specify same reg result is test without data transfer.)	Specif Opera	0 Result fraction = 0 1 Result < 0 2 Result > 0
Load and Test (Short)	LTER	32	RR	R1, R2	Opr 2 into opr 1 (FPR) (FPR) (Low-order half of opr 1 unchanged) (When opr 1 and opr 2 specify same reg result is test without data transfer.)	Specif Opera	0 Result Fraction = 0 1 Result < 0 2 Result > 0
Load Complement	LCR	13	RR	R1, R2	2's complement of opr 2 into opr 1 (overflow when max negative number is complemented)	Fxpt Oflo	0 Result = Expt Uflo 1 Result < 0 2 Result > 0 3 Overflow

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Load Complement (Short)	LCER	33	RR	R1, R2	Opr 2 into opr 1 (FPR) (FPR) (Opr 1 sign inverted, low-order half unchanged) (Opr 2 unchanged)	Specif Opera	0 Result Fract = 0 1 Result<0 2 Result>0
Load Complement (Long)	LCDR	23	RR	R1, R2	Opr 2 into opr 1 (FPR) (FPR) (Opr 1 sign inverted, low-order half unchanged) (Opr 2 unchanged) (Low-order half of opr 1 unchanged)	Specif Opera	0 Result Fract = 0 1 Result<0 2 Result>0
Load Control	LCTL	B7	RS	R1, R3, D2(B2)	Cntl regs from opr 1 to opr 3 loaded with info starting at opr 2.	Addr Specif Priv Protect Opera	Unchanged
Load Halfword	LH	48	RX	R1, D2(X2,B2)	Opr 2 halfword expanded to fullword with sign bits, placed in opr 1 (High-order expanded)	Addr Specif	Unchanged
Load (Long)	LD	68	RX	R1, D2(X2,B2)	Opr 2 into opr 1 (Sto) (FPR)	Addr Specif Opera	Unchanged
Load (Long)	LDR	28	RR	R1, R2	Opr 2 into opr 1 (FPR) (FPR)	Specif Opera	Unchanged

Figure 19. System/370 Instructions (Part 13 of 33)

Figure 19. System/370 Instructions (Part 14 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Load Multiple	LM	98	RS	R1, R3, D2(B2)	Opr 2 into GPRs in ascending order Starting reg specified by opr 1, ending reg specified by opr 3 (Reg wrap-around possible)	Addr Specif	Unchanged
Load Negative	LNR	11	RR	R1, R2	2's complement of opr 2 into opr 1 (Reg) (Reg) (If opr 2 contains a (-) number or zero, the number is unchanged)	None	0 Result = 0 1 Result < 0
Load Negative (Long)	LNDR	21	RR	R1, R2	Opr 2 into opr 1 (FPR) (FPR) Opr 1 sign bit is 1 (negative) Opr 2 unchanged	Specif Opera	0 Result Fract = 0 1 Result < 0
Load Negative (Short)	LNER	31	RR	R1, R2	Opr 2 into opr 1 Opr 1 sign bit is 1 (negative) Opr 2 unchanged (Low-order half of opr 1 unchanged)	Specif Opera	0 Result Fract = 0 1 Result < 0
Load Positive	LPR	10	RR	R1, R2	Opr 2 into opr 1 (Negative numbers are complemented) (Overflow occurs when the max negative number is complemented)	Fxpt Oflo	0 Result = 0 2 Result > 0 3 Overflow
Load Positive (Long)	LPDR	20	RR	R1, R2	Opr 2 into opr 1 (FPR) (FPR) Opr 1 sign bit made a zero (positive) Opr 2 unchanged	Specif Opera	0 Result Fract = 0 1 Result < 0 2 Result > 0
Load Positive (Short)	LPER	30	RR	R1, R2	Opr 2 into opr 1 Opr 1 sign bit made a zero (positive) Opr 2 unchanged (Low-order half of opr 1 unchanged)	Specif Opera	0 Result Fract = 0 1 Result < 0 2 Result > 0

Figure 19. System/370 Instructions (Part 15 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Load PSW	LPSW	82	SI	D1 (B1)	Opr 1 into PSW (Opr 1 low-order 3 bit adr must = 0) (Instruction used to enter the problem or wait state)	Priv Addr Specif	Set according to new PSW bits 34 and 35
Load (Short)	LE	78	RX	R1, D2(X2, B2)	Opr 2 into opr 1 (Sto) (FPR) (Low-order half of opr 1 unchanged)	Addr Specif Opera	Unchanged
Load (Short)	LER	38	RR	R1, R2	Opr 2 into opr 1 (FPR) (FPR) (Low-order half of opr 1 unchanged)	Specif Opera	Unchanged
Load Real Address	LRA	B1	RX	R1, D2(X2, B2)	Real adr corresponding to opr 2 logical adr placed in opr 1.	Priv Addr Specif Opera	0 Translation available 1 Seg tbl entry invalid 2 Page tbl entry invalid 3 Seg or page tbl length violation
Load Rounded (Extended to Long)	LRDR	25	RR	R1, R2	Opr 2 is rounded from extended to long format and put in opr 1 (FPR pair) (FPR) Only FPR 0 and FPR 4 may be specified for opr 2.	Specif Exp Oflo Opera	Unchanged
Load Rounded (Long to Short)	LRER	35	RR	R1, R2	Opr 2 is rounded from long to short format and put into opr 1 (FPR) (FPR) Add an absolute 1 to opr 2, bit 32; carry will ripple left. Lower half of result FPR will remain unchanged.	Specif Exp Oflo Opera	Unchanged

Figure 19. System/370 Instructions (Part 16 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Monitor Call	MC	AF	SI	D1 (B1),I2	Causes program interrupt if monitor-mask bit in cont. reg 8 = appropriate monitor class specified in positions 12-15 of I2. Real storage locations 148 and 156 will zero, loc 149=I2, and loc. 157-159=D1 + contents to B1.	Monitor Specif	Unchanged
Move Characters	MVC	D2	SS	D1(L,B1),D2(B2)	Opr 2 to opr 1 (Left to right byte by byte) (Max number of bytes moved: 256) (No restriction on overlapping fields)	Addr Protect	Unchanged
Move Immediate	MVI	92	SI	D1(B1), I2	Move the 1 byte from the instruction stream (8-15) to opr 1.	Addr Protect	Unchanged
Move Long	MVCL	0E	RR	R1, R2	Move char from area spec in opr 2 to area spec in opr 1. Opr 2 is even/odd reg pair where R2 is "from adr", R2+1 bits 0-7 is padding char, and R2+1 bits 8-31 is length. Opr 1 is even/odd reg. pair where R1 is "to" addr, R1+1 bits 8-31 is length.	Addr Specif	0 Opr cnts = 1 Opr 1 cnt< opr 2 cnt 2 Opr 1 cnt> opr 2 cnt 3 No move due to destructive overlap.

Figure 19. System/370 Instructions (Part 17 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Move Numerics	MVN	D1	SS	D1(L,B1), D2(B2)	The 4 low-order bits of opr 2 bytes into the 4 low-order bits of opr 1 bytes. (Left to right byte by byte) (Max number of bytes moved: 256) (High-order bits of each byte of both opr's unchanged.) (No restriction on overlapping fields.)	Addr Protect	Unchanged
Move with Offset	MVO	F1	SS	D1 (L1,B1), D2 (L2,B2)	Opr 2 to the left of and adjacent to the low-order 4 bits of opr 1. (Right to left byte by byte) (Data can be packed, unpacked, or binary format) (No restriction on overlapping fields) (Processing terminated by high-order bit in opr 1) (If opr 2 field shorter than opr 1, insert leading zeros in opr 2.)	Addr Protect	Unchanged
Move Zones	MVZ	D3	SS	D1(L,B1), D2(B2)	The 4 high-order bits of opr 2 bytes into the 4 high-order bits of opr 1 bytes (Left to right byte by byte) (Max number of bytes moved: 256) (Low-order bits of each byte of both opr's unchanged.) (No restriction on overlapping fields)	Addr Protect	Unchanged
Multiply	M	5C	RX	R1, D2(X2,B2)	Multiply opr 1 by opr 2 Product: even and odd pair regs Opr 1 becomes the product. (Opr 1 must specify an even-numbered reg) (Sign bit extended to 1st significant product digit)	Addr Specif	Unchanged

Figure 19. System/370 Instructions (Part 18 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Multiply	MR	1C	RR	R1, R2	Multiply opr 1 by opr 2 Product: even and odd pair of regs Opr 1 becomes the product. (Opr 1 must specify an even-numbered reg) (Sign bit extended to 1st significant product digit)	Specif	Unchanged
Multiply (Extended)	MXR	26	RR	R1, R2	Multiply extended opr 1 by extended opr 2 (FPR pair) (FPR pair) Extended product is put in opr 1 (FPR pair) (Only FPR 0 and FPR 4 may be specified for either opr 1 or opr 2) (Low-order characteristic is made 14 < high-order characteristic except when the result would be > 0, then the low-order characteristic is made 128 > its correct value; sign of low-order characteristic remains the same as high-order characteristic)	Specif Exp Oflo Exp Uflo Opera	Unchanged
Multiply Decimal	MP	FC	SS	D1(L1,B1), D2(L2,B2)	Multiply opr 1 by opr 2 Multiplier: 8 bytes max size and shorter than the multiplicand. Multiplicand: must have high-order zeros equal to or greater than the size of the multiplier. (Both opr's in packed format) (Right to left byte by byte) Product. must contain at least 1 high-order zero.	Addr Specif Data Protect Opera	Unchanged

Figure 19. System/370 Instructions (Part 19 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Multiply Halfword	MH	4C	RX	R1, D2(X2,B2)	Multiply opr 1 by opr 2 (Opr 2 is expanded to a 32-bit integer) (Only the low-order 32 bits of the product, opr 1, are retained)	Addr Specif	Unchanged
Multiply (Long)	MD	6C	RX	R1, D2(X2,B2)	Multiply opr 1 by opr 2 (FPR) (Sto) Product: prenormalizes the opr's and post-normalizes the intermediate product. (If all fraction digits (15) = zero; the product, sign and char are made zero.) (The intermediate product fraction is truncated before left-shifting.)	Addr Specif Exp Oflo Exp Uflo Opera	Unchanged
Multiply (Long)	MDR	2C	RR	R1, R2	Multiply opr 1 by opr 2 (FPR) (FPR) Product: prenormalizes the opr's and post-normalizes the intermediate product. (If all fraction digits (15) = 0; the product sign and char are made zero.) (The intermediate product fraction is truncated before left-shifting.)	Specif Exp Oflo Exp Uflo Opera	Unchanged
Multiply (Long to Extended)	MXD	67	RX	R1, D2(X2,B2)	Multiply long opr 1 by long opr 2. (FPR) (Sto) Extended product is put in FPR pair specified by opr 1 (Only FPR 0 and FPR 4 may be specified for opr 1) (Signs of FPR pair are the same) (Can only use doubleword boundary in storage) (Continued)	Addr Specif Exp Oflo Exp Uflo Protect Opera	Unchanged

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Multiply (Long to Extended) (Cont'd)	MXD	67	RX	R1, D2(X2,B2)	(Low-order characteristic is made 14< high-order characteristic except when the result would be > 0, then the low-order characteristic is made 128 >its correct value; sign of low-order characteristic remains the same as high-order characteristics)		
Multiply (Long to Extended)	MXDR	27	RR	R1, R2	Multiply long opr 1 by long opr 2. (FPR) (FPR) Extended product is put in FPR pair specified by opr 1 (Only FPR 0 and FPR 4 may be specified for opr 1) (Signs of FPR pair are the same) (Low-order characteristic is made 14< high-order characteristic except when the result would be >0, then the low-order characteristic is made 128> its correct value; sign of low-order characteristic remains the same as the high-order characteristic)	Specif Exp Oflo Exp Uflo Opera	Unchanged
Multiply (Short)	ME	7C	RX	R1, D2(X2,B2)	Multiply opr 1 by opr 2 (FPR) (Sto) Product: prenormalizes the opr's and postnormalizes the intermediate product. (If all fraction digits (14) = 0; the product sign and char are made zero.) (The intermediate product fraction is truncated before left-shifting.) (The 2 low-order fraction digits of the product always = zero.)	Addr Specif Exp Oflo Exp Uflo Opera	Unchanged

Figure 19. System/370 Instructions (Part 20 of 33)

Figure 19. System/370 Instructions (Part 21 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Multiply (Short)	MER	3C	RR	R1, R2	Multiply opr 1 by opr 2 (FPR) (FPR) Product: prenormalizes the opr's and post-normalizes the intermediate product. (If all fraction digits (14) = 0; the product sign and char are made zero.) (The intermediate product fraction is truncated before left-shifting.)	Specif Exp Oflo Exp Uflo Opera	Unchanged
No Operation	NOP	47(BC 0)	RX, Ext.	D2(X2, B2)	Comp mask with cond code	None	Unchanged
No Operation	NOPR	07(BCR 0)	RR, Ext.	R2	Comp mask with cond code	None	Unchanged
OR Logical	O	56	RX	R1, D2(X2, B2)	The ORed sum of both opr's into opr 1	Addr Specif	0 Result = 0 1 Result ≠ 0
OR Logical	OC	D6	SS	D1(L, B1), D2(B2)	The ORed sum of both opr's into opr 1 (Left to right byte by byte) (Max number of bytes ORed: 256)	Addr Protect	0 Result = 0 1 Result ≠ 0
OR Logical	OR	16	RR	R1, R2	The ORed sum of both opr's into opr 1	None	0 Result = 0 1 Result ≠ 0
OR Logical Immediate	OI	96	SI	D1(B1), I2	OR the 1 byte from the instruction stream (8-15) to opr 1	Addr Protect	0 Result = 0 1 Result ≠ 0
Pack	PACK	F2	SS	D1(L1, B1), D2(L2, B2)	Change opr 2 from zoned to packed format and place into opr 1. (Right to left byte by byte) (No restriction on overlapping fields) (Opr 2 may be extended with hi-order zeros)	Addr Protect	Unchanged
Purge Translation Lookaside Buffer	PTLB	B20D	S	---	Invalidate current info in TLB.	Priv Opera	Unchanged

Ext. = Extended Mnemonic

Figure 19. System/370 Instructions (Part 22 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Read Direct	RDD	85	SI	D1(B1), I2	The 1 byte from the instruction stream (8-15) is placed on the signal-out, in a form of 8 timing pulses, along with a 9th pulse at the read-out line. The 8 bit lines at the direct-in lines are stored in 0 or 1.	Priv Addr Protect Opera	Unchanged
Reset Reference Bit	RRB	B213	S	D1(B1)	Set refence-bit=0 for 2048 byte block referenced by opr 1. CC indicates setting of ref and change bits prior to exec of this instruction.	Priv Opera	0 Ref = 0 Chg = 0 1 Ref = 0 Chg = 1 2 Ref = 1 Chg = 0 3 Ref = 1 Chg = 1
Set Clock	SCK	B204	S	D1(B1)	Replace curr val of TOD clock with eight bytes starting at opr 1.	Addr Specif Priv Protect Opera	0 Clock val set 1 Clock val secure 2 -- 3 Clock not oper
Set Clock Comparator	SCKC	B206	S	D1(B1)	Dblwd at opr 1 replaces curr value of clock comparator	Addr Priv Specif Protect Opera	Unchanged
Set CPU Timer	SPT	B208	S	D1(B1)	Dblwd at opr 1 replaces curr value of CPU timer.	Addr Priv Specif Protect Opera	Unchanged
Set Prefix	SPX	B210	S	D2(B2)	Prefix reg contents replaced by contents of bit pos 8-19 of word located by opr 2 address.	Specif Opera Priv	Unchanged

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Set Program Mask	SPM	04	RR	R1	Opr 1 (2-7) replaces the cond code and program mask bits of the current PSW (34-39) (Bits 0, 1 and 8-31 of opr 1 are ignored and unchanged.)	None	Set by bits 2 and 3
Set PSW Key From Address	SPKA	B20A	S	D1(B1)	Protection key of PSW replaced by bits 24-27 of the opr address.	Opera Priv	Unchanged
Set Storage Key	SSK	08	RR	R1, R2	Opr 1 (24-30) replaces the storage key specified by opr 2 (Opr 1 bits 0-23 and 31 are ignored) (Opr 2 bits 0-7 and 21-27 are ignored) (Bits 28-31 must be zero)	Addr Priv Specif Opera	Unchanged
Set System Mask	SSM	80	S	D1(B1)	Opr 1 (1 byte) replaces the system mask bits of the current PSW (0-7).	Priv Addr	Unchanged
Shift and Round Decimal	SRP	F0	SS	D1(L1,B1), D2(B2), I3	Shift opr 1 as specified by opr 2. If shift is right, round by factor in opr 3.	Protect Opera Addr Data Dec Oflo	0 Result = 0 1 Result < 0 2 Result > 0 3 Result Oflo
Shift Left Double Algebraic	SLDA	8F	RS	R1, D2(B2)	Opr 1 (even and odd regs) is shifted left the number of times equal to opr 2 (low-order 6 bits).	Specif Fxpt Oflo	0 Result = 0 1 Result < 0 2 Result > 0 3 Overflow
Shift Left Double Logical	SLDL	8D	RS	R1, D2(B2)	Opr 1 (even and odd regs) is shifted left the number of times equal to opr 2 (low-order 6 bits). (Hi-order bit participates in the shift)	Specif	Unchanged
Shift Left Single Algebraic	SLA	8B	RS	R1, D2(B2)	Opr 1 is shifted left the number of times equal to opr 2 (low-order 6 bits).	Fxpt Oflo	0 Result = 0 1 Result < 0 2 Result > 0 3 Overflow

Figure 19. System/370 Instructions (Part 23 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Shift Left Single Logical	SLL	89	RS	R1, D2(B2)	Opr 1 is shifted left the number of times equal to opr 2 (low-order 6 bits). (Hi-order bit participates in the shift)	None	Unchanged
Shift Right Double Algebraic	SRDA	8E	RS	R1, D2(B2)	Opr 1 (even and odd regs) is shifted right the number of times equal to opr 2 (Low-order 6 bits).	Specif	0 Result = 0 1 Result < 0 2 Result > 0
Shift Right Double Logical	SRDL	8C	RS	R1, D2(B2)	Opr 1 (even and odd regs) is shifted right the number of times equal to opr 2 (low-order 6 bits). (Hi-order bit participates in the shift)	Specif	Unchanged
Shift Right Single Algebraic	SRA	8A	RS	R1, D2(B2)	Opr 1 is shifted right the number of times equal to opr 2 (low-order 6 bits). (Shifting (+) numbers: vacated bits are replaced with zeros.) (Shifting (-) numbers: vacated bits are replaced with ones.)	None	0 Result = 0 1 Result < 0 2 Result > 0
Shift Right Single Logical	SRL	88	RS	R1, D2(B2)	Opr 1 is shifted right the number of times equal to opr 2 (low-order 6 bits). (Vacated bits are replaced with zeros) (Hi-order bit participates in the shift)	None	Unchanged
Signal Processor	SIGP	AE	RS	R1, R3, D2(B2)	An eight-bit order code (bits 24-31 of the second-operand address) is transmitted to the CPU designated by the processor address (bits 16-31) in the third operand.	Opera Priv	0 = Order code accepted 1 = Status stored 2 = Channel or subchannel busy 3 = Channel not operational
Start I/O	SIO	9C00	S	D1(B1)	Opr 1 (16-31) identifies the selected chan, ctl unit and I/O device to perform write, read, read bkwd, control or sense oper. The CAW at loc 48 is fetched, which locates the first CCW. The SIO is initiated providing the addressed chan, ctl unit and I/O device are available without pending interrupt errors. Exceptional conditions pending (Full op cd - 1001 1100 xxxx xxx0)	Priv	0 = I/O oper initiated and chan proceeding with operation. 1 = CSW stored 2 = Chan or subchannel busy 3 = Not operational

Figure 19. System/370 Instructions (Part 24 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Start I/O Fast Release	SIOF	9C01	S	D1(B1)	This instruction takes advantage of the block-multiplex channel, but is otherwise identical to SIO. (Full op cd - 1001 1100 xxxx xxx1).	Priv	Same as SIO
Store	ST	50	RX	R1, D2(X2,B2)	Opr 1 is stored into opr 2.	Addr Specif Protect	Unchanged
Store Channel ID	STIDC	B203	S	D1(B1)	Store opr 1 at loc 168 in main storage.	Priv Opera	0 ID stored 1 CSW stored 2 Chan activity ID not stored 3 Not oper.
Store Character	STC	42	RX	R1, D2(X2,B2)	Opr 1 (24-31) replaces the character at opr 2's address.	Addr Protect	Unchanged
Store Characters Under Mask	STCM	BE	RS	R1, M3, D2(B2)	Bytes selected from opr 1 under control of mask are stored at opr 2.	Addr Opera Protect	Unchanged
Store Clock	STCK	B205	S	D1(B1)	Current val of TOD clock stored in 8 bytes at opr 1.	Addr Protect Opera	0 Clock in set state 1 Clk in not-set state 2 Clk in error 3 Clk not oper or in stopped state
Store Clock Comparator	STCKC	B207	S	D1(B1)	Curr contents of clock comparator stored at opr 1.	Addr Priv Specif Protect Opera	Unchanged

Figure 19. System/370 Instructions
(Part 25 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Store Control	STCTL	B6	RS	R1, R3, D2(B2)	Control regs from opr 1 to opr 3 stored at opr 2.	Priv Addr Specif Protect Opera	Unchanged
Store CPU Address	STAP	B212	S	D2(B2)	CPU address stored at halfword location designated by second-operand address.	Specif Opera Priv	Unchanged
Store CPU ID	STIDP	B202	S	D1(B1)	CPU info stored in 8 bytes at opr1.	Priv Addr Specif Protect Opera	Unchanged
Store CPU Timer	STPT	B209	S	D1(B1)	Curr contents of CPU timer stored in dblwd at opr 1.	Priv Addr Specif Protect Opera	Unchanged
Store Halfword	STH	40	RX	R1, D2(X2,B2)	Opr 1 (16 low-order bits) is stored at opr 2's location. (Hi-order bits, opr 1, ignored and unchanged)	Addr Specif Protect	Unchanged
Store (Long)	STD	60	RX	R1, D2(X2,B2)	FP opr 1 to opr 2's location.	Addr Protect Specif Opera	Unchanged
Store Multiple	STM	90	RS	R1, R2, D2(B2)	Opr 1 thru opr 3 are stored at opr 2's location in ascending order. Starting reg specified by opr 1, ending reg specified by opr 3. (Reg wrap-around possible)	Addr Specif Protect	Unchanged

Figure 19. System/370 Instructions (Part 26 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Store Prefix	STPX	B211	S	D2(B2)	Prefix register contents are stored at word location designated by second operand address.	Specif Opera Priv	Unchanged
Store (Short)	STE	70	RX	R1, D2(X2,B2)	FP opr 1 is stored at opr 2's location (Low-order half of FPR ignored and unchanged)	Opera Addr Specif Protect	Unchanged
Store Then AND System Mask	STNSM	AC	SI	D1(B1),I2	Bits 0-7 current PSW stored at opr 1, then these bits ANDed with opr 2 and replaced in current PSW.	Addr Priv Protect Opera	Unchanged
Store Then OR System Mask	STOSM	AD	SI	D1(B1), I2	Bits 0-7 of current PSW stored at opr 1, then these bits ORed with opr 2 and replaced in current PSW.	Addr Priv Protect Opera	Unchanged
Subtract	S	5B	RX	R1, D2(X2)	Subtract opr 2 from opr 1 and place the difference into opr 1.	Addr Fxpt Oflo Specif	0 Dif = 0 1 Dif < 0 2 Dif > 0 3 Overflow
Subtract	SR	1B	RR	R1, R2	Subtract opr 2 from opr 1; difference placed into opr 1.	Fxpt Oflo	0 Dif = 0 1 Dif < 0 2 Dif > 0 3 Overflow

Figure 19. System/370 Instructions (Part 27 of 33)

Figure 19. System/370 Instructions (Part 28 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Subtract Decimal	SP	FB	SS	D1(L1,B1), D2(L2,B2)	Subtract dec opr 2 from opr 1; difference stored into opr 1. (Right to left byte by byte) (Both opr's must be in packed format) (Fields can overlap if low-order bytes coincide)	Opera Addr Data Dec Oflo Protect	0 Dif = 0 1 Dif < 0 2 Dif > 0 3 Overflow
Subtract Halfword	SH	4B	RX	R1, D2(X2,B2)	Opr 2 halfword expanded to fullword and subtracted from opr 1; difference placed into opr 1.	Addr Fxpt Oflo Specif	0 Dif = 0 1 Dif < 0 2 Dif > 0 3 Overflow
Subtract Logical	SL	5F	RX	R1, D2(X2,B2)	Subtract opr 2 from opr 1; difference placed into opr 1.	Addr Specif	0 -- 1 Dif ≠ 0 No Carry 2 Dif = 0 Carry 3 Dif ≠ 0 Carry
Subtract Logical	SLR	1F	RR	R1, R2	Subtract opr 2 from opr 1; difference placed into opr 1.	None	0 -- 1 Dif ≠ 0 No Carry 2 Dif = 0 Carry 3 Dif ≠ 0 Carry
Subtract Normalized (Extended)	SXR	37	RR	R1, R2	FP subtract extended opr 2 from extended opr 1. (FPR pair) (FPR pair) Extended difference is put in opr 1 (FPR pair) (Sign of extended opr 2 is inverted before the addition) (Only FPR 0 and FPR 4 may be specified for either opr 1 or opr 2) (Continued)	Specif Exp Oflo Exp Uflo Signif	0 Fract = 0 1 Fract < 0 2 Fract > 0 3 --

Figure 19. System/370 Instructions (Part 29 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Subtract Normalized (Extended) (Cont'd)	SXR	37	RR	R1, R2	(High-order and low-order signs of a FPR pair are always the same in extended precision) (Low-order characteristic is made 14 < high-order characteristic except when the result would be > 0, then the low-order characteristic is made 128 > its correct value; sign of low-order characteristic remains the same as high-order characteristic)		
Subtract Normalized (Long)	SD	6B	RX	R1, D2(X2,B2)	FP Subtract opr 2 from opr 1 and the difference placed into opr 1. (The sign of opr 2 is inverted before the addition.)	Addr Specif Signif Exp Oflo Exp Uflo	Result 0 Fract = 0 1 Result < 0 2 Result > 0 3 Exp Oflo
Subtract Normalized (Long)	SDR	2B	RR	R1, R2	FP Subtract opr 2 from opr 1 (FPR) (FPR) (The sign of opr 2 is inverted before the addition.)	Specif Signif Exp Oflo Exp Uflo	Result 0 Fract = 0 1 Result < 0 2 Result > 0 3 Exp Oflo
Subtract Normalized (Short)	SE	7B	RX	R1, D2(X2,B2)	FP Subtract opr 2 from opr 1 (The sign of opr 2 is inverted before the addition.) (Low-order halves of FPR ignored and unchanged).	Addr Specif Signif Exp Oflo Exp Uflo	Result 0 Fract = 0 1 Result < 0 2 Result > 0 3 Exp Oflo

Figure 19. System/370 Instructions (Part 30 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Subtract Normalized (Short)	SER	3B	RR	R1, R2	Subtract opr 2 from opr 1 (The sign of opr 2 is inverted before the addition.) (Low-order halves of FPRs ignored and unchanged)	Specif Signif Exp Oflo Exp Uflo	Result 0 Fract = 0 1 Result < 0 2 Result > 0 3 Exp Oflo
Subtract Unnormalized (Long)	SW	6F	RX	R1, D2(X2,B2)	FP Subtract opr 2 from opr 1 (Sto) (FPR) (The sign of opr 2 is inverted before the addition.)	Addr Specif Signif Exp Oflo Opera	Result 0 Fract = 0 1 Result < 0 2 Result > 0 3 Exp Oflo
Subtract Unnormalized (Long)	SWR	2F	RR	R1, R2	FP Subtract opr 2 from opr 1 (FPR) (FPR) (The sign of opr 2 is inverted before the addition.)	Specif Signif Exp Oflo Opera	Result 0 Fract = 0 1 Result < 0 2 Result > 0 3 Exp Oflo
Subtract Unnormalized (Short)	SU	7F	RX	R1, D2(X2,B2)	FP Subtract opr 2 from opr 1 (Sto) (FPR) (Low-order half of FPR ignored and unchanged) (The sign of opr 2 is inverted before the addition.)	Addr Specif Signif Exp Oflo Opera	Result 0 Fract = 0 1 Result < 0 2 Result > 0 3 Exp Oflo
Subtract Unnormalized (Short)	SUR	3F	RR	R1, R2	FP Subtract opr 2 from opr 1 (FPR) (FPR) (Low-order halves of FPRs ignored and unchanged) (The sign of opr 2 is inverted before the addition.)	Specif Signif Exp Oflo Opera	Result 0 Fract = 0 1 Result < 0 2 Result > 0 3 Exp Oflo

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Supervisor Call	SVC	OA	RR	I	Immediate bits (8-15) placed in loc. 138 and PSW swap performed. (16-23) are made zero. (Old PSW at loc 32). (New PSW from loc 96).	None	Unchanged
Test and Set	TS	93	SI	D1(B1)	Hi-order bit of 1st byte of opr adr sets cond code. Entire byte then set to 1's	Addr Protect	0 Hi-order bit = 0 1 Hi-order bit = 1 2 -- 3 --
Test Channel	TCH	9F	S	D1(B1)	Opr 1 (16-23) identifies the tested channel. (Bits 24-31 are ignored.) (Instruction checks the channel's status and sets appropriate cond code.)	Priv	0 Chan Avl 1 Int Pending 2 Chan in Burst Mode 3 Chan not Operational
Test I/O	TIO	9D	S	D1(B1)	Opr 1 (16-31) identifies the tested channel, control unit, and I/O device. Used to clear a pending interrupt. (CSW stored at loc 64): Subchannel contains a pending interrupt. I/O device contains a pending interrupt. Control unit or I/O device is executing a previous operation or a pending channel-end/control unit-end for another I/O device. Channel or I/O device equipment error or device not ready.	Priv	0 Available 1 CSW Stored 2 Channel or Subchan Busy 3 Not Operational

Figure 19. System/370 Instructions (Part 31 of 33)

Figure 19. System/370 Instructions (Part 32 of 33)

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Test Under Mask	TM	91	SI	D1(B1), I2	Immediate bits (8-15) used as a mask to compare against opr 1. Mask bit 1: storage bit tested. Mask bit 0: storage bit ignored.	Addr	0 Selected bits all zero (mask is all zero) 1 Selected bits mixed 0's and 1's 3 Selected bits all 1's
Translate	TR	DC	SS	D1(L,B1), D2(B2)	Opr 1 (argument byte) added to the initial adr of opr 2 (24-31). This adr now is the loc of the function byte which replaces the original argument byte (left to right byte by byte) (All data is valid) (Oper is terminated when opr 1 field is exhausted)	Addr Protect	Unchanged
Translate and Test	TRT	DD	SS	D1(L,B1), D2(B2)	(Same as TR) When the function byte is a zero the next argument byte is translated. Both opr's remain unchanged. When the function byte is a non-zero the operation is completed. The generated argument adr is placed into GPR 1, 8-31. Bits 0-7 remain unchanged. The function byte is placed into GPR 2, 24-31. (Left to right byte by byte). Bits 0-23 remain unchanged. If opr 1 is exhausted before a non-zero cond, the opr is completed and GPRs 1 and 2 remain unchanged.	Addr	0 All function bytes 0 1 Non-0 function byte met 2 Last function byte non-0 3 Not used

Operation	Mnemonic	Op Code	Format	Operands	Description	Exceptions	Cond Code
Unpack	UNPK	F3	SS	D1(L1,B1), D2(L2,B2)	Change opr 2 from packed to zoned format and place into opr 1. (Right to left byte by byte) (No restrictions on overlapping fields) (Opr 2 may be extended with hi-order zeros.)	Addr Protect	Unchanged
Write Direct	WRD	84	SI	D1(B1), I2	The 1 byte from the instruction stream (8-15) is placed on the timing signal out, in a form of 8 timing pulses, along with a 9th pulse at the write-out line. The 8 bit lines at the direct-out lines are brought up by opr 1.	Priv Addr Opera	Unchanged
Zero and Add	ZAP	F8	SS	D1(L1,B1), D2(L2,B2)	Opr 1 cleared and opr 2 placed in opr 1 (Low-order opr's may coincide) (Opr 2 must be in packed format) (Opr 1 field must be large enough for all opr 2 significant digits) (Opr 2 extended with zeros to fill opr 1.)	Addr Data Dec Oflo Protect Opera	0 Result = 0 1 Result < 0 2 Result > 0 3 Overflow

Figure 19. System/370 Instructions (Part 33 of 33)

Figure 20. Assembler Instructions (Part 1 of 4)

Operation	Name Entry	Operand Entry	Is used to:
ACTR	A sequence symbol or blank	A SETA expression	Limit the number of AGO and AIF operations executed: prevent incessant looping.
AGO	A sequence symbol or blank	A sequence symbol	Unconditionally alter the sequence in which statements are processed.
AIF	A sequence symbol or blank	A logical expression enclosed in parentheses, immediately followed by a sequence symbol	Conditionally alter the sequence in which statements are processed.
ANOP	A sequence symbol or blank	Must not be present	Act as the target of AGO and AIF instructions.
CCW	Any symbol or blank	Four operands, separated by commas	Define and generate an 8-byte Channel Command word having doubleword alignment.
CNOP	Any symbol or blank	Two absolute expressions, separated by a comma	Align the location on a specified halfword boundary.
COM	Any symbol or blank	Must not be present	Reserve a common area of storage referred to by independent assemblies that are linked and loaded together or execution.
COPY	Must not be present	One ordinary symbol	Obtain and copy source code from a PDS member into the program being assembled.
CSECT	Any symbol or blank	Must not be present	Identify the beginning or continuation of a control section (see DSECT instruction).
CXD	Any symbol or blank	Must not be present	Allocate a fullword that will contain the sum of the lengths of all external dummy sections when the program is executed.
DC	Any symbol or blank	One or more operands, separated by commas	Define data constants in storage (see DS instruction).
DROP	A sequence symbol or blank	One to sixteen absolute expressions, separated by commas; or blank	Inform the assembler that specified registers are no longer to be used as base registers (see USING instruction).
DS	Any symbol or blank	One or more operands, separated by commas	Reserve areas of storage without assembling their contents (see DC instruction).
DSECT	Any symbol or blank	Must not be present	Initiate or continue a dummy section; describe an area of storage without reserving it (see CSECT instruction).
DXD	Any symbol	One or more operands, separated by commas	Identify and define an external dummy section.

Figure 20. Assembler Instructions (Part 2 of 4)

Operation	Name Entry	Operand Entry	Is used to:
EJECT	A sequence symbol or blank	Must not be present	Start a new page in the assembly listing; specify the sectioning of the assembly listing.[1]
END	A sequence symbol or blank	A relocatable expression or blank	Terminate the assembly of a source module.
ENTRY	A sequence symbol or blank	One or more relocatable symbols separated by commas	Identify symbols that are defined in the same source module, but are referred to in another source module.
EQU	An ordinary symbol or a variable symbol	One to three operands, separated by commas	Assign values to symbols.
EXTRN	A sequence symbol or blank	One or more relocatable symbols, separated by commas	Identify symbols that are referred to in the same source module but are defined in another source module (see WXTRN instruction).
GBLA GBLB GBLC	Must not be present	One or more variable symbols that are to be used as SET symbols, separated by commas.[2]	Define a global Arithmetic, Binary, or Character SET symbol.
ICTL	Must not be present	One to three decimal values, separated by commas	Alter the position of Begin, End, and Continuation columns in the source module.
ISEQ	Must not be present	Two decimal values, separated by commas	Sequence-check the source module statements.
LCIA LCLB LCLC	Must not be present	One or more variable symbols, that are to be used as SET symbols, separated by commas.[2]	Define a local Arithmetic, Binary, or Character SET symbol.
LTORG	Any symbol or blank	Not required	Position a literal pool at other than the end of the first control section; ensure addressability of literals in a large control section.
MACRO[3]	Must not be present	Not required	Indicate the beginning of a macro definition.
MEND[3]	A sequence symbol or blank	Not required	Indicate the end of a macro definition.
MEXIT[3]	A sequence symbol or blank	Not required	Indicate an exit from a macro definition.
MNOTE	A sequence symbol or blank	A severity code (optional), comma, characters enclosed in apostrophes	Display an error severity code, generate a message.

Figure 20. Assembler Instructions (Part 3 of 4)

Operation	Name Entry	Operand Entry	Is used to:
OPSYN	An ordinary symbol	A mnemonic operation, a macro or assembler operation, a machine instruction operation, or a blank	Define a symbol to represent an operation code, or delete its properties as an operation code.
ORG	Any symbol or blank	A relocatable expression	Change the location counter to redefine poritons of a control section, especially, constant tables.
POP	A sequence symbol or blank	One or more operands, separated by commas	Restore the PRINT or USING status saved by the most recent PUSH instruction.
PRINT	A sequence symbol or blank	One to three operands	Control the amount of detail printed in the assembly listing.
PUNCH	A sequence symbol or blank	One to eighty characters, enclosed in apostrophes	Punch one card with the data specified in the operand, substituting values for variable symbols (see REPRO instruction).
PUSH	A sequence symbol or blank	One or more operands, separated by a comma	Save the current PRINT or USING status (see POP instruction).
REPRO	A sequence symbol or blank	Not required	Punch one card with the characters specified in the statement that follows (see PUNCH instruction).
SETA SETB SETC	A SETA symbol A SETB symbol A SETC symbol	An arithmetic expression, a logical expression, or a character expression	Assign a value to an Arithmetic, Binary, or Character SET symbol.
SPACE	A sequence symbol or blank	A decimal self-defining term or blank	Insert blank lines into the source module assembly listing to separate sections of code.[1]
START	Any symbol or blank	A self-defining term or blank	Initialize the location counter for, and name the first control section of the module.
TITLE	A variable symbol, and/or character string, or sequence symbol, or blank	One to 100 characters, enclosed in apostrophes	Produce headings on the assembly listing pages, punch identifying characters into the object deck.[1]
USING	A sequence symbol or blank	An absolute or relocatable expression followed by 1 to 16 absolute expressions, separated by commas	Identify registers that may be used by the assembler as base registers (see DROP instruction).

Operation	Name Entry	Operand Entry	Is used to:
WXTRN	A sequence symbol or blank	One or more relocatable symbols, separated by commas	Identify symbols referred to in the same source module, but defined in another source module in the same load module (see EXTRN instruction).

[1] The statement itself does not appear in the assembly listing.
[2] SET symbols can be defined as subscripted SET symbols.
[3] Can be used only as part of a macro definition.

Figure 20. Assembler Instructions (Part 4 of 4)

Instruction	Name Entry	Operand Entry
Model Statements	An ordinary symbol, a variable symbol, a sequence symbol, a combination of variable symbols and other characters that is equivalent to a symbol, or blank	Any combination of characters (including variable symbols)
Prototype Statement[1]	A symbolic parameter or blank	Zero or more operands that are symbolic parameters, separated by commas
Macro-Instruction Statement[2]	An ordinary symbol, a variable symbol, a sequence symbol, a combination of variable symbols and other characters that is equivalent to a symbol,[2] or blank	Zero or more positional operands and/or zero or more keyword operands separated by commas[2]
Assembler Language Statement	An ordinary symbol, a variable symbol, a sequence symbol, a combination of variable symbols and other characters that is equivalent to a symbol, or blank	Any combination of characters (including variable symbols)

[1] Can only be used as part of a macro definition.

[2] Variable symbols appearing in a macro instruction are replaced by their values before the macro instruction is processed.

Figure 21. Assembler Statements

TYPE	IMPLICIT LENGTH (BYTES)	ALIGN-MENT	LENGTH MODI-FIER RANGE	SPECIFIED BY	NUMBER OF CON-STANTS PER OPERAND	RANGE FOR EX-PONENTS	RANGE FOR SCALE	TRUN-CATION/ PADDING SIDE
C	as needed	byte	.1 to 256 (1)	characters	one			right
X	as needed	byte	.1 to 256 (1)	hexadecimal digits	multi-ple			left
B	as needed	byte	.1 to 256	binary digits	multi-ple			left
F	4	word	.1 to 8	decimal digits	multi-ple	-85 to +75	-187 to +346	left (3)
H	2	half word	.1 to 8	decimal digits	multi-ple	-85 to +75	-187 +346	left (3)
E	4	word	.1 to 8	decimal digits	multi-ple	-85 to +75	0-14	right (3)
D	8	double word	.1 to 8	decimal digits	multi-ple	-85 to +75	0-14	right (3)
L	16	double word	.1 to 16	decimal digits	multi-ple	-85 to +75	0-28	right (3)
P	as needed	byte	.1 to 16	decimal digits	multi-ple			left
Z	as needed	byte	.1 to 16	decimal digits	multi-ple			left
A	4	word	.1 to 4 (2)	any expression	multi-ple			left
Q	4	word	1-4	symbol nam-ing a DXD or DSECT	multi-ple			left
V	4	word	3, 4	relocatable symbol	multi-ple			left
S	2	half word	2 only	one absolute or relocat-able ex-pression or two absolute expressions: exp (exp)	multi-ple			
Y	2	half word	.1 to 2 (2)	any expression	multi-ple			left

(1) In a DS assembler instruction C and X type constants can have length specification to 65535.

(2) Bit length specification permitted with absolute expressions only. Relocatable A-type constants, 3 or 4 bytes only; relocatable Y-type constants, 2 bytes only.

(3) Errors will be flagged if significant bits are truncated or if the value specified cannot be contained in the implicit length of the constant.

Figure 22. Assembler Constants

Expression	Arithmetic Expressions	Character Expressions	Logical Expressions
Can contain	• Self-defining terms • Length, scaling, integer, count, and number attributes • SETA and SETB symbols • SETC symbols whose values are a decimal self-defining term • &SYSPARM if its value is a decimal self-defining term • Symbolic parameters if the corresponding operand is a decimal self-defining term • &SYSLIST (n) if the corresponding operand is a decimal self-defining term • &SYSLIST (n, m) if the corresponding operand is a decimal self-defining term • &SYSNDX	• Any combination of characters enclosed in apostrophes • Any variable symbol enclosed in apostrophes • A concatenation of variable symbols and other characters enclosed in apostrophes • A type attribute reference	• A 0 or a 1 • SETB symbols • Arithmetic relations[1] • Character relations[2] • Arithmetic value
Operations are	+, - (unary and binary), *, and /; parentheses permitted	concatenation, with a period (.)	AND, OR, and NOT parentheses permitted
Range of values	-2^{31} to $+2^{31}-1$	0 through 255 characters	0 (false) or 1 (true)
May be used in	• SETA operands • Arithmetic relations[1] • Subscripted SET symbols • SYSLIST subscript (s) • Substring notation • Sublist notation	• SETC operands • Character relations[2]	• SETB operands • AIF operands

[1] **An arithmetic relation consists of two arithmetic expressions related by the operators GT, LT, EQ, NE, GE, or LE.**

[2] **A character relation consists of two character expressions related by the operators GT, LT, EQ, NE, GE, or LE. Type attribute notation and Substring notation may also be used in character relations. The maximum size of the character expressions that can be compared is 255 characters. If the two character expressions are of unequal size, the smaller one will always compare less than the larger one.**

Figure 23. Assembler Conditional Assembly Expressions

Attribute	Notation	Can be used with:	Can be used only if type attribute is:	Can be used in:
Type	T'	Ordinary Symbols defined in open code; symbolic parameters inside macro definitions; SET symbols, &SYSPARM, &SYSDATE, &SYSTIME, inside or outside macro definitions; &SYSLIST (m), &SYSLIST (m,n), &SYSECT, &SYSNDX inside macro definitions	(May always be used)	1. SETC operand fields 2. Character relations
Length	L'	Ordinary Symbols defined in open code; symbolic parameters inside macro definitions; &SYSLIST (m), and &SYSLIST (n,n) inside macro definitions	Any letter except M,N,O,T and U	Arithmetic expressions
Scaling	S'	Ordinary Symbols defined in open code; symbolic parameters inside macro definitions; &SYSLIST (m), and &SYSLIST (m,n) inside macro definitions	H,F,G,D,E,L,K,P, and Z	Arithmetic expressions
Integer	I'	Ordinary Symbols defined in open code; symbolic parameters inside macro definitions; &SYSLIST (m), and &SYSLIST (m,n) inside macro definitions	H,F,G,D,E,L,K,P, and Z	Arithmetic expressions
Count	K'	Symbolic parameters inside macro definitions; SET symbols; all system variable symbols	Any letter	Arithmetic expressions
Number	N'	Symbolic parameters, &SYSLIST (m), and &SYSLIST (m,n) inside macro definitions	Any letter	Arithmetic expressions

Figure 24. Assembler Attributes

Variable Symbol	Declared by:	Initialized, or set to:	Value changed by:	May be used in:
Symbolic[1] parameter	Prototype statement	Corresponding macro instruction operand	(Constant throughout definition)	• Arithmetic expressions if operand is decimal self-defining term • Character expressions
SETA	LCLA or GBLA instruction	0	SETA instruction	• Arithmetic expressions • Character expressions
SETB	LCLB or GBLB instruction	0	SETB instruction	• Arithmetic expressions • Character expressions • Logical expressions
SETC	LCLC or GBLC instruction	String of length 0 (null)	SETC instruction	• Arithmetic expressions if value is decimal self-defining term • Character expressions
&SYSNDX[1]	The assembler	Macro instruction index	(Constant throughout definition; unique for each macro instruction)	• Arithmetic expressions • Character expressions
&SYSECT[1]	The assembler	Control section in which macro instruction appears	(Constant throughout definition; set by CSECT, DSECT, START, and COM)	• Character expressions

Figure 25. Assembler Variable Symbols (Part 1 of 2)

Variable Symbol	Declared by:	Initialized, or set to:	Value changed by:	May be used in:
&SYSLIST[1]	The assembler	Not applicable	Not applicable	● N'&SYSLIST in arithmetic expressions
&SYSLIST (n)[1] &SYSLIST (n,M)[1]	The assembler	Corresponding macro instruction operand	(Constant throughout definition)	● Arithmetic expressions if operand is decimal self-defining term ● Character expressions
&SYSPARM	PARM field	User defined or null	Constant throughout assembly	● Arithmetic expression if value is decimal self-defining term ● Character expression
&SYSTIME	The assembler	System time	Constant throughout assembly	● Character expression
&SYSDATE	The assembler	System date	Constant throughout assembly	● Character expression

[1] Can be used only in macro definitions.

Figure 25. Assembler Variable Symbols (Part 2 of 2)

HEXADECIMAL AND DECIMAL CONVERSION

From hex: locate each hex digit in its corresponding columr position and note the decimal equivalents. Add these to obtain the decimal value.

From decimal: (1) locate the largest decimal value in the table that will fit into the decimal number to be converted, and (2) note its hex equivalent and hex column position. (3) Find the decimal remainder. Repeat the process on this and subsequent remainders.

HEXADECIMAL COLUMNS											
6		5		4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
0123		4567		0123		4567		0123		4567	
BYTE				BYTE				BYTE			

POWERS OF 2

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$
$2^{44} = 16^{11}$
$2^{48} = 16^{12}$
$2^{52} = 16^{13}$
$2^{56} = 16^{14}$
$2^{60} = 16^{15}$

POWERS OF 16 TABLE

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

Figure 26. Hexadecimal and Decimal Conversion

Hexadecimal Addition and Subtraction Table

Example: 6 + 2 = 8, 8 - 2 = 6, and 8 - 6 = 2

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Hexadecimal Multiplication Table

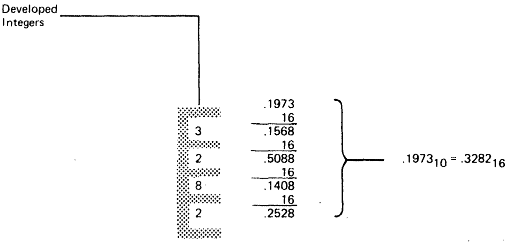
Example: 2 x 4 = 08, F x 2 = 1E

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
2	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	03	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	04	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	05	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	06	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	07	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	08	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	09	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

Figure 27. Hexadecimal Addition, Subtraction, and Multiplication Tables

Decimal to Hexadecimal Conversion: Locate the decimal fraction (.1973) in the table. If the exact figure is not shown, locate the next higher and lower fractions (.19726563, .19750977). The first digits of the hexadecimal fraction are at the top of the column (.32). To locate the third digit, determine by observation or subtraction the smaller difference between the known fraction and each of the found fractions. The smaller difference identifies the correct line (.008). The hexadecimal equivalent is .328.

If more places to the right of the decimal point are required in the hexadecimal fraction, multiply the decimal fraction by 16 and develop integers as successive terms of the hexadecimal fraction. Using the previous sample decimal fraction:



Hexadecimal to Decimal Conversion: Locate the first two digits (.1E) of the hexadecimal fraction (.1E9) in the horizontal row of column headings. Locate the third digit (.009) in the left most column of the table. Follow the .009 line horizontally to the right to the .1E column. The decimal equivalent is .11938477. The decimal fractions in the table were carried to eight places and rounded. If 2 places are required, or if the hexadecimal fraction exceeds the capacity of the table, express the hexadecimal fraction as powers of 16 (expansion). For example:

$$
\begin{aligned}
.1E94_{16} &= 1(16^{-1}) + 14(16^{-2}) + 9(16^{-3}) + 4(16^{-4}) \\
&= 1(.0625) + 14(.00390625) + 9(.000244140625) + 4(.0000152587890625) \\
&= .11944580078125000_{10}
\end{aligned}
$$

Negative Powers of 16 Table

n	16^n
0	1.0
−1	0.0625
−2	0.0039 0625
−3	0.0002 4414 0625
−4	1.5258 7890 6250 x 10^{-5}
−5	9.5367 4316 4062 x 10^{-7}
−6	5.9604 6447 7539 x 10^{-8}
−7	3.7252 9029 8461 x 10^{-9}
−8	2.3283 0643 6538 x 10^{-10}

Figure 28. Decimal to Hexadecimal Conversion Information (Part 1 of 5)

Trim Along This Line

IBM Virtual Machine Facility/370: Quick Guide for Users — Reference Summary

Order No. GX20-1926-3

Your suggestions help us produce better publications. We would appreciate any comments you have about the clarity, accuracy, and especially the usability of this reference summary.

FIRST CLASS
PERMIT NO. 172
BURLINGTON, MASS.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
VM/370 Publications
24 New England Executive Park
Burlington, Massachusetts 01803

	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09	.0A	.0B	.0C	.0D	.0E	.0F
.000	.00000000	.00390625	.00781250	.01171875	.01562500	.01953125	.02343750	.02734375	.03125000	.03515625	.03906250	.04296875	.04687500	.05078125	.05468750	.05859375
.001	.00024414	.00415039	.00805664	.01196289	.01586914	.01977539	.02368164	.02758789	.03149414	.03540039	.03930664	.04321289	.04711914	.05102539	.05493164	.05883789
.002	.00048828	.00439453	.00830078	.01220703	.01611328	.02001953	.02392578	.02783203	.03173828	.03564453	.03955078	.04345703	.04736328	.05126953	.05517578	.05908203
.003	.00073242	.00463867	.00854492	.01245117	.01635742	.02026367	.02416992	.02807617	.03198242	.03588867	.03979492	.04370117	.04760742	.05151367	.05541992	.05932617
.004	.00097656	.00488281	.00878906	.01269531	.01660156	.02050781	.02441406	.02832031	.03222656	.03613281	.04003906	.04394531	.04785156	.05175781	.05566406	.05957031
.005	.00122070	.00512695	.00903320	.01293945	.01684570	.02075195	.02465820	.02856445	.03247070	.03637695	.04028320	.04418945	.04809570	.05200195	.05590820	.05981445
.006	.00146484	.00537109	.00927734	.01318359	.01708984	.02099609	.02490234	.02880859	.03271484	.03662109	.04052734	.04443359	.04833984	.05224609	.05615234	.06005859
.007	.00170898	.00561523	.00952148	.01342773	.01733398	.02124023	.02514648	.02905273	.03295898	.03686523	.04077148	.04467773	.04858398	.05249023	.05639648	.06030273
.008	.00195313	.00585938	.00976563	.01367188	.01757813	.02148438	.02539063	.02929688	.03320313	.03710938	.04101563	.04492188	.04882813	.05273438	.05664063	.06054688
.009	.00219727	.00610352	.01000977	.01391602	.01782227	.02172852	.02563477	.02954102	.03344727	.03735352	.04125977	.04516602	.04907227	.05297852	.05688477	.06079102
.00A	.00244141	.00634766	.01025391	.01416016	.01806641	.02197266	.02587891	.02978516	.03369141	.03759766	.04150391	.04541016	.04931641	.05322266	.05712891	.06103516
.00B	.00268555	.00659180	.01049805	.01440430	.01831055	.02221680	.02612305	.03002930	.03393555	.03784180	.04174805	.04565430	.04956055	.05346680	.05737305	.06127930
.00C	.00292969	.00683594	.01074219	.01464844	.01855469	.02246094	.02636719	.03027344	.03417969	.03808594	.04199219	.04589844	.04980469	.05371094	.05761719	.06152344
.00D	.00317383	.00708008	.01098633	.01489258	.01879883	.02270508	.02661133	.03051758	.03442383	.03833008	.04223633	.04614258	.05004883	.05395508	.05786133	.06176758
.00E	.00341797	.00732422	.01123047	.01513672	.01904297	.02294922	.02685547	.03076172	.03466797	.03857422	.04248047	.04638672	.05029297	.05419922	.05810547	.06201172
.00F	.00366211	.00756836	.01147461	.01538086	.01928711	.02319336	.02709961	.03100586	.03491211	.03881836	.04272461	.04663086	.05053711	.05444336	.05834961	.06225586
	.10	.11	.12	.13	.14	.15	.16	.17	.18	.19	.1A	.1B	.1C	.1D	.1E	.1F
.000	.06250000	.06640625	.07031250	.07421875	.07812500	.08203125	.08593750	.08984375	.09375000	.09765625	.10156250	.10546875	.10937500	.11328125	.11718750	.12109375
.001	.06274414	.06665039	.07055664	.07446289	.07836914	.08227539	.08618164	.09008789	.09399414	.09790039	.10180664	.10571289	.10961914	.11352539	.11743164	.12133789
.002	.06298828	.06689453	.07080078	.07470703	.07861328	.08251953	.08642578	.09033203	.09423828	.09814453	.10205078	.10595703	.10986328	.11376953	.11767578	.12158203
.003	.06323242	.06713867	.07104492	.07495117	.07885742	.08276367	.08666992	.09057617	.09448242	.09838867	.10229492	.10620117	.11010742	.11401367	.11791992	.12182617
.004	.06347656	.06738281	.07128906	.07519531	.07910156	.08300781	.08691406	.09082031	.09472656	.09863281	.10253906	.10644531	.11035156	.11425781	.11816406	.12207031
.005	.06372070	.06762695	.07153320	.07543945	.07934570	.08325195	.08715820	.09106445	.09497070	.09887695	.10278320	.10668945	.11059570	.11450195	.11840820	.12231445
.006	.06396484	.06787109	.07177734	.07568359	.07958984	.08349609	.08740234	.09130859	.09521484	.09912109	.10302734	.10693359	.11083984	.11474609	.11865234	.12255859
.007	.06420898	.06811523	.07202148	.07592773	.07983398	.08374023	.08764648	.09155273	.09545898	.09936523	.10327148	.10717773	.11108398	.11499023	.11889648	.12280273
.008	.06445313	.06835938	.07226563	.07617188	.08007813	.08398438	.08789063	.09179688	.09570313	.09960938	.10351563	.10742188	.11132813	.11523438	.11914063	.12304688
.009	.06469727	.06860352	.07250977	.07641602	.08032227	.08422852	.08813477	.09204102	.09594727	.09985352	.10375977	.10766602	.11157227	.11547852	.11938477	.12329102
.00A	.06494141	.06884766	.07275391	.07666016	.08056641	.08447266	.08837891	.09228516	.09619141	.10009766	.10400391	.10791016	.11181641	.11572266	.11962891	.12353516
.00B	.06518555	.06909180	.07299805	.07690430	.08081055	.08471680	.08862305	.09252930	.09643555	.10034180	.10424805	.10815430	.11206055	.11596680	.11987305	.12377930
.00C	.06542969	.06933594	.07324219	.07714844	.08105469	.08496094	.08886719	.09277344	.09667969	.10058594	.10449219	.10839844	.11230469	.11621094	.12011719	.12402344
.00D	.06567383	.06958008	.07348633	.07739258	.08129883	.08520508	.08911133	.09301758	.09692383	.10083008	.10473633	.10864258	.11254883	.11645508	.12036133	.12426758
.00E	.06591797	.06982422	.07373047	.07763672	.08154297	.08544922	.08935547	.09326172	.09716797	.10107422	.10498047	.10888672	.11279297	.11669922	.12060547	.12451172
.00F	.06616211	.07006836	.07397461	.07788086	.08178711	.08569336	.08959961	.09350586	.09741211	.10131836	.10522461	.10913086	.11303711	.11694336	.12084961	.12475586
	.20	.21	.22	.23	.24	.25	.26	.27	.28	.29	.2A	.2B	.2C	.2D	.2E	.2F
.000	.12500000	.12890625	.13281250	.13671875	.14062500	.14453125	.14843750	.15234375	.15625000	.16015625	.16406250	.16796875	.17187500	.17578125	.17968750	.18359375
.001	.12524414	.12915039	.13305664	.13696289	.14086914	.14477539	.14868164	.15258789	.15649414	.16040039	.16430664	.16821289	.17211914	.17602539	.17993164	.18383789
.002	.12548828	.12939453	.13330078	.13720703	.14111328	.14501953	.14892578	.15283203	.15673828	.16064453	.16455078	.16845703	.17236328	.17626953	.18017578	.18408203
.003	.12573242	.12963867	.13354492	.13745117	.14135742	.14526367	.14916992	.15307617	.15698242	.16088867	.16479492	.16870117	.17260742	.17651367	.18041992	.18432617
.004	.12597656	.12988281	.13378906	.13769531	.14160156	.14550781	.14941406	.15332031	.15722656	.16113281	.16503906	.16894531	.17285156	.17675781	.18066406	.18457031
.005	.12622070	.13012695	.13403320	.13793945	.14184570	.14575195	.14965820	.15356445	.15747070	.16137695	.16528320	.16918945	.17309570	.17700195	.18090820	.18481445
.006	.12646484	.13037109	.13427734	.13818359	.14208984	.14599609	.14990234	.15380859	.15771484	.16162109	.16552734	.16943359	.17333984	.17724609	.18115234	.18505859
.007	.12670898	.13061523	.13452148	.13842773	.14233398	.14624023	.15014648	.15405273	.15795898	.16186523	.16577148	.16967773	.17358398	.17749023	.18139648	.18530273
.008	.12695313	.13085938	.13476563	.13867188	.14257813	.14648438	.15039063	.15429688	.15820313	.16210938	.16601563	.16992188	.17382813	.17773438	.18164063	.18554688
.009	.12719727	.13110352	.13500977	.13891602	.14282227	.14672852	.15063477	.15454102	.15844727	.16236352	.16625977	.17016602	.17407227	.17797852	.18188477	.18579102
.00A	.12744141	.13134766	.13525391	.13916016	.14306641	.14697266	.15087891	.15478516	.15869141	.16259766	.16650391	.17041016	.17431641	.17822266	.18212891	.18603516
.00B	.12768555	.13159180	.13549805	.13940430	.14331055	.14721680	.15112305	.15502930	.15893555	.16284180	.16674805	.17065430	.17456055	.17846680	.18237305	.18627930
.00C	.12792969	.13183594	.13574219	.13964844	.14355469	.14746094	.15136719	.15527344	.15917969	.16308594	.16699219	.17089844	.17480469	.17871094	.18261719	.18652344
.00D	.12817383	.13208008	.13598633	.13989258	.14379883	.14770508	.15161133	.15551758	.15942383	.16333008	.16723633	.17114258	.17504883	.17895508	.18286133	.18676758
.00E	.12841797	.13232422	.13623047	.14013672	.14404297	.14794922	.15185547	.15576172	.15966797	.16357422	.16748047	.17138672	.17529297	.17919922	.18310547	.18701172
.00F	.12866211	.13256836	.13647461	.14038086	.14428711	.14819336	.15209961	.15600586	.15991211	.16381836	.16772461	.17163086	.17553711	.17944336	.18334961	.18725586
	.30	.31	.32	.33	.34	.35	.36	.37	.38	.39	.3A	.3B	.3C	.3D	.3E	.3F
.000	.18750000	.19140625	.19531250	.19921875	.20312500	.20703125	.21093750	.21484375	.21875000	.22265625	.22656250	.23046875	.23437500	.23828125	.24218750	.24609375
.001	.18774414	.19165039	.19555664	.19946289	.20336914	.20727539	.21118164	.21508789	.21899414	.22290039	.22680664	.23071289	.23461914	.23852539	.24243164	.24633789
.002	.18798828	.19189453	.19580078	.19970703	.20361328	.20751953	.21142578	.21533203	.21923828	.22314453	.22705078	23095703	.23486328	.23876953	.24267578	.24658203
.003	.18823242	.19213867	.19604492	.19995117	.20385742	.20776367	.21166992	.21557617	.21948242	.22338867	.22729492	.23120117	.23510742	.23901367	.24291992	.24682617
.004	.18847656	.19238281	.19628906	.20019531	.20410156	.20800781	.21191406	.21582031	.21972656	.22363281	.22753906	.23144531	.23535156	.23925781	.24316406	.24707031
.005	.18872070	.19262695	.19653320	.20043945	.20434570	.20825195	.21215820	.21606445	.21997070	.22387695	.22778320	.23168945	.23559570	.23950195	.24340820	.24731445
.006	.18896484	.19287109	.19677734	.20068359	.20458984	.20849609	.21240234	.21630859	.22021484	.22412109	.22802734	.23193359	.23583984	.23974609	.24365234	.24755859
.007	.18920898	.19311523	.19702148	.20092773	.20483398	.20874023	.21264648	.21655273	.22045898	.22436523	.22827148	.23217773	.23608398	.23999023	.24389648	.24780273
.008	.18945313	.19335938	.19726563	.20117188	.20507813	.20898438	.21289063	.21679688	.22070313	.22460938	.22851563	.23242188	.23632813	.24023438	.24414063	.24804688
.009	.18969727	.19360352	.19750971	.20141602	.20532227	.20922852	.21313477	.21704102	.22094727	.22485352	.22875977	.23266602	.23657227	.24047852	.24438477	.24829102
.00A	.18994141	.19384766	.19775391	.20166016	.20556641	.20947266	.21337891	.21728516	.22119141	.22509766	.22900391	.23291016	.23681641	.24072266	.24462891	.24853516
.00B	.19018555	.19409180	.19799805	.20190430	.20581055	.20971680	.21362305	.21752930	.22143555	.22534180	.22924805	.23315430	.23706055	.24096680	.24487305	.24877930
.00C	.19042969	.19433594	.19824219	.20214844	.20605469	.20996094	.21386719	.21777344	.22167969	.22558594	.22949219	.23339844	.23730469	.24121094	.24511719	.24902344
.00D	.19067383	.19458008	.19848633	.20239258	.20629883	.21020508	.21411133	.21801758	.22192383	.22583008	.22973633	.23364258	.23754883	.24145508	.24536133	.24926758
.00E	.19091797	.19482422	.19873047	.20263672	.20654297	.21044922	.21435547	.21826172	.22216797	.22607422	.22998047	.23388672	.23779297	.24169922	.24560547	24951172
.00F	.19116211	.19506836	.19897461	.20288086	.20678711	.21069336	.21459961	.21850586	.22241211	.22631836	.23022461	.23413086	.23803711	.24194336	.24584961	.24975586

Figure 28. Decimal to Hexadecimal Conversion Information (Part 2 of 5)

	.40	.41	.42	.43	.44	.45	.46	.47	.48	.49	.4A	.4B	.4C	.4D	.4E	4F
.000	.25000000	.25390625	.25781250	.26171875	.26562500	.26953125	.27343750	.27734375	.28125000	.28515625	.28906250	.29296875	.29687500	.30078125	.30468750	.30859375
.001	.25024414	.25415039	.25805664	.26196289	.26586914	.26977539	.27368164	.27758789	.28149414	.28540039	.28930664	.29321289	.29711914	.30102529	.30493164	.30883789
.002	.25048828	.25439453	.25830078	.26220703	.26611328	.27001953	.27392578	.27783203	.28173828	.28564453	.28955078	.29345703	.29736328	.30126953	.30517578	.30908203
.003	.25073242	.25463867	.25854492	.26245117	.26635742	.27026367	.27416992	.27807617	28198242	.28588867	.28979492	.29370117	.29760742	.30151367	.30541992	.30932617
.004	.25097656	.25488281	.25878906	.26269531	.26660156	.27050781	.27441406	.27832031	.28222656	.28613281	.29003906	.29394531	.29785156	.30175781	.30566406	.30957031
.005	.25122070	.25512695	.25903320	.26293945	.26684570	.27075195	.27465820	.27856445	.28247070	.28637695	.29028320	.29418945	.29809570	.30200195	.30590820	.30981445
.006	.25146484	.25537109	.25927734	.26318359	.26708984	.27099609	.27490234	.27880859	.28271484	.28662109	.29052734	.29443359	.29833984	.30224609	.30615234	.31005859
.007	.25170898	.25561523	.25952148	.26342773	.26733398	.27124023	.27514648	.27905273	.28295898	.28686523	.29077148	.29467773	.29858398	.30249023	.30639648	.31030273
.008	.25195313	.25585938	.25976563	.26367188	.26757813	.27148438	.27539063	.27929688	.28320313	.28710938	.29101563	.29492188	.29882813	.30273438	.30664063	.31054688
.009	.25219727	.25610352	.26000977	.26391602	.26782227	.27172852	.27563477	.27954102	.28344727	.28735352	.29125977	.29516602	.29907227	.30297852	.30688477	.31079102
.00A	.25244141	.25634766	.26025391	.26416016	.26806641	.27197266	.27587891	.27978516	.28369141	.28759766	.29150390	.29541016	.29931641	.30322266	.30712891	.31103516
.00B	.25268555	.25659180	.26049805	.26440430	.26831055	.27221680	.27612305	.28002930	.28393555	.28784180	.29174805	.29565430	.29956055	.30346680	.30737305	.31127930
.00C	.25292969	.25683594	.26074219	.26464844	.26855469	.27246094	.27636719	.28027344	.28417969	.28808594	.29199219	.29589844	.29980469	.30371094	.30761719	.31152344
.00D	.25317383	.25708008	.26098633	.26489258	.26879883	.27270508	.27661133	.28051758	.28442383	.28833008	.29223633	.29614258	.30004883	.30395508	.30786133	.31176758
.00E	.25341797	.25732422	.26123047	.26513672	.26904297	.27294922	.27685547	.28076172	.28466797	.28857422	.29248047	.29638672	.30029297	.30419922	.30810547	.31201172
.00F	.25366211	.25756836	.26147461	.26538086	.26928711	.27319336	.27709961	.28100586	.28491211	.28881836	.29272461	.29663086	.30053711	.30444336	.30834961	.31225586
	.50	.51	.52	.53	.54	.55	.56	.57	.58	.59	.5A	.5B	.5C	.5D	.5E	.5F
.000	.31250000	.31640625	.32031250	.32421875	.32812500	.33203125	.33593750	.33984375	.34375000	.34765625	.35156250	.35546875	.35937500	.36328125	.36718750	.37109375
.001	.31274414	.31665039	.32055664	.32446289	.32836914	.33227539	.33618164	.34008789	.34399414	.34790039	.35180664	.35571289	.35961914	.36352539	.36743164	.37133789
.002	.31298828	.31689453	.32080078	.32470703	.32861328	.33251953	.33642578	.34033203	.34423828	.34814453	.35205078	.35595703	.35986328	.36376953	.36767578	.37158203
.003	.31323242	.31713867	.32104492	.32495117	.32885742	.33276367	.33666992	.34057617	.34448242	.34838867	.35229492	.35620117	.36010742	.36401367	.36791992	.37182617
.004	.31347656	.31738281	.32128906	.32519531	.32910156	.33300781	.33691406	.34082031	.34472656	.34863281	.35253906	.35644531	.36035156	.36425781	.36816406	.37207031
.005	.31372070	.31762695	.32153320	.32543945	.32934570	.33325195	.33715820	.34106445	.34497070	.34887695	.35278320	.35668945	.36059570	.36450195	.36840820	.37231445
.006	.31396484	.31787109	.32177734	.32568359	.32958984	.33349609	.33740234	.34130859	.34521484	.34912109	.35302734	.35693359	.36083984	.36474609	.36865234	.37255859
.007	.31420898	.31811523	.32202148	.32592773	.32983398	.33374023	.33764648	.34155273	.34545898	.34936523	.35327148	.35717773	.36108398	.36499023	.36889648	.37280273
.008	.31445313	31835938	.32226563	.32617188	.33007813	.33398438	.33789063	.34179688	.34570313	.34960938	.35351563	.35742188	.36132813	.36523438	.36914063	.37304688
.009	.31469727	.31860352	.32250977	.32641602	'33032227	.33422852	.33813477	.34204102	.34594727	.34985352	.35375977	.35766602	.36157227	.36547852	.36938477	.37329102
.00A	.31494141	.31884766	.32275391	.32666016	.33056641	.33447266	.33837891	.34228516	.34619141	.35009766	.35400391	.35791016	.36181641	.36572266	.36962891	.37353516
.00B	.31518555	.31909180	.32299805	.32690430	.33081055	.33471680	.33862305	.34252930	.34643555	.35034180	.35424805	.35815430	.36206055	.36596680	.36987305	.37377930
.00C	.31542969	.31933594	.32324219	.32714844	.33105469	.33496094	.33886719	.34277344	.34667969	.35058594	.35449219	.35839844	.36230469	.36621094	.37011719	.37402344
.00D	.31567383	.31958008	.32348633	.32739258	.33129883	.33520508	.33911133	.34301758	.34692383	.35083008	.35473633	.35864258	.36254883	.36645508	.37036133	.37426758
.00E	.31591797	.31982422	.32373047	.32763672	.33154297	.33544922	.33935547	.34326172	.34716797	.35107422	.35498047	.35888672	.36279297	.36669922	.37060547	.37451172
.00F	.31616211	.32006836	.32397461	.32788086	.33178711	.33569336	.33959961	.34350586	.34741211	.35131836	.35522461	.35913086	.36303711	.36694336	.37084961	.37475586
	.60	.61	.62	.63	.64	.65	.66	.67	.68	.69	.6A	.6B	.6C	.6D	.6E	.6F
.000	.37500000	.37890625	.38281250	.38671875	.39062500	.39453125	.39843750	.40234375	.40625000	.41015625	.41406250	.41796875	.42187500	.42578125	.42968750	.43359375
.001	.37524414	.37915039	.38305664	.38696289	.39086914	.39477539	.39868164	.40258789	.40649414	.41040039	.41430664	.41821289	.42211914	.42602539	.42993164	.43383789
.002	.37548828	.37939453	.38330078	.38720703	.39111328	.39501953	.39892578	.40283203	.40673828	.41064453	.41455078	.41845703	.42236328	.42626953	.43017578	.43408203
.003	.37573242	.37963867	.38354492	.38745117	.39135742	.39526367	.39916992	.40307617	.40698242	.41088867	.41479492	.41870117	.42260742	.42651367	.43041992	.43432617
.004	.37597656	.37988281	.38378906	.38769531	.39160156	.39550781	.39941406	.40332031	.40722656	.41113281	.41503906	.41894531	.42285156	.42675781	.43066406	.43457031
.005	.37622070	.38012695	.38403320	.38793945	.39184570	.39575195	.39965820	.40356445	.40747070	.41137695	.41528320	.41918945	.42309570	.42700195	.43090820	.43481445
.006	.37646484	.38037109	.38427734	.38818359	.39208984	.39599609	.39990234	.40380859	.40771484	.41162109	.41552734	.41943359	.42333984	.42724609	.43115234	.43505859
.007	.37670898	.38061523	.38452148	.38842773	.39233398	.39624023	.40014648	.40405273	.40795898	.41186523	.41577148	.41967773	.42358398	.42749023	.43139648	.43530273
.008	.37695313	.38085938	.38476563	.38867188	.39257813	.39648438	.40039063	.40429688	.40820313	.41210938	.41601563	.41992188	.42382813	.42773438	.43164063	.42554688
.009	.37719727	.38110352	.38500977	.38891602	.39282227	.39672852	.40063477	.40454102	.40844727	.41235352	.41625977	.42016602	.42407227	.42797852	.43188477	.43579102
.00A	.37744141	.38134766	.38525391	.38916016	.39306641	.39697266	.40087891	.40478516	.40869141	.41259766	.41650391	.42041016	.42431641	.42822266	.43212891	.43603516
.00B	.37768555	.38159180	.38549805	.38940430	.39331055	.39721680	.40112305	.40502930	.40893555	.41284180	.41674805	.42065430	.42456055	.42846680	.43237305	.43627930
.00C	.37792969	.38183594	.38574219	.38964844	.39355469	.39746094	.40136719	.40527344	.40917969	.41308594	.41699219	.42089844	.42480469	.42871094	.43261719	.43652344
.00D	.37817383	.38208008	.38598633	.38989258	.39379883	.39770508	.40161133	.40551758	.40942383	.41333008	.41723633	.42114258	.42504883	.42895508	.43286133	.43676758
.00E	.37841797	.38232422	.38623047	.39013672	.39404297	.39794922	.40185547	.40576172	.40966797	.41357422	.41748047	.42138672	.42529297	.42919922	.43310547	.43701172
.00F	.37866211	.38256836	.38647461	.39038086	.39428711	.39819336	.40209961	.40600586	.40991211	.41381836	.41772461	.42163086	.42553711	.42944336	.43334961	.43725586
	.70	.71	.72	.73	.74	.75	.76	.77	.78	.79	.7A	.7B	.7C	.7D	.7E	.7F
.000	.43750000	.44140625	.44531250	.44921875	.45312500	.45703125	.46093750	.46484375	.46875000	.47265625	.47656250	.48046575	.48437500	.48828125	.49218750	.49609375
.001	.43774414	.44165039	.44555664	.44946289	.45336914	.45727539	.46118164	.46508789	.46899414	.47290039	.47680664	.48071289	.48461914	.48852539	.49243164	.49633789
.002	.43798828	.44189453	.44580078	.44970703	.45361328	.45751953	.46142578	.46533203	.46923828	.47314453	.47705078	.48095703	.48486328	.48876953	.49267578	.49658203
.003	.43823242	.44213867	.44604492	.44995117	.45385742	.45776367	.46166992	.46557617	.46948242	.47338867	.47729492	.48120117	.48510742	.48901367	.49291992	.49682617
.004	.43847656	.44238281	.44628906	.45019531	.45410156	.45800781	.46191406	.46582031	.46972656	.47363281	.47753906	.48144531	.48535156	.48925781	.49316406	.49707031
.005	.43872070	.44262695	.44653320	.45043945	.45434570	.45825195	.46215820	.46606445	.46997070	.47387695	.47778320	.48168945	.48559570	.48950195	.49340820	.49731445
.006	.43896484	.44287109	.44677734	.45068359	.45458984	.45849609	.46240234	.46630859	.47021484	.47412109	.47802734	.48193359	.48583984	.48974609	.49365234	.49755859
.007	.43920898	.44311523	.44702148	.45092773	.45483398	.45874023	.46264648	.46655273	.47045898	.47436523	.47827148	.48217775	.48608398	.48999023	.49389648	.49780273
.008	.43945313	.44335938	.44726563	.45117188.	.45507813	.45898438	.46289063	.46679688	.47070313	.47460938	.47851563	.48242188	.48632813	.49023438	.49414063	.49804688
.009	.43969727	.44360352	.44750977	.45141602	.45532227	.45922852	.46313477	.46704102	.47094727	.47485352	.47875977	.48266602	.48657227	.49047852	.49438477	.49829102
.00A	.43994141	.44384766	.44775391	.45166016	.45556641	.45947266	.46337891	.46728516	.47119141	.47509766	.47900391	.48291016	.48681641	.49072266	.49462891	.49853516
.00B	.44018555	.44409180	.44799805	.45190430	.45581055	.45971680	.46362305	.46752930	.47143555	.47534180	.47924805	.48315430	.48706055	.49096680	.49487305	.49877930
.00C	.44042969	.44433594	.44824219	.45214844	.45605469	.45996094	.46386719	.46777344	.47167969	.47558594	.47949219	.48339844	.48730469	.49121094	.49511719	.49902344
.00D	.44067383	.44458008	.44848633	.45239258	.45629883	.46020508	.46411133	.46801758	.47192383	.47583008	.47973633	.48364258	.48754883	.49145508	.49536133	.49926758
.00E	.44091797	.44482422	.44873047	.45263672	.45654297	.46044922	.46435547	.46826172	.47216797	.47607422	.47998047	.48388672	.48779297	.49169922	.49560547	.49951172
.00F	.44116211	.44506836	.44897461	.45288086	.45678711	.46069336	.46459961	.46850586	.47241211	.47631836	.48022461	.48413086	.48803711	.49194336	.49584961	.49975586

Figure 28. Decimal to Hexadecimal Conversion Information (Part 3 of 5)

	.80	.81	.82	.83	.84	.85	.86	.87	.88	.89	.8A	.8B	.8C	.8D	.8E	.8F
.000	.50000000	.50390625	.50781250	.51171875	.51562500	.51953125	.52343750	.52734375	.53125000	.53515625	.53906250	.54296875	.54687500	.55078125	.55468750	.55859375
.001	.50024414	.50415039	.50805664	.51196289	.51586914	.51977539	.52368164	.52758789	.53149414	.53540039	.53930664	.54321289	.54711914	.55102539	.55493164	.55883789
.002	.50048828	.50439453	.50830078	.51220703	.51611328	.52001953	.52392578	.52783203	.53173828	.53564453	.53955078	.54345703	.54736328	.55126953	.55517578	.55908203
.003	.50073242	.50463867	.50854492	.51245117	.51635742	.52026367	.52416992	.52807617	.53198242	.53588867	.53979492	.54370117	.54760742	.55151367	.55541992	.55932617
.004	.50097656	.50488281	.50878906	.51269531	.51660156	.52050781	.52441406	.52832031	.53222656	.53613281	.54003906	.54394531	.54785156	.55175781	.55566406	.55957031
.005	.50122070	.50512695	.50903320	.51293945	.51684570	.52075195	.52465820	.52856445	.53247070	.53637695	.54028320	.54418945	.54809570	.55200195	.55590820	.55981445
.006	.50146484	.50537109	.50927734	.51318359	.51708984	.52099609	.52490234	.52880859	.53271484	.53662109	.54052734	.54443359	.54833984	.55224609	.55615234	.56005859
.007	.50170898	.50561523	.50952148	.51342773	.51733398	.52124023	.52514648	.52905273	.53295898	.53686523	.54077148	.54467773	.54858398	.55249023	.55639648	.56030273
.008	.50195313	.50585938	.50976563	.51367188	.51757813	.52148438	.52539063	.52929688	.53320313	.53710938	.54101563	.54492188	.54882813	.55273438	.55664063	.56054688
.009	.50219727	.50610352	.51000977	.51391602	.51782227	.52172852	.52563477	.52954102	.53344727	.53735352	.54125977	.54516602	.54907227	.55297852	.55688477	.56079102
.00A	.50244141	.50634766	.51025391	.51416016	.51806641	.52197266	.52587891	.52978516	.53369141	.53759766	.54150391	.54541016	.54931641	.55322266	.55712891	.56103516
.00B	.50268555	.50659180	.51049805	.51440430	.51831055	.52221680	.52612305	.53002930	.53393555	.53784180	.54174805	.54565430	.54956055	.55346680	.55737305	.56127930
.00C	.50292969	.50683594	.51074219	.51464844	.51855469	.52246094	.52636719	.53027344	.53417969	.53808594	.54199219	.54589844	.54980469	.55371094	.55761719	.56152344
.00D	.50317383	.50708008	.51098633	.51489258	.51879883	.52270508	.52661133	.53051758	.53442383	.53833008	.54223633	.54614258	.55004883	.55395508	.55786133	.56176758
.00E	.50341797	.50732422	.51123047	.51513672	.51904297	.52294922	.52685547	.53076172	.53466797	.53857422	.54248047	.54638672	.55029297	.55419922	.55810547	.56201172
.00F	.50366211	.50756836	.51147461	.51538086	.51928711	.52319336	.52709961	.53100586	.53491211	.53881836	.54272461	.54663086	.55053711	.55444336	.55834961	.56225586

	.90	.91	.92	.93	.94	.95	.96	.97	.98	.99	.9A	.9B	.9C	.9D	.9E	.9F
.000	.56250000	.56640625	.57031250	.57421875	.57812500	.58203125	.58593750	.58984375	.59375000	.59765625	.60156250	.60546875	.60937500	.61328125	.61718750	.62109375
.001	.56274414	.56665039	.57055664	.57446289	.57836914	.58227539	.58618164	.59008789	.59399414	.59790039	.60180664	.60571289	.60961914	.61352539	.61743164	.62133789
.002	.56298828	.56689453	.57080078	.57470703	.57861328	.58251953	.58642578	.59033203	.59423828	.59814453	.60205078	.60595703	.60986328	.61376953	.61767578	.62158203
.003	.56323242	.56713867	.57104492	.57495117	.57885742	.58276367	.58666992	.59057617	.59448242	.59838867	.60229492	.60620117	.61010742	.61401367	.61791992	.62182617
.004	.56347656	.56738281	.57128906	.57519531	.57910156	.58300781	.58691406	.59082031	.59472656	.59863281	.60253906	.60644531	.61035156	.61425781	.61816406	.62207031
.005	.56372070	.56762695	.57153320	.57543945	.57934570	.58325195	.58715820	.59106445	.59497070	.59887695	.60278320	.60668945	.61059570	.61450195	.61840820	.62231445
.006	.56396484	.56787109	.57177734	.57568359	.57958984	.58349609	.58740234	.59130859	.59521484	.59912109	.60302734	.60693359	.61083984	.61474609	.61865234	.62255859
.007	.56420898	.56811523	.57202148	.57592773	.57983398	.58374023	.58764648	.59155273	.59545898	.59936523	.60327148	.60717773	.61108398	.61499023	.61889648	.62280273
.008	.56445313	.56835938	.57226563	.57617188	.58007813	.58398438	.58789063	.59179688	.59570313	.59960938	.60351563	.60742188	.61132813	.61523438	.61914063	.62304688
.009	.56469727	.56860352	.57250977	.57641602	.58032227	.58422852	.58813477	.59204102	.59594727	.59985352	.60375977	.60766602	.61157227	.61547852	.61938477	.62329102
.00A	.56494141	.56884766	.57275391	.57666016	.58056641	.58447266	.58837891	.59228516	.59619141	.60009766	.60400391	.60791016	.61181641	.61572266	.61962891	.62343516
.00B	.56518555	.56909180	.57299805	.57690430	.58081055	.58471680	.58862305	.59252930	.59643555	.60034180	.60424805	.60815430	.61206055	.61596680	.61987305	.62377930
.00C	.56542969	.56933594	.57324219	.57714844	.58105469	.58496094	.58886719	.59277344	.59667969	.60058594	.60449219	.60839844	.61230469	.61621094	.62011719	.62402344
.00D	.56567383	.56958008	.57348633	.57739258	.58129883	.58520508	.58911133	.59301758	.59692383	.60083008	.60473633	.60864258	.61254883	.61645508	.62036133	.62426758
.00E	.56591797	.56982422	.57373047	.57763672	.58154297	.58544922	.58935547	.59326172	.59716797	.60107422	.60498047	.60888672	.61279297	.61669922	.62060547	.62451172
.00F	.56616211	.57006836	.57397461	.57788086	.58178711	.58569336	.58959961	.59350586	.59741211	.60131836	.60522461	.60913086	.61303711	.61694336	.62084961	.62475586

	.A0	.A1	.A2	.A3	.A4	.A5	.A6	.A7	.A8	.A9	.AA	.AB	.AC	.AD	.AE	.AF
.000	.62500000	.62890625	.63281250	.63671875	.64062500	.64453125	.64843750	.65234375	.65625000	.66015625	.66406250	.66796875	.67187500	.67578125	.67968750	.68359375
.001	.62524414	.62915039	.63305664	.63696289	.64086914	.64477539	.64868164	.65258789	.65649414	.66040039	.66430664	.66821289	.67211914	.67602539	.67993164	.68383789
.002	.62548828	.62939453	.63330078	.63720703	.64111328	.64501953	.64892578	.65283203	.65673828	.66064453	.66455078	.66845708	.67236328	.67626953	.68017578	.68408203
.003	.62573242	.62963867	.63354492	.63745117	.64135742	.64526367	.64916992	.65307617	.65698242	.66088867	.66479492	.66870117	.67260742	.67651367	.68041992	.68432617
.004	.62597656	.62988281	.63378906	.63769531	.64160156	.64550781	.64941406	.65332031	.65722656	.66113281	.66503906	.66894531	.67285156	.67675781	.68066406	.68457031
.005	.62622070	.63012695	.63403320	.63793945	.64184570	.64575195	.64965820	.65356445	.65747070	.66137695	.66528320	.66918945	.67309570	.67700195	.68090820	.68481445
.006	.62646484	.63037109	.63427734	.63818359	.64208984	.64599609	.64990234	.65000859	.65771484	.66162109	.66552734	.66943359	.67333984	.67724609	.68115234	.68505859
.007	.62670898	.63061523	.63452148	.63842773	.64233398	.64624023	.65014648	.65405273	.65795898	.66186523	.66577148	.66967773	.67358398	.67749023	.68139648	.68530273
.008	.62695313	.63085938	.63476563	.63867188	.64257818	.64648438	.65039063	.65429688	.65820313	.66210938	.66601563	.66992188	.67382813	.67773438	.68164063	.68554688
.009	.62719727	.63110352	.63500977	.63891602	.64282227	.64672852	.65063477	.65454102	.65844727	.66235352	.66625977	.67016602	.67407227	.67797852	.68188477	.68579102
.00A	.62744141	.63134766	.63525391	.63916016	.64306641	.64697266	.65087891	.65478516	.65869141	.66259766	.66650391	.67041016	.67431641	.67822266	.68212891	.68603516
.00B	.62768555	.63159180	.63549805	.63940430	.64331055	.64721680	.65112305	.65502930	.65893555	.66284180	.66674805	.67065430	.67456055	.67846680	.68237305	.68627930
.00C	.62792969	.63183594	.63574219	.63964844	.64355469	.64746094	.65136719	.65527344	.65917969	.66308594	.66699219	.67089844	.67480469	.67871094	.68261719	.68652344
.00D	.62817383	.63208008	.63598633	.63989258	.64379883	.64770508	.65161133	.65551758	.65942383	.66333008	.66723633	.67114258	.67504883	.67895508	.68286133	.68676758
.00E	.62841797	.63232422	.63623047	.64013672	.64404297	.64794922	.65185547	.65576172	.65966797	.66357422	.66748047	.67138672	.67529297	.67919922	.68310547	.68701172
.00F	.62866211	.63256836	.63647461	.64038086	.64428711	.64819336	.65209961	.65600586	.65991211	.66381836	.66772461	.67163086	.67553711	.67944336	.68334961	.68725586

	.B0	.B1	.B2	.B3	.B4	.B5	.B6	.B7	.B8	.B9	.BA	.BB	.BC	.BD	.BE	.BF
.000	.68750000	.69140625	.69531250	.69921875	.70312500	.70703125	.71093750	.71484375	.71875000	.72265625	.72656250	.73046875	.73437500	.73828125	.74218750	.74609375
.001	.68774414	.69165039	.69555664	.69946289	.70336914	.70727539	.71118164	.71508789	.71899414	.72290039	.72680664	.73071289	.73461914	.73852539	.74243164	.74633789
.002	.68798828	.69189453	.69580078	.69970703	.70361328	.70751953	.71142578	.71533203	.71923828	.72314453	.72705078	.73095703	.73486328	.73876953	.74267578	.74658203
.003	.68823242	.69213867	.69604492	.69995117	.70385742	.70776367	.71166992	.71557617	.71948242	.72338867	.72729492	.73120117	.73510742	.73901367	.74291992	.74682617
.004	.68847656	.69238281	.69628906	.70019531	.70410156	.70800781	.71191406	.71582031	.71972656	.72363281	.72753906	.73144531	.73535156	.73925781	.74316406	.74707031
.005	.68872070	.69262695	.69653320	.70043945	.70434570	.70825195	.71215820	.71606445	.71997070	.72387695	.72778320	.73168945	.73559570	.73950195	.74340820	.74731445
.006	.68896484	.69287109	.69677734	.70068359	.70458984	.70849609	.71240234	.71630859	.7202'484	.72412109	.72802734	.73193359	.73583984	.73973609	.74365234	.74755859
.007	.68920898	.69311523	.69702148	.70092773	.70483398	.70874023	.71264648	.71655273	.72045898	.72436523	.72827148	.73217773	.73608398	.73999023	.74389648	.74780273
.008	.68945313	.69335938	.69726563	.70117188	.70507813	.70898438	.71289063	.71679688	.72070313	.72460938	.72851563	.73242188	.73632813	.74023438	.74414063	.74804688
.009	.68969727	.69360352	.69750977	.70141602	.70532227	.70922852	.71313477	.71704102	.72094727	.72485352	.72875977	.73266602	.73657227	.74047852	.74438477	.74829102
.00A	.68994141	.69384766	.69775391	.70166016	.70556641	.70947266	.71337891	.71728516	.72119141	.72509766	.72900391	.73291016	.73681641	.74072266	.74462891	.74853516
.00B	.69018555	.69409180	.69799805	.70190430	.70581055	.70971680	.71362305	.71752930	.72143555	.72534180	.72924805	.73315430	.73706055	.74096680	.74487305	.74877930
.00C	.69042969	.69433594	.69824219	.70214844	.70605469	.70996094	.71386719	.71777344	.72167969	.72558594	.72949219	.73339844	.73730469	.74121094	.74511719	.74902344
.00D	.69067383	.69458008	.69848633	.70239258	.70629883	.71020508	.71411133	.71801758	.72192383	.72583008	.72973633	.73364258	.73754883	.74145508	.74536133	.74926758
.00E	.69091797	.69482422	.69873047	.70263672	.70654297	.71044922	.71435547	.71826172	.72216797	.72607422	.72998047	.73388672	.73779297	.74169922	.74560547	.74951172
.00F	.69116211	.69506836	.69897461	.70288086	.70678711	.71069336	.71459961	.71850586	.72241211	.72631836	.73022461	.73413086	.73803711	.74194336	.74584961	.74975586

Figure 28. Decimal to Hexadecimal Conversion Information (Part 4 of 5)

	.C0	.C1	.C2	.C3	.C4	.C5	.C6	.C7	.C8	.C9	.CA	.CB	.CC	.CD	.CE	.CF
.000	.75000000	.75390625	.75781250	.76171875	.76562500	.76953125	.77343750	.77734375	.78125000	.78515625	.78906250	.79296875	.79687500	.80078125	.80468750	.80859375
.001	.75024414	.75415039	.75805664	.76196289	.76586914	.76977539	.77368164	.77758789	.78149414	.78540039	.78930664	.79321289	.79711914	.80102539	.80493164	.80883789
.002	.75048828	.75439453	.75830078	.76220703	.76611328	.77001953	.77392578	.77783203	.78173828	.78564453	.78955078	.79345703	.79736328	.80126953	.80517578	.80908203
.003	.75073242	.75463867	.75854492	.76245117	.76635742	.77026367	.77416992	.77807617	.78198242	.78588867	.78979492	.79370117	.79760742	.80151367	.80541992	.80932617
.004	.75097656	.75488281	.75878906	.76269531	.76660156	.77050781	.77441406	.77832031	.78222656	.78613281	.79003906	.79394531	.79785156	.80175781	.80566406	.80957031
.005	.75122070	.75512695	.75903320	.76293945	.76684570	.77075195	.77465820	.77856445	.78247070	.78637695	.79028320	.79418945	.79809570	.80200195	.80590820	.80981445
.006	.75146484	.75537109	.75927734	.76318359	.76708984	.77099609	.77490234	.77880859	.78271484	.78662109	.79052734	.79443359	.79833984	.80224609	.80615234	.81005859
.007	.75170898	.75561523	.75952148	.76342773	.76733398	.77124023	.77514648	.77905273	.78295898	.78686523	.79077148	.79467773	.79858398	.80249023	.80639648	.81030273
.008	.75195313	.75585938	.75976563	.76367188	.76757813	.77148438	.77539063	.77929688	.78320313	.78710938	.79101563	.79492188	.79882813	.80273438	.80664063	.81054688
.009	.75219727	.75610352	.76000977	.76391602	.76782227	.77172852	.77563477	.77954102	.78344727	.78735352	.79125977	.79516602	.79907227	.80297852	.80688477	.81079102
.00A	.75244141	.75634766	.76025391	.76416016	.76806641	.77197266	.77587891	.77978516	.78369141	.78759766	.79150391	.79541016	.79931641	.80322266	.80712891	.81103516
.00B	.75268555	.75659180	.76049805	.76440430	.76831055	.77221680	.77612305	.78002930	.78393555	.78784180	.79174805	.79565430	.79956055	.80346680	.80737305	.81127930
.00C	.75292969	.75683594	.76074219	.76464844	.76855469	.77246094	.77636719	.78027344	.78417969	.78808594	.79199219	.79589844	.79980469	.80371094	.80761719	.81152344
.00D	.75317383	.75708008	.76098633	.76489258	.76879883	.77270508	.77661133	.78051758	.78442383	.78833008	.79223633	.79614258	.80004883	.80395508	.80786133	.81176758
.00E	.75341797	.75732422	.76123047	.76513672	.76904297	.77294922	.77685547	.78076172	.78466797	.78857422	.79248047	.79638672	.80029297	.80419922	.80810547	.81201172
.00F	.75366211	.75756836	.76147461	.76538086	.76928711	.77319336	.77709961	.78100586	.78491211	.78881836	.79272461	.79663086	.80053711	.80444336	.80834961	.81225586

	.D0	.D1	.D2	.D3	.D4	.D5	.D6	.D7	.D8	.D9	.DA	.DB	.DC	.DD	.DE	.DF
.000	.81250000	.81640625	.82031250	.82421875	.82812500	.83203125	.83593750	.83984375	.84375000	.84765625	.85156250	.85546875	.85937500	.86328125	.86718750	.87109375
.001	.81274414	.81665039	.82055664	.82446289	.82836914	.83227539	.83618164	.84008789	.84399414	.84790039	.85180664	.85571289	.85961914	.86352539	.86743164	.87133789
.002	.81298828	.81689453	.82080078	.82470703	.82861328	.83251953	.83642578	.84033203	.84423828	.84814453	.85205078	.85595703	.85986328	.86376953	.86767578	.87158203
.003	.81323242	.81713867	.82104492	.82495117	.82885742	.83276367	.83666992	.84057617	.84448242	.84838867	.85229492	.85620117	.86010742	.86401367	.86791992	.87182617
.004	.81347656	.81738281	.82128906	.82519531	.82910156	.83300781	.83691406	.84082031	.84472656	.84863281	.85253906	.85644531	.86035156	.86425781	.86816406	.87207031
.005	.81372070	.81762695	.82153320	.82543945	.82934570	.83325195	.83715820	.84106445	.84497070	.84887695	.85278320	.85668945	.86059570	.86450195	.86840820	.87231445
.006	.81396484	.81787109	.82177734	.82568359	.82958984	.83349609	.83740234	.84130859	.84521484	.84912109	.85302734	.85693359	.86083984	.86474609	.86865234	.87255859
.007	.81420898	.81811523	.82202148	.82592773	.82983398	.83374023	.83764648	.84155273	.84545898	.84936523	.85327148	.85717773	.86108398	.86499023	.86889648	.87280273
.008	.81445313	.81835938	.82226563	.82617188	.83007813	.83398438	.83789063	.84179688	.84570313	.84960938	.85351563	.85742188	.86132813	.86523438	.86914063	.87304688
.009	.81469727	.81860352	.82250977	.82641602	.83032227	.83422852	.83813477	.84204102	.84594727	.84985352	.85375977	.85766602	.86157227	.86547852	.86938477	.87329102
.00A	.81494141	.81884766	.82275391	.82666016	.83056641	.83447266	.83837891	.84228516	.84619141	.85009766	.85400391	.85791016	.86181641	.86572266	.86962891	.87353516
.00B	.81518555	.81909180	.82299805	.82690430	.83081055	.83471680	.83862305	.84252930	.84643555	.85034180	.85424805	.85815430	.86206055	.86596680	.86987305	.87377930
.00C	.81542969	.81933594	.82324219	.82714844	.83105469	.83496094	.83886719	.84277344	.84667969	.85058594	.85449219	.85839844	.86230469	.86621094	.87011719	.87402344
.00D	.81567383	.81958008	.82348633	.82739258	.83129883	.83520508	.83911133	.84301758	.84692383	.85083008	.85473633	.85864258	.86254883	.86645508	.87036133	.87426758
.00E	.81591797	.81982422	.82373047	.82763672	.83154297	.83544922	.83935547	.84326172	.84716797	.85107422	.85498047	.85888672	.86279297	.86669922	.87060547	.87451172
.00F	.81616211	.82006836	.82397461	.82788086	.83178711	.83569336	.83959961	.84350586	.84741211	.85131836	.85522461	.85913086	.86303711	.86694336	.87084961	.87475586

	.E0	.E1	.E2	.E3	.E4	.E5	.E6	.E7	.E8	.E9	.EA	.EB	.EC	.ED	.EE	.EF
.000	.87500000	.87890625	.88281250	.88671875	.89062500	.89453125	.89843750	.90234375	.90625000	.91015625	.91406250	.91796875	.92187500	.92578125	.92968750	.93359375
.001	.87524414	.87915039	.88305664	.88696289	.89086914	.89477539	.89868164	.90258789	.90649414	.91040039	.91430664	.91821289	.92211914	.92602539	.92993164	.93383789
.002	.87548828	.87939453	.88330078	.88720703	.89111328	.89501953	.89892578	.90283203	.90673828	.91064453	.91455078	.91845703	.92236328	.92626953	.93017578	.93408203
.003	.87573242	.87963867	.88354492	.88745117	.89135742	.89526367	.89916992	.90307617	.90698242	.91088867	.91479492	.91870117	.92260742	.92651367	.93041992	.93432617
.004	.87597656	.87988281	.88378906	.88769531	.89160156	.89550781	.89941406	.90332031	.90722656	.91113281	.91503906	.91894531	.92285156	.92675781	.93066406	.93457031
.005	.87622070	.88012695	.88403320	.88793945	.89184570	.89575195	.89965820	.90356445	.90747070	.91137695	.91528320	.91918945	.92309570	.92700195	.93090820	.93481445
.006	.87646484	.88037109	.88427734	.88818359	.89208984	.89599609	.89990234	.90380859	.90771484	.91162109	.91552734	.91943359	.92333984	.92724609	.93115234	.93505859
.007	.87670898	.88061523	.88452148	.88842773	.89233398	.89624023	.90014648	.90405273	.90795898	.91186523	.91577148	.91967773	.92358398	.92749023	.93139648	.93530273
.008	.87695313	.88085938	.88476563	.88867188	.89257813	.89648438	.90039063	.90429688	.90820313	.91210938	.91601563	.91992188	.92382813	.92773438	.93164063	.93554688
.009	.87719727	.88110352	.88500977	.88891602	.89282227	.89672852	.90063477	.90454102	.90844727	.91235352	.91625977	.92016602	.92407227	.92797852	.93188477	.93579102
.00A	.87744141	.88134766	.88525391	.88916016	.89306641	.89697266	.90087891	.90478516	.90869141	.91259766	.91650391	.92041016	.92431641	.92822266	.93212891	.93603516
.00B	.87768555	.88159180	.88549805	.88940430	.89331055	.89721680	.90112305	.90502930	.90893555	.91284180	.91674805	.92065430	.92456055	.92846680	.93237305	.93627930
.00C	.87792969	.88183594	.88574219	.88964844	.89355469	.89746094	.90136719	.90527344	.90917969	.91308594	.91699219	.92089844	.92480469	.92871094	.93261719	.93652344
.00D	.87817383	.88208008	.88598633	.88989258	.89379883	.89770508	.90161133	.90551758	.90942383	.91333008	.91723633	.92114258	.92504883	.92895508	.93286133	.93676758
.00E	.87841797	.88232422	.88623047	.89013672	.89404297	.89794922	.90185547	.90576172	.90966797	.91357422	.91748047	.92138672	.92529297	.92919922	.93310547	.93701172
.00F	.87866211	.88256836	.88647461	.89038086	.89428711	.89819336	.90209961	.90600586	.90991211	.91381836	.91772461	.92163086	.92553711	.92944336	.93334961	.93725586

	.F0	.F1	.F2	.F3	.F4	.F5	.F6	.F7	.F8	.F9	.FA	.FB	.FC	.FD	.FE	.FF
.000	.93750000	.94140625	.94531250	.94921875	.95312500	.95703125	.96093750	.96484375	.96875000	.97265625	.97656250	.98046875	.98437500	.98828125	.99218750	.99609375
.001	.93774414	.94165039	.94555664	.94946289	.95336914	.95727539	.96118164	.96508789	.96899414	.97290039	.97680664	.98071289	.98461914	.98852539	.99243164	.99633789
.002	.93798828	.94189453	.94580078	.94970703	.95361328	.95751953	.96142578	.96533203	.96923828	.97314453	.97705078	.98095703	.98486328	.98876953	.99267578	.99658203
.003	.93823242	.94213867	.94604492	.94995117	.95385742	.95776367	.96166992	.96557617	.96948242	.97338867	.97729492	.98120117	.98510742	.98901367	.99291992	.99682617
.004	.93847656	.94238281	.94628906	.95019531	.95410156	.95800781	.96191406	.96582031	.96972656	.97363281	.97753906	.98144531	.98535156	.98925781	.99316406	.99707031
.005	.93872070	.94262695	.94653320	.95043945	.95434570	.95825195	.96215820	.96606445	.96997070	.97387695	.97778320	.98168945	.98559570	.98950195	.99340820	.99731445
.006	.93896484	.94287109	.94677734	.95068359	.95458984	.95849609	.96240234	.96630859	.97021484	.97412109	.97802734	.98193359	.98583984	.98974609	.99365234	.99755859
.007	.93920898	.94311523	.94702148	.95092773	.95483398	.95874023	.96264648	.96655273	.97045898	.97436523	.97827148	.98217773	.98608398	.98999023	.99389648	.99780273
.008	.93945313	.94335938	.94726563	.95117188	.95507813	.95898438	.96289063	.96679688	.97070313	.97460938	.97851563	.98242188	.98632813	.99023438	.99414063	.99804688
.009	.93969727	.94360352	.94750977	.95141602	.95532227	.95922852	.96313477	.96704102	.97094727	.97485352	.97875977	.98266602	.98657227	.99047852	.99438477	.99829102
.00A	.93994141	.94384766	.94775391	.95166016	.95556641	.95947266	.96337891	.96728516	.97119141	.97509766	.97900391	.98291016	.98681641	.99072266	.99462891	.99853516
.00B	.94018555	.94409180	.94799805	.95190430	.95581055	.95971680	.96362305	.96752930	.97143555	.97534180	.97924805	.98315430	.98706055	.99096680	.99487305	.99877930
.00C	.94042969	.94433594	.94824219	.95214844	.95605469	.95996094	.96386719	.96777344	.97167969	.97558594	.97949219	.98339844	.98730469	.99121094	.99511719	.99902344
.00D	.94067383	.94458008	.94848633	.95239258	.95629883	.96020508	.96411133	.96801758	.97192383	.97583008	.97973633	.98364258	.98754883	.99145508	.99536133	.99926758
.00E	.94091797	.94482422	.94873047	.95263672	.95654297	.96044922	.96435547	.96826172	.97216797	.97607422	.97998047	.98388672	.98779297	.99169922	.99560547	.99951172
.00F	.94116211	.94506836	.94897461	.95288086	.95678711	.96069336	.96459961	.96850586	.97241211	.97631836	.98022461	.98413086	.98803711	.99194336	.99584961	.99975586

Figure 28. Decimal to Hexadecimal Conversion Information (Part 5 of 5)

	Variable Symbols													Attributes						
		Global SET Symbols			Local SET Symbols			System Variable Symbols												
Requirement	Symbolic Parameter	SETA	SETB	SETC	SETA	SETB	SETC	&SYSNDX	&SYSECT	&SYSLIST	&SYSPARM	&SYSDATE	&SYSTIME	Type	Length	Scaling	Integer	Count	Number	Sequence Symbol
MACRO																				
Prototype Statement	Name Operand																			
GBLA		Operand																		
GBLB			Operand																	
GBLC				Operand																
LCLA					Operand															
LCLB						Operand														
LCLC							Operand													
Model Statement	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Operand	Operand							Name
SETA	Operand[2]	Name Operand	Operand[3]	Operand[9]	Name Operand	Operand[3]	Operand[9]	Operand		Operand[2]	Operand[9]				Operand	Operand	Operand	Operand	Operand	
SETB	Operand[6]	Operand[6]	Name Operand	Operand[6]	Operand[6]	Nume Operand	Operand[6]	Operand[6]	Operand[4]	Operand[6]	Operand[6]			Operand[4]	Operand[5]	Operand[5]	Operand[5]	Operand[5]	Operand[5]	
SETC	Operand	Operand[7]	Operand[8]	Name Operand	Operand[7]	Operand[8]	Name Operand	Operand	Operand	Operand	Operand	Operand	Operand	Operand						
AIF	Operand[6]	Operand[6]	Operand	Operand[6]	Operand[6]	Operand	Operand[6]	Operand[6]	Operand[4]	Operand[6]	Operand[6]			Operand[4]	Operand[5]	Operand[5]	Operand[5]	Operand[5]	Operand[5]	Name Operand
AGO																				Name Operand
ACTR	Operand[2]	Operand	Operand[3]	Operand[2]	Operand	Operand[3]	Operand[2]	Operand		Operand[2]	Operand[2]				Operand	Operand	Operand	Operand	Operand	
ANOP																				Name
MEXIT																				Name
MNOTE	Operand	Operand	Operand	Operand	Operand	Operand	Operand	Operand	Operand	Operand	Operand	Operand	Operand							Name
MEND																				Name
Outer Macro		Name Operand	Name Operand	Name Operand	Name Operand	Name Operand	Name Operand				Name Operand	Operand	Operand							Name
Inner Macro	Name Operand	Name, Operand	Name Operand	Name Operand	Name Operand	Name Operand	Name Operand	Name Operand	Name Operand	Name Operand	Name Operand	Operand	Operand							Name
Assembler Language Statement		Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand	Name Operation Operand													Name

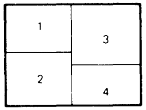
1. Variable symbols in macro instructions are replaced by meir values before processing.
2. Only if value is self-defining term.
3. Converted to arithmetic +1 or +0.
4. Only in character relations.
5. Only in arithmetic relations.
6. Only in arithmetic or character relations.
7. Converted to unsigned number.
8. Converted to character 1 or 0.
9. Only if one to one decimal digits (from 0 through 2, 147, 483, 647).

Figure 29. Assembler Macro Language Statements

Extended Binary-Coded-Decimal Interchange Code (EBCDIC)

The following 256 position table, outlined by the heavy black lines, shows the graphic characters and control character representations for EBCDIC. The bit position numbers, bit patterns, hexadecimal representations and card hole patterns for these and other possible EBCDIC characters are also shown.

To find the card hole patterns for most characters, partition the 256 position table into four blocks as follows



Block 1 Zone punches at top of table; digit punches at left
Block 2 Zone punches at bottom of table; digit punches at left
Block 3 Zone punches at top of table, digit punches at right
Block 4 Zone punches at bottom of table, digit punches at right

Fifteen positions in the table are exceptions to the above arrangement. These positions are indicated by small numbers in the upper right corners of their boxes in the table. The card hole patterns for these positions are given at the bottom of the table. Bit position numbers, bit patterns, and hexadecimal representations for these positions are found in the usual manner.

Following are some examples of the use of the EBCDIC chart:

Character	Type	Bit Pattern	Hex	Hole Pattern	
				Zone Punches	Digit Punches
PF	Control Character	00 00 0100	04	12 - 9	- 4
%	Special Graphic	01 10 1100	6C	0	- 8 - 4
R	Upper Case	11 01 1001	D9	11	- 9
a	Lower Case	10 00 0001	81	12 - 0	- 1
	Control Character, function not yet assigned	00 11 0000	30	12 - 11 - 0 - 9	- 8 - 1

Bit Positions
01 23 4567

EBCDIC Codes

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	Digit Punches	00				01				10				11					Bit Positions 0,1
			00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11		Bit Positions 2,3
			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		First Hexadecimal Digit
			12				12	12		12	12	12		12	12					Zone Punches
				11				11	11	11		11	11	11		11				
					0		0		0	0	0		0	0			0			
			9	9	9	9	9	9	9	9										Digit Punches
0000	0	8-1	① NUL	② DLE	③ DS	④	⑤ SP	⑥ &	⑦ -	⑧					⑨	⑩	⑪	⑫ 0	8-1	
0001	1	1	SOH	DC1	SOS				/ ⑬		a	j			A	J	⑭	1	1	
0010	2	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2	2	
0011	3	3	ETX	TM							c	l	t		C	L	T	3	3	
0100	4	4	PF	RES	BYP	PN					d	m	u		D	M	U	4	4	
0101	5	5	HT	NL	LF	RS					e	n	v		E	N	V	5	5	
0110	6	6	LC	BS	ETB	UC					f	o	w		F	O	W	6	6	
0111	7	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7	7	
1000	8	8		CAN							h	q	y		H	Q	Y	8	8	
1001	9	8-1		EM							i	r	z		I	R	Z	9	9	
1010	A	8-2	SMM	CC	SM		¢	!	⑮	:									8-2	
1011	B	8-3	VT	CU1	CU2	CU3	.	$	,	#									8-3	
1100	C	8-4	FF	IFS		DC4	<	*	%	@									8-4	
1101	D	8-5	CR	IGS	ENQ	NAK	(	)	_	'									8-5	
1110	E	8-6	SO	IRS	ACK		+	;	>	=									8-6	
1111	F	8-7	SI	IUS	BEL	SUB	|	¬	?	"									8-7	
			12				12				12	12		12	12	12		12		Zone Punches
				11				11				11	11	11		11	11	11		
					0				0		0		0	0	0		0	0		
			9	9	9	9									9	9	9	9		

Card Hole Patterns

① 12-0-9-8-1	⑤ No Punches	⑨ 12-0	⑬ 0-1
② 12-11-9-8-1	⑥ 12	⑩ 11-0	⑭ 11-0-9-1
③ 11-0-9-8-1	⑦ 11	⑪ 0-8-2	⑮ 12-11
④ 12-11-0-9-8-1	⑧ 12-11-0	⑫ 0	

Control Character Representations

ACK	Acknowledge
BEL	Bell
BS	Backspace
BYP	Bypass
CAN	Cancel
CC	Cursor Control
CR	Carriage Return
CU1	Customer Use 1
CU2	Customer Use 2
CU3	Customer Use 3
DC1	Device Control 1
DC2	Device Control 2
DC4	Device Control 4
DEL	Delete
DLE	Data Link Escape
DS	Digit Select
EM	End of Medium
ENQ	Enquiry
EOT	End of Transmission
ESC	Escape
ETB	End of Transmission Block
ETX	End of Text
FF	Form Feed
FS	Field Separator
HT	Horizontal Tab
IFS	Interchange File Separator
IGS	Interchange Group Separator
IL	Idle
IRS	Interchange Record Separator
IUS	Interchange Unit Separator
LC	Lower Case
LF	Line Feed
NAK	Negative Acknowledge
NL	New Line
NUL	Null
PF	Punch Off
PN	Punch On
RES	Restore
RS	Reader Stop
SI	Shift In
SM	Set Mode
SMM	Start of Manual Message
SO	Shift Out
SOH	Start of Heading
SOS	Start of Significance
SP	Space
STX	Start of Text
SUB	Substitute
SYN	Synchronous Idle
TM	Tape Mark
UC	Upper Case
VT	Vertical Tab

Special Graphic Characters

¢	Cent Sign
.	Period, Decimal Point
<	Less-than Sign
(	Left Parenthesis
+	Plus Sign
|	Logical OR
&	Ampersand
!	Exclamation Point
$	Dollar Sign
*	Asterisk
)	Right Parenthesis
;	Semicolon
¬	Logical NOT
-	Minus Sign, Hyphen
/	Slash
,	Comma
%	Percent
_	Underscore
>	Greater-than Sign
?	Question Mark
:	Colon
#	Number Sign
@	At Sign
'	Prime, Apostrophe
=	Equal Sign
"	Quotation Mark

Figure 30. Extended Binary Coded Decimal Interchange Code (EBCDIC)

GX20-1926-3

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Printed in U.S.A.