GA22-6963-1
File No. S/370-13

Systems

# 7080 Compatibility Feature for IBM System/370 Models 165, 165 II, and 168

IBM

# Preface

The IBM 7080 compatibility feature, operating with a complementary set of programs, constitutes an emulator. Basically, an emulator is a hardware-aided simulator program. It operates much like an interpretive-routine simulator program, but at a speed about five times faster. With the emulator, System/370 Models 165, 165 II, and 168 can run 7080 programs at speeds that, in general, equal or exceed those of the original systems. Operating in a System/370, the emulator relieves the user of the need for conversion of existing 7080 programs.

## Prerequisite Publications

The reader of this publication should be familiar with the information contained in the following publications:

*IBM 7080 Principles of Operation*, GA22-6560
*IBM System/370 System Summary*, GA22-7001
*IBM System/370 Principles of Operation*, GA22-7000
*7080 OS Emulator on Models 165/168 Reference*, GC27-6952.

## Organization of This Publication

The information in this manual is organized as follows:

Following the Contents is a list of abbreviations and notation forms used in this publication.

The Introduction contains a general description of emulation and provides basic information on the compatibility feature and the emulator program.

The section "Emulator Organization" describes the functions of the compatibility feature and of the emulator program and their relationship to each other. This section also includes information on acceptable data formats and program/feature communication.

The section "Emulator Instruction Set" describes the format, application, and action of each instruction provided by the compatibility feature.

The Appendix contains a glossary, a summary of instructions, a table of 7080 indicator and logout addresses, and code conversion tables.

*Second Edition* (June 1973)
This major revision obsoletes GA22-6963-0 and expands the scope of this manual to include System/370 Models 165 II and 168. Any significant technical change (addition, deletion, or revision) is indicated by a vertical line to the left of the change.

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/360 and System/370 Bibliography*, GA22-6822, and associated technical newsletters for the editions that are applicable and current.

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

This manual has been prepared by the IBM System Products Division, Product Publications, Dept. B98, PO Box 390, Poughkeepsie, N.Y. 12602. A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be sent to the above address. Comments become the property of IBM.

# Contents

## Abbreviations

| | |
|---|---|
| ACC | accumulator |
| ALU | arithmetic and logic unit |
| ASU | auxiliary storage unit |
| BCD | binary-coded decimal |
| CASU | channel auxiliary storage unit |
| C(x) | contents of x (example: C(R1)) |
| CB | coded binary |
| CPU | central processing unit |
| CS | control storage |
| EBCDIC | extended binary-coded decimal interchange code |
| E80 | emulated 7080 |
| FPR | floating-point register |
| GR | general register |
| hex | hexadecimal |
| I | instruction |
| IA | instruction address |
| IAR | initial address register |
| IC | instruction counter |
| I/O | input/output |
| K | = 1,024 (in System/370) |
| MAC I | memory address counter I |
| MAC II | memory address counter II |
| MCW | maintenance control word |
| op code | operation code |
| OS | operating system |
| P | parity |
| ROS | read-only storage |
| RR | System/370 register-to-register instruction format |
| RS | System/370 register-and-storage instruction format |
| SAL | serial adder latch |
| SAR | storage address register |
| SPC | starting point counter |
| SR | storage register |
| SS | System/370 storage-to-storage instruction format |
| SSR | storage select register |
| WCS | writable control storage |

| | |
|---|---|
| R1 | The first operand field of an RR- or RS-format instruction, used primarily to specify a general register. |
| R2 | The second operand field of an RR-format instruction, used primarily to specify a general register. |
| R3 | The third operand field of an RS-format instruction, used primarily to specify an op-code modifier. |
| B2, D2 | The second operand field of an RS-format instruction. B2 specifies a general register, and D2 specifies a displacement from the contents of the B2 register. |

When used with its associated emulator program, the 7080 compatibility feature facilitates the transition from the 7080 Data Processing System to System/370. The compatibility feature adds special instructions and registers to System/370. The emulator program employs these facilities and those of the System/370 universal instruction set to simulate 7080 instructions. The compatibility feature and the emulator program together constitute the emulator.

Through the emulator, a System/370 Model 165, 165 II, or 168 interprets and executes programs written for the IBM 7080. Internal performance is about twice that of the 7080, but depends on the instructions used.

Installation of the compatibility feature does not affect normal operation of the Model 165, 165 II, or 168.

### Purpose of the Emulator

The 7080 emulator is an aid to transition from the 7080 Data Processing System to System/370. With the emulator under OS control, Models 165, 165 II, and 168 execute current 7080 programs with little or no reprogramming. Also under OS control, the emulator can be multiprogrammed with other user problem programs.

### Modes of Operation

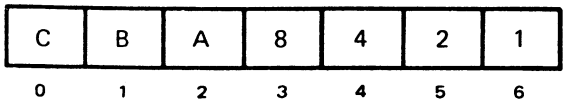
The emulator is capable of performing functions of the 7080 CPU while in 7080 mode, 705 I/II mode, or 705 III mode.

### Character Conversion and Word Correspondence

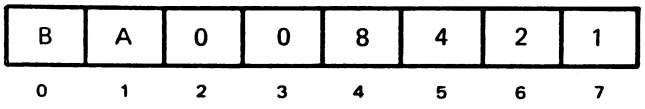
*Character Conversion*

A 6-bit 7080 character is mapped into bits 0-1 and 4-7 of an 8-bit System/370 byte, with bits 2 and 3 set to zero. The parity bit of the 7080 character is dropped and odd parity bit of System/370 is assumed. The character is then referred to as an E80 character. (E80 refers to anything in System/370 that is made to correspond to the 7080 system.)

*7080 Character*



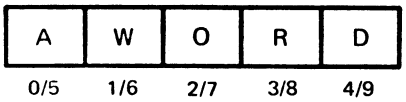

*E80 Character*



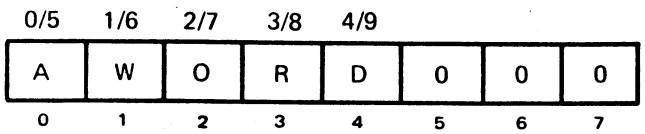

*Memory Word Correspondence*

A five-character 7080 memory word is mapped into bytes 0-4 of a System/370 doubleword. Bytes 5-7 are set to zero.

*7080 Memory Word*




*E80 Memory Word*




*Storage Word Correspondence*

An eight-character 7080 storage word is mapped into a System/370 doubleword.

### Memory and Storage Mapping

*E80 Memory*

Since five 7080 characters require eight bytes, the E80 memory requires more System/370 storage bytes than the corresponding number of 7080 characters. Also, the addressing scheme used to convert 7080 addresses causes blocks of System/370 storage to be left unused. As a result, 128K (131,072) bytes of System/370 storage are needed to emulate 80,000 7080 characters. Since the 7080 emulator operates under OS control, the base address for 7080 memory (0000) is relocatable and is kept in FPR 2 bytes 4-7. The 7080 memory is loaded from this starting point.

*E80 Storage*

A 7080 storage address with a starting-point counter setting of 0000 is taken to be the origin. Thus, storage bank 0 occupies storage addresses 000-255. Corresponding to 7080 storage, E80 storage originates at the address in FPR 4 bytes 4-7 and occupies 2,048 consecutive bytes to provide 7080 storage banks 0-4. Banks 5-7 are used for programs and for microprogramming. The emulated 7080 storage must be maintained in Model 165, 165 II, or 168 main storage during the execution of a 7080 program All E80 operations using E80 storage stay within the bank designated at the beginning of the operation. and the E80 storage address wraps at the end of the 256-byte E80 storage block. If auxiliary storage units (ASU's) are designated, the appropriate addresses are developed to point to the correct locations in E80 storage.

The branch table is composed of E80 storage and three other blocks of System/370 storage. All four blocks are contiguous in System/370 storage, and the location of the branch table is specified by the origin of the E80 storage block (FPR 4 bytes 4-7). The blocks and their sequence of loading are:

1. E80 multiply, in which 80 bytes of storage for the emulator multiply algorithm are stored at the address 80 less than the one in FPR 4 bytes 4-7.
2. E80 storage.

3. Select table, in which 2K (2,048) bytes are used in the conversion of 7080 I/O addresses to System/370 I/O device addresses.
4. Subroutine area, in which 1,024 bytes are reserved, with 16 bytes allocated for each 7080 instruction subroutine. (Twelve of the subroutines require more space and branch out of the table.)

### Register Assignments

The internal address registers of the 7080 are emulated by System/370 general registers (GR's). Their contents are dynamically updated to reflect that of the 7080. Their assignments along with the assignments of the floating-point registers (FPR's) are as follows:

| *GR* | *Function or Contents* | *Format* |
|---|---|---|
| 0-3 | Work registers for emulator instructions | |
| 4 | Memory address counter I (MAC I) | Zoned-address type B* |
| 5 | Storage address register (SAR) | Coded binary |
| 6 | Initial address register (IAR) | Zoned-address type A* |
| 7 | Byte 0: storage bank<br>bytes 1-3: starting-point counter (SPC) | Coded binary |
| 8 | Subroutine branch address | |
| 9 | MAC I | Coded binary |
| 10 | Unused | |
| 11 | Memory address counter II (MAC II) | Zoned-address type B |
| 12 | Bytes 0-2, zeros; byte 3 bits 0-3 storage select register (SSR), byte 3 bits 4-7 0000 | |
| 13-15 | Unused | |

*Zoned-address type A refers to bytes 1-4 of the E80 memory word (a System/370 doubleword) with E80 ASU zones set to zeros. All bytes representing zeros have the numeric portion (bits 4-7) set to binary 1010.
Zoned-address type B is similar to zoned-address type A, except that zeros are represented by the numeric portion of the byte set to binary 0000.

| *FPR* | *Function or Contents* |
|---|---|
| 0 | Bytes 0-3: emulator dynamic status<br>Bytes 4-7: emulator hardware status |
| 2 | Byte 0: DIL count.<br>Bytes 1-3: base address for unusual 7080 conditions**.<br>Bytes 4-7: E80 memory relocation value (must be used on a doubleword boundary) |
| 4 | Bytes 0-3: 7080 IC (zoned-address type B).<br>Bytes 4-7: branch table base address. (Each bank pointed to in the select table must be on a hex 100 boundary.) |
| 6 | Work register. |

**This address is incremented by a fixed value corresponding to the situation causing the 7080 unusual condition branch, as follows:

| *Increment (hex)* | *Reason* |
|---|---|
| +00 | Unused |
| +10 | 904-905 check switches on |
| +20 | 7080 operation check |
| +30 | DIL interrupt |
| +40 | DIL count = zero |
| +50 | 904 check on ERND instruction |
| +60 | No bit character on LB instruction |
| +70 | Restricted operation |
| +80 | 7080 hang condition |

### Memory and Storage Address Conversion

#### *Memory Address Conversion*

Every 7080 address between 00000 and 159,999, inclusive, is changed into a unique 24-bit System/370 address ranging from 000000 to 03FFFC. Due to the algorithm of the conversion, only characters of the same 7080 word have contiguous addresses within a System/370 doubleword. Furthermore, sequential 7080 addresses do not necessarily yield sequential System/370 addresses after conversion.

The conversion leaves 64 areas of 96 contiguous unused bytes. These addresses may be identified as follows:

Unused address (in hex) WXYZZ plus E80 base (FPR 3)
where
W = 0, 1, 2, or 3
X = 3, 7, B, or F
Y = 7 or F
ZZ = 00 to 5F or 80 to DF

#### *Storage Address Conversion*

The storage addresses are in System/370 address format, beginning at the pointer in FPR 4 bytes 4-7. Each bank has a starting address hex 100 (256 decimal) greater than the starting address of the bank preceding it. A special emulator instruction, ESPC (Set Starting Point Counter), is provided to change a 7080 storage address into its corresponding E80 storage address.

### Status Indicators

The status of a 7080 changes dynamically during execution of stored programs, and program results are affected by changes in status. The emulator maintains status in triggers, in a local storage word, and in main storage words.

#### *Triggers*

The following triggers are added for emulation. Special microorders are used to alter and interrogate them.

| *System/370 Status Triggers* | *Corresponding 7080 Status* |
|---|---|
| 7080 mode | 7080 mode |
| 80K memory | For 7080 system that has only 80K memory |
| 40K memory | 40K switch |
| 904 check switch | 904 check switch |
| 905 check switch | 905 check switch |
| 705 I/II mode | 705 I/II switch |
| 7080 interrupt flag | (No 7080 correspondence) |

| *System/370 Status Triggers* | *Corresponding 7080 Status* |
|---|---|
| DIL counter flag | (No 7080 correspondence) |
| Emulator internal work triggers: ASU ≠ 0, DNZT, digit carry, zone carry, do index, complement, recomplement, increment factor 1, increment factor 2, three triggers for interrupt and retry, and emulator status valid. | (No direct correspondence) |

*The Emulator Status Valid Trigger* is one of the more important triggers in the emulator. When set, it signifies that the status of the 7080 is loaded into the status registers, and that Model 165, 165 II, and 168 recognize the emulator SS-type instruction, executing the secondary op code if it is valid. If the status valid trigger is not on, the SS-instruction op code (E9) goes into execution to load the hardware status, turn on the status valid trigger, decrement the System/370 IC by 6, and end operation. This causes the op code to be entered again, and the status valid trigger is set. When a System/370 interruption occurs, all the 7080 status triggers are preserved in FPR 0, and this trigger is reset. Upon reentering emulation, the status triggers are restored and the status value trigger is again set.

### *Local Storage Word*

The second half of the 7080 status word is stored in FPR 0. The format is the same as that stored in channel auxiliary storage unit 15 (CASU 15) on an interruption in the 7080. Being located in FPR 0 permits these *dynamic status indicators* to be easily accessed. When an interruption occurs, storing of the old status and restoring of the new status can be done with a minimum of programming. The bit correspondence is listed in the following dynamic status indicators.

*Dynamic Status Indicators*

| *System/370 FPR 0 Bit* | *Bit Status* | *E80 Status* |
|---|---|---|
| 0 | 1 | Set to one |
| 1 | x | ASU 0 (for x = 1) |
| 2-3 | 00 | None |
| 4 | 0 | Set to zero |
| 5 | x | ACC 0 (for x = 1) |
| 6 | x | ASU ( ) (for x = 1) |
| 7 | x | ACC ( ) (for x = 1) |
| 8 | 1 | Set to one |
| 9 | x | 903 (for x = 1) |
| 10-11 | 00 | None |
| 12 | 0 | Set to zero |
| 13 | x | 902 (for x = 1) |
| 14 | x | 901 (for x = 1) |
| 15 | x | 900 (for x = 1) |
| 16 | 1 | Set to one |
| 17 | x | 7080 mode (for x = 1) |
| 18-19 | 00 | None |
| 20 | 0 | Set to zero |
| 21 | x | ANY (for x = 1) |
| 22 | x | 905 (for x = 1) |
| 23 | x | 904 (for x = 1) |
| 24 | 1 | Set to one |
| 25 | 0 | Set to zero |
| 26-27 | 00 | None |
| 28 | 0 | Set to zero |
| 29 | 0 | Set to zero |
| 30 | x | Low compare (for x = 1) |
| 31 | x | High compare (for x = 1) |

### *Main Storage Words*

Status information kept in main storage (memory) words includes the I/O indicators, I/O ready indicators, interrupt mode, the interrupt program, read-while-write, the nonstop switch, the I/O interpret switch, the alteration switches, check indicators, all of the interruption requests, and duplicates of status as maintained in hardware triggers and CASU 15. This status information is initialized and maintained by the emulation program.

## Input/Output and Console Operations

Input/output and console operations are performed by the emulation-program I/O routines. Two groups of System/370 operations (load buffer and unload buffer) are designed to handle the code conversion and I/O buffer manipulation tasks required by the program. These operations, which are discussed later, supply the bridge between the I/O buffers and E80 main memory. A DIL-counter operation is provided to give the program the ability to execute a fixed number of DIL's and then branch to a predetermined System/370 storage location. This gives the emulator an added level of flexibility and power in programming.

The E80 console operations are controlled by simulation routines.

## Exceptions and Restrictions

In general, any 7080 operation that is beyond the defined limits of the basic 7080 is beyond the defined limits of emulation. For example, multiplication of nonnumeric fields in the 7080 produces unpredictable results. Emulation does not necessarily produce the same results. Any other exceptions or restrictions are listed with the instruction descriptions. Exceptions associated with program-simulated instructions are specified in the external specifications of the 7080 emulation program.

Three types of exceptions are referred to in this publication: operation, specification, and access. For the Model 165, access exceptions include both the protection and addressing exceptions. For Models 165 II and 168, access exceptions include not only the protection and addressing exceptions but also the segment-translation, page-translation, and translation-specification exceptions.

### Emulator Features and Instructions

To facilitate emulation, special features and instructions are used which contribute to the speed and ease of programming. The instructions are classified as the general and special emulator instructions. The general instructions assist emulation by performing functions which partially contribute to the duplication of specific 7080 operations (such as 7080 I-time and select register decoding). The special instructions exactly duplicate the functions of 7080 instructions (such as ADD and LOD).

### Emulator Hardware

Some of the differences between System/370 and the 7080 require additional features to implement functions not immediately available in System/370. For example, the 7080 serial adder handles zoned decimal data as well as special characters, whereas the System/370 serial adder adds only packed decimal data; and the 7080 memory address registers are in BCD format, whereas the System/370 address registers are in binary format.

The following devices are used to compensate for some of the differences.

#### *Operation-Code Branch-Address Decoder*

The 7080 6-bit operation code (op code) is used to form a 24-bit binary address. This address points to an emulator program routine that executes the 7080 operation.

#### *SPC Decoder*

The address portion of the 7080 Set Starting Point Counter (SPC) instruction is used to form a 24-bit binary address that points to an E80 storage location in System/370 storage.

#### *UNSPC Decoder*

A System/370 storage address is converted to an SPC value (zoned-address type A)

#### *Address Incrementer-Decrementer*

Many 7080 instructions can be more readily emulated if their addresses are preserved and updated in zoned-address format. The address incrementer-decrementer can perform four different functions: increment zoned address by five, by one, and by zero or decrement zoned address by five. The following rules govern the operation:

1. Input can be in either zoned-address type A or B format. The output is always in zoned-address type B format.
2. The zone bits over the thousands and units positions of the output (bits 0 and 1) are adjusted according to the status of the 7080 mode, the 705 I/II mode, and the 40K and 80K switch triggers.
3. Incrementing by five or decrementing by five always produces an address ending in 4 or 9 which points to the next higher or next lower block of five E80 characters from the input address.

#### *Address Converter*

The address converter changes a zoned address to a binary-coded 24-bit System/370 address. The output of the converter is added to the E80 memory relocation value (in FPR 2) to arrive at the actual System/370 address. Internally, the address converter consists of three parts:

*The Units-Position Decoder*, which translates the digit portion of the units position of the zoned address.

*The Decimal-to-Binary Decoder*, which uses an algorithm to translate the digit portion of the tens, hundreds, and thousands positions of the zoned address.

*The Wrap Decoder*, which converts the zone bits (bit positions 0 and 1) of the units and thousands position of the zoned address and adjusts the bits according to the status of the 7080 mode, 705 I/II mode, and the 40K and 80K switch triggers.

#### *Select Encoder*

The select encoder is used in 7080-address to System/370-address conversion for pointing to a select-table byte used in I/O operations.

#### *E80 Digit Operating Hardware*

Associated with the serial adder path of Models 165, 165 II, and 168 are devices which operate on E80 data, one byte at a time. These devices include termination controls, hardware status registers, a data translator, multiply-shift controls, and zero and decimal corrects.

The *termination controls* detect data termination conditions as the 7080 formatted data is processed through the serial adder.

The *hardware status registers* maintain the status and control bits (shown in the following chart) necessary for the 7080 microprogram. (If the system leaves emulator mode, the program status is placed in FPR 0 bytes 5-7.)

| *Group* | *Bit* | *Significance or Use* |
|---|---|---|
| 1 | 0 | Complement WIB |
| | 1 | Hardware status (stat x) |
| | 2 | Digit not zero trigger (DNZT) |
| | 3 | Do index 4 |
| | 4 | Do index 2 (A term) |
| | 5 | Do index 1 (B term) |
| | 6 | Zone carry trigger |
| | 7 | Digit carry trigger |
| 2 | 0 | Increment 2 factor |
| | 1 | Increment 1 factor |
| | 2 | Miscellaneous control status |
| | 3 | Instruction interrupted |
| | 4 | Check 2 |
| | 5 | Check 1 |

| Group | Bit | Significance or Use |
|---|---|---|
| | 6 | ASU ≠ 0 |
| | 7 | Miscellaneous control status |
| 3 | 0 | DIL interrupt flag |
| | 1 | DIL count flag |
| | 2 | 7080 mode |
| | 3 | 705 I/II mode |
| | 4 | 40K switch |
| | 5 | 80K switch |
| | 6 | 905 check |
| | 7 | 904 check |

The *data translator* is a two-way eight-bit code translator. Under the appropriate microorder control, the translator changes EBCDIC bytes to E80 characters, or vice versa.

The *multiply-shift controls* are used for a three-bit shift control in an E80 multiply operation.

A *zero correct* is provided for characters put into either side of the serial adder. A *decimal correct* is provided on the serial adder output, along with digit and zone carry triggers.

### *Microprogram Control*

The 7080 instructions are executed under microprogram control. The information for microprogram control requires 1,024 words of control storage (CS), with the words extended from 108 bits per word to 126 bits per word. Control storage includes both read-only storage (ROS) and writable control storage (WCS). To provide for the required amount of control storage, the WCS of the Model 165 is increased by 1,024 words, and both the ROS and WCS of Models 165 II and 168 are increased by 512 words.

Bit 125 is used to provide odd parity for the additional bits. The additional fields and the corresponding bits are shown in the following chart:

| *Function* | *Bit Position* |
|---|---|
| Emulator control field 1 | 108-111 |
| Emulator control field 2 | 112-113 |
| Emulator control field 3 | 114-116 |
| Emulator control field 4 | 117-118 |
| Emulator control field 5 | 119 |
| Emulator control field 6 | 120-121 |
| Emulator emit control field | 122 |
| Emulator control field 7 | 123 |
| Emulator control field 8 | 124 |
| Emulator parity for bits 108-124 | 125 |

### *7080 Emulator Error Detection*

The 7080 emulator provides functions for additional error checks within the Model 165, 165 II, and 168 data path, and also for the extended writable control storage. Areas affected are described in the following text.

*Serial Adder.* Since the 7080 is a BCD machine, the data contained in 7080 memory is BCD-formatted and bits 2 and 3 of any E80 byte are not used. A parity adjustment must be made for these bits, since they are prevented from going through the serial adder. The parity adjustment prevents false serial-adder halfsum checks.

The serial adder latches perform another check for the 7080 emulator. Data gated through the 7080 translator is parity-checked on both input and output. This is done because the translator is a bypass for the serial adder, and its output must be gated back into the serial adder's data flow.

*Parallel Adder.* With the additional hardware in the parallel adder data path, the emulator cannot gate a byte-by-byte parity bit. Therefore, the emulator generates a sum bit for the data going into the parallel adder.

*Control Storage.* The emulator parity-checks the 18 bits which it adds for its CS words.

### *Maintenance Control Word (MCW)*

The bits provided in the Model 165, 165 II, and 168 MCW for emulator use are:

| *Bit* | *Function* |
|---|---|
| 0 | Reverses the parity of extended control storage. |
| 1 | Forces an emulator translator check (this appears as a parity check when the serial adder latches are set). |

### *Machine Error Retry*

For machine-check errors, each 7080 emulator instruction is retryable up to the point where pertinent input data (for a restart of that instruction) is destroyed.

#### Instruction Interruption

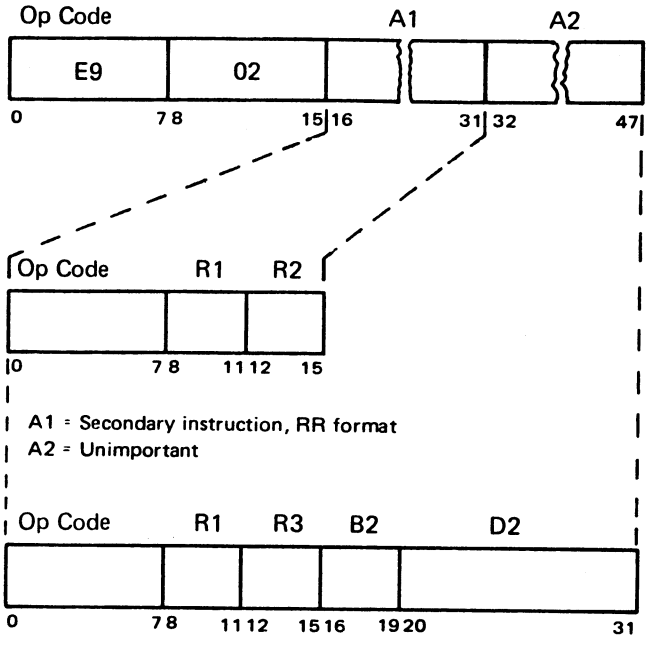
Many of the instructions are interruptible prior to completion. (See individual instruction descriptions to determine which instructions are interruptible.) When an instruction is to be interrupted, the microprogram stores the necessary intermediate results and status in general registers and floating-point registers. Then the System/370 IC is decremented by 6 and the instruction ends operation. The emulator status valid trigger is turned off during the interruption routine, and additional status is stored in FPR 1. To resume the instruction, it is necessary to restore all general registers and floating-point registers to the exact values set up during the interruption process, and reexecute the feature instruction. The emulator status valid trigger must not be on when the instruction is reentered.

This section contains the formats and descriptions of the primary and secondary instructions of the emulator instruction set. (See Appendix C for an instruction summary.)

## PRIMARY EMULATOR INSTRUCTION

### EMU (Emulator Feature Instruction)

*SS Format*




A1 + A2 = Secondary instruction, RS format

System/370 Models 165, 165 II, and 168 emulate 7080 instructions via the emulator instruction EMU. This System/370 instruction, the primary instruction of this feature, has a special SS format. The first byte contains the op code, the second byte contains the emulator flag, and the next four bytes contain any of the RR- or RS-format secondary instructions. The first two bytes are, in effect, a prefix for each of the secondary instructions.

Any defined secondary (general or special) emulator instruction causes the EMU instruction to terminate and cause a specification exception.

Before executing the secondary instruction, the EMU instruction examines the emulator status valid trigger. If the trigger is on, EMU proceeds to the secondary instruction. If it is off, EMU loads the 7080 hardware status (from FPR 0 bytes 5-7), turns on the emulator status valid trigger, and does the entire instruction a second time.

*Indicator:*

Emulator status valid trigger

*Program Interruptions:*

Access exception
Operation exception
Specification exception

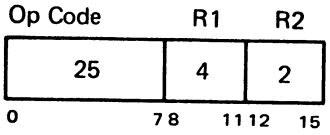
## GENERAL EMULATOR INSTRUCTIONS

The general emulator operations are roughly grouped as follows:

1. The DIL and DILEX.
2. Mode manipulation—SETMD, MDOFF, MDON.
3. Branch—BRIND, BRIF, and BSOF.
4. Zoned-address manipulation—ZACB, ZAPZ, ZAPON, ZAPF, ZAMF, and CERZ.
5. E80 storage address manipulation—ATFS, CINZA, RSTER, and UNSPC.
6. I/O manipulation—CSELR, UBP, UBS, LB, and LBM.
7. Character recognition—SALSM.

### DIL (Do Interpretive Loop)

*RR Format*



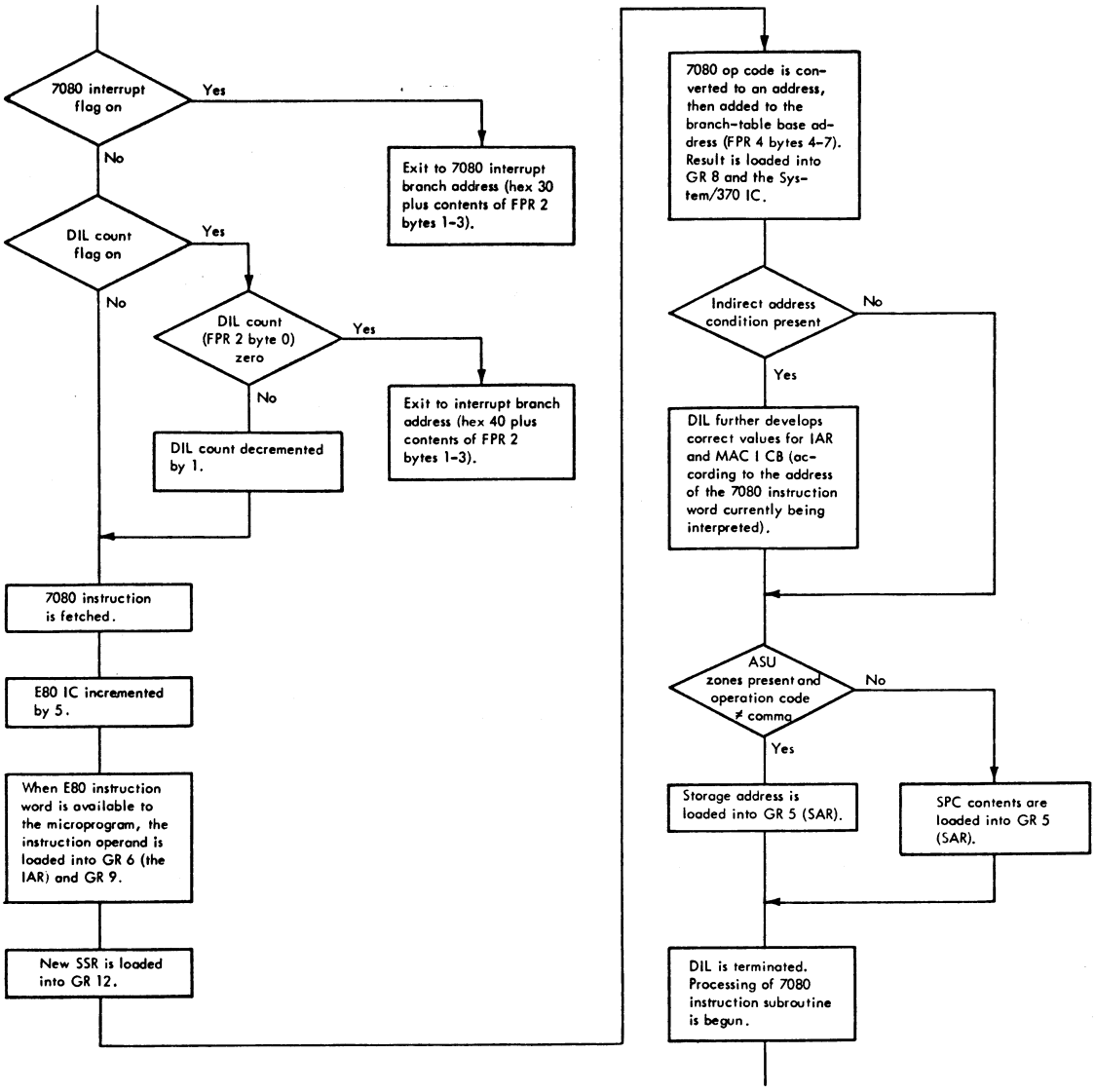

*Function*

The DIL instruction handles the 7080 I-time and IA-time functions. It also buffers the initial 7080 IC and DIL count for machine errors during DIL execution, and has provision for counting DIL's and handling 7080 interruptions.

This instruction is retryable up to the third machine cycle in a Model 165, but not retryable in a Model 165 II or 168.

Access exceptions are possible on System/370 storage references to 7080 memory.

Figure 1 shows the DIL instruction operation.




Note: This chart is a schematic of the DIL instruction. It should not be construed as being a program subroutine.

Figure 1 DIL Instruction–Internal Logic

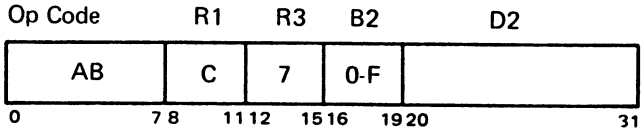
### *Registers Affected*

| *GR* | *Contents* |
|---|---|
| 5* | Current SAR (replaces old SAR). |
| 6* | Initial address register (IAR) is filled from the operand field of the next 7080 instruction. (ASU bits are meaningless here if the IAR was set up as a result of indirect addressing.) |
| 8 | Address of first instruction of DIL subroutine for the 7080 instruction. |
| 9 | New MAC I coded binary. |
| 12 | New storage select register (SSR) in byte 3 bits 0–3. |
| *FPR* | |
| 2 | If in count mode, byte 0 contains the updated DIL count. |
| 4 | IC incremented by 5. |

* Model 165 II or 168. During an access exception, GR 5 contains the address of the converted 7080 IC. and GR 6 contains the System/370 address of the DIL instruction incremented by 6.

## DILEX (DIL Extension)

### *RS Format*




D2 contains a variable whose value depends on programming requirements.

B2 + D2 must begin on a word boundary.

### *Function*

This instruction is used for all the DIL subroutine areas where the SSR must be interrogated to determine what

operation must be performed. (Example: Branching on comma op codes must be modified by the ASU zoning to point to the proper instruction, such as SPC, LFC, and UFC).

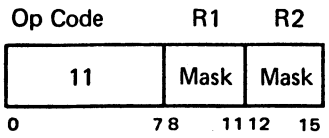
*Description*

DILEX fetches a word from C(B2)+D2, adds it to GR 12 byte 3, and stores the result in GR 8 and the System/370 IC.

This instruction is retryable up to the last machine cycle.

**SETMD (Set Mode)**

*RR Format*




*Function*

SETMD sets or resets hardware mode control triggers according to the R1 and R2 mask fields.

*Description*

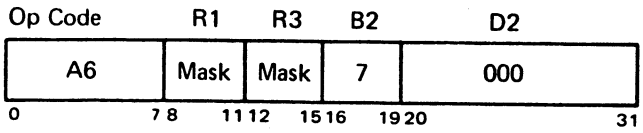
A bit in the one state sets the trigger it controls, and a bit in the zero state resets the trigger it controls. Bits 8-15 are assigned as follows:

| *Mode Bit* | *Trigger* |
|---|---|
| 8 | 7080 interrupt flag |
| 9 | DIL counter flag |
| 10 | 705 I/II mode |
| 11 | 7080 mode |
| 12 | 40K switch |
| 13 | 80K memory |
| 14 | 905 check switch (On=automatic) |
| 15 | 904 check switch (On=automatic) |

This instruction is completely retryable.

**MDON (Mode On)**

*RS Format*




*Function*

MDON sets the hardware mode control triggers according to the mode bits in the R1 and R3 mask fields.

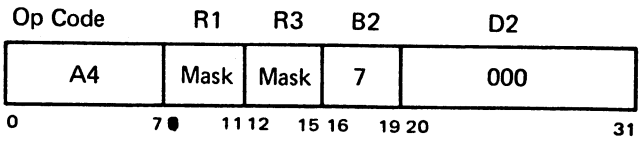
*Description*

A bit in the one state causes the hardware mode control trigger to be set. A bit in the zero state does not change the state of that trigger. (See the SETMD instruction for the bit assignments.)

This instruction is retryable up to the last machine cycle in a Model 165, but not retryable in a Model 165 II or 168. Also, all the general registers remain unchanged.

**MDOFF (Mode Off)**

*RS Format*




*Function*

MDOFF resets the hardware mode control triggers according to the mode bits in the R1 and R3 mask fields.

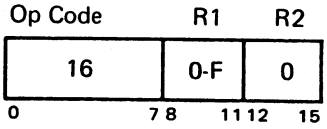
*Description*

A bit in the zero state causes the hardware mode control trigger to be reset. A bit in the one state does not change the state of the respective mode bit. (See the SETMD instruction for the bit assignments.)

This instruction is retryable up to the last machine cycle in a Model 165, but not retryable in a Model 165 II or 168. Also, all the general registers remain unchanged.

**BSOF (Branch on 705 I/II Mode)**

*RR Format*




*Function*

BSOF interrogates the 705 I/II mode trigger and effects a System/370 branch if the trigger is on.

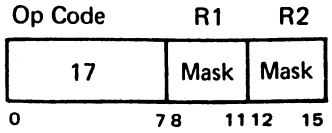
*Description*

If the 705 I/II mode trigger is on and the 7080 mode trigger is off, a System/370 branch is forced to the location indicated by the R1 general register. If the trigger is off, the next sequential System/370 instruction is executed.

This instruction is completely retryable.

### BRIND (Branch on Indicator)

*RR Format*




*Function*

BRIND emulates several of the 7080 conditional transfer instructions.

*Description*

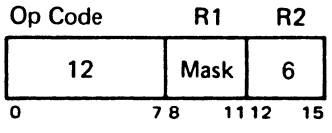
BRIND tests the E80 dynamic status indicators in FPR 0 to emulate the 7080 transfer instructions TRS, TRH, and TRE. If the particular bit being tested is on, the branch is made by moving the 7080 IAR to the 7080 IC, converting the IAR contents to zoned-address type B in the process.

This instruction is retryable up to the last two machine cycles in a Model 165, and up to the fourth machine cycle in a Model 165 II or 168.

| *Bits* | *Function* |
|---|---|
| 8 | If on, the FPR 0 bits tested by the mask are reset. |
| 9 | Mask bit for bit 1 of selected byte in FPR 0. |
| 10 | Mask bit for bit 5 of selected byte in FPR 0. |
| 11 | Mask bit for bit 6 of selected byte in FPR 0. |
| 12 | Mask bit for bit 7 of selected byte in FPR 0. |
| 13 | If = 0 (off) and if tested bits ≠ 0, branch is taken.<br>If = 1 (on) and if tested bits = 0, branch is taken. |
| 14, 15 | Selects the indicator byte in FPR 0 to be tested. |

### BRIF (Branch If)

*RR Format*




*Function*

BRIF effects a conditional 7080 transfer based on the System/370 condition code.

*Description*

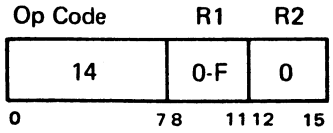
The condition code specified by the mask is tested. If the test is successful, a branch is made by moving the 7080 IAR contents to the 7080 IC, converting the IAR contents to zoned-address type B in the process.

| *Condition Code* | *Instruction Bit* |
|---|---|
| 0 | 8 |
| 1 | 9 |
| 2 | 10 |
| 3 | 11 |

This instruction is completely retryable.

### ZACB (Zoned Address to Coded Binary)

*RR Format*




*Function*

ZACB converts the contents of the R1 general register from zoned-address format to its coded binary equivalent, and returns the results to the same register.

*Description*

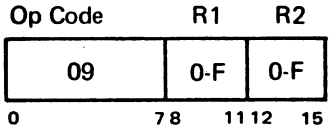
In conversion, each zoned decimal address yields a unique (but not necessarily sequential) binary address.

The address conversion takes into consideration the mode of operation, the memory size of the emulated system, and the relocation address in FPR 2 bytes 4-7.

This instruction is retryable up to the last machine cycle in a Model 165, and up to the second machine cycle in a Model 165 II or 168.

### ZAPZ (Zoned Address Plus Zero)

*RR Format*




*Function*

ZAPZ removes the ASU zones from a zoned address in the R2 general register, forces it to conform to proper memory bounds by wraparound, and places the resulting zoned address in the R1 general register.

*Description*

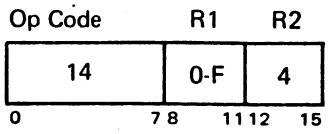
The contents of the R2 general register are changed from zoned-address type A or B to zoned-address type B (with ASU zones removed) and are placed in the R1 register. In addition, bits 25, 24, and 0 (corresponding to the 7080 80K, 40K, and 20K address bits) are altered under the control of the mode triggers in the following manner:

1. Bit 25 (the 80K bit) is blocked if the 7080 mode trigger is off.
2. Bit 24 (the 40K bit) is blocked if the 7080 mode trigger is off and the 705 I/II mode trigger or 40K switch trigger is on.
3. Bit 0 (the 20K bit) is blocked if the 705 I/II mode trigger is on and the 40K switch trigger and 7080 mode trigger are both off.

This instruction is retryable up to the last machine cycle in a Model 165, and up to the second machine cycle in a Model 165 II or 168.

### ZAPON (Zoned Address Plus One)

*RR Format*




*Function*

The zoned address in R1 is incremented by one.

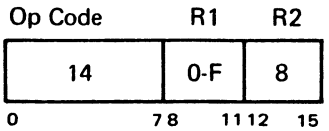
*Description*

The contents of the R1 general register (in zoned-address type A or B format) are incremented by one in the manner of a 7080 AAM (Add Address to Memory) instruction and then returned to the R1 register (in zoned-address type B format) with ASU zones stripped and bits 25, 24, and 0 altered in the manner described in the ZAPZ instruction.

This instruction is retryable up to the last machine cycle in a Model 165, and up to the second machine cycle in a Model 165 II or 168.

### ZAPF (Zoned Address Plus Five)

*RR Format*




*Function*

The zoned address in R1 is incremented to the next higher 4 or 9 location.

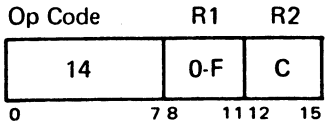
*Description*

The contents of the R1 register (in zoned-address type A or B format) are incremented to the 4 or 9 location of the next higher 7080 word. (For example, 0004 goes to 0009 or 0007 goes to 0014.) The carryout of the digit portion of the zoned address is added to the zones (bits 25, 24, 0 and 1) in the manner of a 7080 AAM instruction. The result is returned to the same general register (in zoned-address type B format) with ASU zones removed and bits 25, 24 and 0 altered in the manner described in the ZAPZ instruction.

This instruction is retryable up to the last machine cycle in a Model 165, and up to the second machine cycle in a Model 165 II or 168.

### ZAMF (Zoned Address Minus Five)

*RR Format*




*Function*

The zoned address in the R1 general register is decremented to the next lower 4 or 9 location.

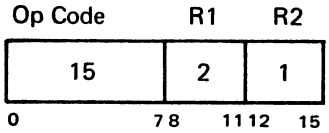
*Description*

ZAMF is essentially identical to ZAPF, except that the address in R1, instead of going to the next higher 4 or 9 address, goes to the next lower 4 or 9 address. (For example, 0009 goes to 0004, and 0013 goes to 0009.) If the content of the R1 general register is less than 4, the result is the corrected output after the mode-trigger blocking of bits 25, 24, and 0 is applied to the decimal value of 159,999.

This instruction is completely retryable in a Model 165, and is retryable up to the second machine cycle in a Model 165 II or 168.

### CSELR (Convert Select Register)

*RR Format*



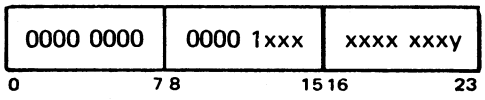

*Function*

CSELR is used to convert a 7080 select address to a System/370 address.

*Description*

A 24-bit System/370 address is obtained by (1) performing an eight-bit left shift on the R1 select register (GR 2), (2) inserting bits 0-7 of GR 2 into bits 24-31, (3) converting the result by way of the select encoder, and (4) shifting the result right three bit positions. The address then has the following structure:

In the resulting address:




1. Bits 0-11 are always zeros.
2. Bit 12 is always one.
3. Bits 13-22 (x's) are the output of the zoned-address to coded-binary converter.
4. Bit 23 (y) is set to one if bit 6 of the original content of GR 2 is one; otherwise it is set to zero.

The relocation value in FPR 4 is added to the 24-bit System/370 address.

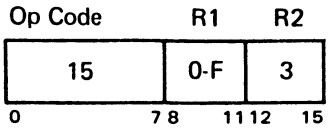
The address in GR 3 is incremented by the contents of select-table byte 3, which is fetched via the System/370 address calculated by CSELR.

This instruction is completely retryable in a Model 165, and is retryable up to the third machine cycle in a Model 165 II or 168.

An access exception can occur on the fetch of a select-table byte.

### CINZA (Convert Zoned Address)

*RR Format*




*Function*

CINZA facilitates E80 interruption handling by providing the capability for rapid storing of register contents in E80 storage. Binary zeros are converted to BCD format and the byte sequence is reversed before storing.

*Description*

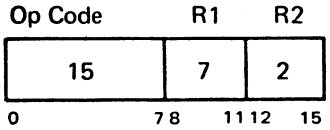
The contents of the R1 general register are converted from zoned-address type B to type A, with the byte sequence reversed. (Thus the contents of byte 0 go into byte 3, the contents of byte 1 go into byte 2, etc.) The doubleword (with the converted contents of the R1 general register in bytes 0-3 and the E80 zeros in bytes 4-7) is then stored in System/370 main storage at the address contained in GR 3. The address in GR 3, which must point to the doubleword boundary, is incremented by eight.

This instruction is retryable up to the last machine cycle in a Model 165, and up to the third machine cycle in a Model 165 II or 168.

An access exception can occur on a storage reference to the address in GR 3.

### UNSPC (Convert and Store SPC)

*RR Format*




*Function*

UNSPC converts the E80 starting point counter (SPC) to a four-byte constant which corresponds to the four characters stored in CASU 15 word 1 by the 7080 when an automatic interruption occurs.

*Description*

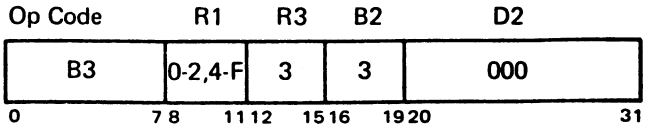
The UNSPC result is achieved by (1) subtracting the relocation value (contained in FPR 4 bytes 4-7) from the System/370 address in GR 7 and (2) sending this value through the UNSPC decoder (described earlier). The byte sequence is then reversed and this function, together with four E80 BCD zeros in bytes 4-7, is stored in the doubleword location given by GR 3. GR 3 is then incremented by eight.

This instruction is completely retryable in a Model 165, and is retryable up to the third machine cycle in a Model 165 II or 168.

An access exception can occur on a storage reference to the address in GR 3.

### RSTER (Restore E80 Register)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

RSTER facilitates emulation of the 7080 LIP (Leave Interrupt Program) instruction by providing the capability for restoring E80 register contents from central storage (CASU 15). RSTER performs essentially the reverse of the CINZA special instruction.

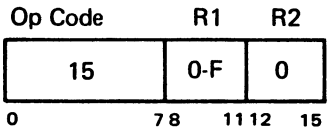
*Description*

The System/370 word specified by the address in GR 3 is fetched and changed from zoned-address type A to type B, and the byte sequence is reversed. The result is placed in the R1 general register. In addition, the address in GR 3 is incremented by eight.

This instruction is retryable up to the last machine cycle.

### ATFS (Modulo 256 Addition)

*RR Format*




*Function*

ATFS provides 256-byte wraparound capability for emulation of E80 storage functions.

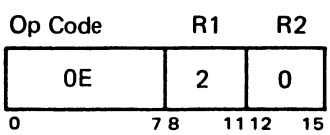
*Description*

GR 3 byte 3 is added to byte 3 of the R1 general register. Any carryout of the low-order byte of R1 is ignored. The result replaces byte 3 of the R1 register.

This instruction is retryable up to the last machine cycle in a Model 165, and up to the third machine cycle in a Model 165 II or 168.

### LB (Load Buffer)

*RR Format*



*Function*

LB moves characters in E80 memory serially to the System/370 I/O buffer, and converts them from E80 BCD format to EBCDIC format.

*Description*

This operation depends on GR's 2, 0, and 1 to determine, respectively, the number of bytes to be moved, the E80 address of the first character to be moved, and the System/370 address of the I/O buffer. It terminates when an E80 group mark is sensed or when the count in GR 2 is reduced to zero. If a no-bit byte (hex 00) is sensed, an abnormal termination is forced. Such termination forces a System/370 branch to the address obtained from FPR 2 plus hex 60, and the contents of GR's 0, 1, and 2 are the same as for the normal termination by sensing a group mark. (The microprogram samples for a pending interruption at the completion of every fifth doubleword going to the System/370 I/O buffer.) The contents of GR's 0, 1, and 2 are as follows:

| *GR* | *Contents at Start* | *Contents at Termination* |
|---|---|---|
| 0 | Zoned-address type A or B pointing to the first E80 memory character to be moved. | The zoned address of the next character in E80 memory if terminated by buffer count equal to zero. Otherwise, the zoned address points to a 0/5 E80 memory location which is ten higher than that of the block containing the group mark. |
| 1 | System/370 I/O buffer address. | Unpredictable. |
| 2 | Binary count equal to the buffer size. | Zero if no group mark is sensed; otherwise the residue count is equal to the number of unfilled buffer positions. |

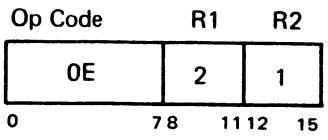
The contents of GR3, a work register, are unpredictable at instruction termination.

An access exception can occur on a reference to a 7080 memory or buffer address.

This instruction is retryable in a Model 165 until GR's 0, 1, and 2 are changed in the ending routine, and is retryable up to the seventh machine cycle in a Model 165 II or 168.

**LBM (Load Buffer to Memory Boundary)**

*RR Format*




*Function*

LBM works identically to the LB instruction, except that LBM terminates when the buffer count is reduced to zero or when the end of 20K E80 memory is sensed.

*Description*

LBM samples every fifth doubleword going into the System/370 I/O buffer. Sensing a no-bit byte in LBM results in the same abnormal termination as sensing a no-bit byte in LB. The contents of GR's 0, 1, and 2 are as follows:

| *GR* | *Contents at Start* | *Contents at Termination* |
|---|---|---|
| 0 | Zoned-address type A or B pointing to the first E80 memory character to be moved. | The zoned address of the next character in E80 memory, if terminated by buffer count equal to zero; otherwise the zoned address is one higher than the 20K E80 memory boundary. |
| 1 | System/370 I/O buffer address. | Unpredictable. |
| 2 | Binary count equal to the buffer size. | Zero if no end of 20K E80 memory is sensed; Minus one if end of 20K E80 memory is sensed at the same time count reaches zero. A residue count specifying the number of unfilled buffer positions when the operation is terminated. |

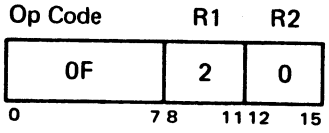
The contents of GR3, a work register, are unpredictable at instruction termination.

An access exception can occur on a reference to a 7080 memory or buffer address.

This instruction is retryable in a Model 165 until GR's 0, 1, and 2 are changed in the ending routine, and is retryable up to the seventh machine cycle in a Model 165 II or 168.

**UBS (Unload Buffer Serial)**

*RR Format*




*Function*

UBS moves characters serially from a System/370 I/O buffer into E80 memory and translates these characters from EBCDIC format to E80 BCD format at the same time. UBS is designed for the emulation of 7080 serial read operations.

*Description*

Data is moved from the System/370 I/O buffer to E80 memory and converted from EBCDIC format to E80 BCD format. This operation is defined by the contents of GR's 0, 1, and 2, specifying, respectively, the E80 address in zoned-address format, the System/370 I/O buffer address, and the binary count of the number of bytes to be moved. During the operation, if end-of-memory is encountered (as determined by the current settings of the 7080 mode trigger, the 705 I/II mode trigger, and the 40K switch trigger), the zoned address wraps, which forces data to be put into E80 memory position 0000. Sensing the count of zero terminates the operation. After every fifth doubleword is unloaded, translated, and the corresponding 7080 word (the eighth) is written into 7080 memory, the microprogram samples for a pending interruption.

The contents of the GR's involved are as follows:

| *GR* | *Contents at Start* | *Contents at Termination* |
|---|---|---|
| 0 | Zoned-address type A or B pointing to the first position of the E80 memory receiving area. | Zoned address one higher than the last receiving zoned-address. |
| 1 | System/370 I/O buffer address. | Last doubleword address referenced. |
| 2 | Binary count equal to the number of bytes to be moved. | Zero. |

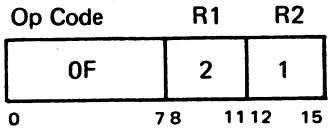
The contents of GR 3, a work register, are unpredictable at instruction termination.

An access exception can occur on a reference to a 7080 memory or buffer address.

This instruction is retryable in a Model 165 until GR's 0, 1, and 2 are changed in the ending routine, and is retryable up to the second machine cycle in a Model 165 II or 168.

### UBP (Unload Buffer Parallel)

*RR Format*




*Function*

UBP moves characters five at a time from a System/370 I/O buffer into E80 memory and translates these characters from EBCDIC format to E80 BCD format. UBP aids emulation of 7080 communication channel read operations.

*Description*

UBP performs essentially the same functions as the UBS instruction, including sampling for interruptions, but UBP has a different set of terminating conditions. It terminates either on zero count (GR 5) or on E80 end-of-memory. If the last character transmitted into E80 memory has no 4/9 address, the remaining character positions of that E80 word are filled with E80 group marks. E80 end-of-memory is caused by the following conditions:

1. The zoned address steps past 19,999 if the 7080 mode trigger and 40K switch trigger are off, and the 705 I/II trigger is on.
2. The zoned address steps past 39,999 if the 7080 mode trigger is off and the 705 I/II mode trigger or 40K switch trigger is on.
3. The zoned address steps past 79,999 if the 7080 mode trigger is off or if the 80K switch trigger is on.
4. The zoned address steps past 159,999.

The contents of the GR's are as follows:

| *GR* | *Contents at Start* | *Contents at Termination* |
|---|---|---|
| 0 | Zoned-address type A or B pointing to the first position of the E80 memory receiving area. | The zoned address pointing to a 0/5 address of the next block of five characters, if the count is zero; otherwise it is 0000. |
| 1 | System/370 I/O buffer address. | Last doubleword address referenced. |
| 2 | Binary count equal to the number of bytes to be moved. | Zero or a residual count which equals the number of characters remaining in the buffer when E80 end-of-memory is sensed. |

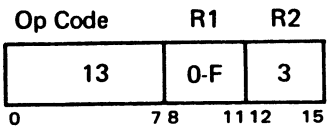
The contents of GR 3, a work register, are unpredictable at instruction termination.

An access exception can occur on a reference to a 7080 memory or buffer address.

This instruction is retryable in a Model 165 until GR's 0, 1, and 2 are changed in the ending routine, and is retryable up to the second machine cycle in a Model 165 II or 168.

### CERZ (Convert Eighty Register Zeros)

*RR Format*




*Function*

The CERZ instruction converts binary zeros in a general register to BCD zeros.

*Description*

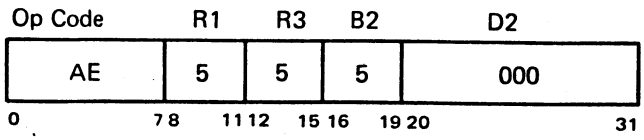
The zoned address from the R1 general register is converted into zoned-address type A and stored back into the general register. For example, those characters with 0000 (binary zero) values for the numeric bits are changed to 1010 (BCD zero). No adjustment is made to the zones of the thousands and units positions of the zoned address. The digit portion of each zoned-address byte having value greater than

hex nine yields results subject to the rules of the E80 adder under decimal-correct operation.

This instruction is retryable up to the last machine cycle in a Model 165, but not retryable in a Model 165 II or 168.

### SALSM (Search and Locate Storage Mark)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

SALSM searches E80 storage addressed by SAR (GR 5) and determines the location of the limiting storage mark.

*Description*

The address of the storage mark is placed in GR 4. The absence of a limiting storage mark in the addressed storage bank causes the instruction to exit to the address hex 80 greater than the one in FPR 2.

This instruction is completely retryable.

## SPECIAL EMULATOR INSTRUCTIONS

To achieve the speed required, a number of 7080 operations must be emulated entirely by microprogramming and hardware. They are grouped as follows:

*Memory to Storage:* EADD, ECMP, ELDA, ELFC, ELOD, ELSB, EMPY, ERAD, ERSU, ESUB, ETLH, and ETLE.

*Storage to Memory:* EAAM, EADM, ESPR, EST, EUFC, EUTA, EUNL, and EUSB.

*Memory to Memory:* EBLM, EBLMS, ESND, ETMT, ETMTS, ETCT, and ETCR.

*Storage Field Manipulation:* ESET, ESHR, ESPC, and ERND.

*Indirect Address Instruction:* EEIA (which enables indirect addressing in 7080 mode).

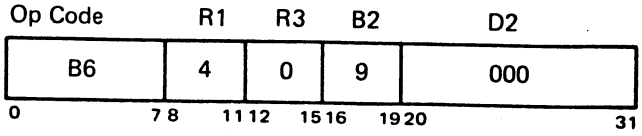
*Transfer:* ETRP, ETRZ, and ETR.

These instructions work identically to their 7080 counterparts, with the exception of some peculiarities of the 7080 that are not duplicated. Where such cases exist, a description of the dissimilarity is given.

Arithmetic operations involving special characters do not necessarily yield the same inconsistent results that the 7080 yields.

### EADD (Emulated Add)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

EADD emulates the 7080 ADD instruction.

*Description*

The numeric field whose units position in E80 memory is addressed by MAC I CB (memory address counter I coded binary) (GR 9) is added to the E80 storage field addressed by SAR (GR 5). EADD follows the rules of the 7080 ADD instruction, with the following exceptions:

1. Inclusion of special characters in the arithmetic fields may not yield the same inconsistent results as in the 7080.
2. A 904 check condition does not terminate the add operation, which continues as if the 904 switch were not on. This allows emulator program handling of overflow conditions.
3. If a 904 or 905 check condition is detected and the corresponding check switch trigger is on, the appropriate check bit in FPR 0 is turned on and a forced System/370 branch is taken to the address hex 10 greater than the one in FPR 2.
4. In a Model 165, the microprogram samples for a pending interruption after every fifth doubleword of E80 storage used.
5. If no terminating storage mark is found, the microprogram exits to the address hex 80 greater than the one in FPR 2.

An access exception can occur on a reference to 7080 memory.

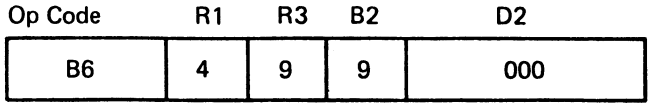
This instruction is retryable until the first word of the sum is stored, which could be the last machine cycle if the sum is contained in one doubleword.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-8 | - | Unchanged. |
| 9 | | (Model 165) Unchanged. |
| 9 | MAC I CB | (Model 165 II or 168) Original contents restored after GR 9 used as a work register. |
| 10-15 | - | Unchanged. |
| *FPR* | | |
| 6 | Work register | Unpredictable. |

### ECMP (Emulated Compare)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ECMP emulates the 7080 CMP instruction.

*Description*

The E80 storage field addressed by the SAR (GR 5) is compared to the E80 memory field addressed by the MAC I CB (GR 9), following the rules of the 7080 CMP instruction, with two exceptions:

1. The microprogram samples for interruptions after the comparison of every fifth doubleword of E80 storage.
2. If no terminating storage mark is found, the microprogram exits to the address hex 80 greater than the one in FPR 2.

An access exception can occur on a reference to 7080 memory.

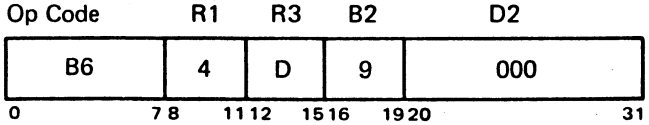
This instruction is completely retryable.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0, 2, 3 | Work registers | Unpredictable. |
| 4 | MAC I | Zoned-address type B of the last block compared, minus 5. |
| 5 | SAR | Address of last doubleword of E80 storage to be used, plus 8. |
| 6 | IAR | Current E80 operand. |
| 7 | SPC | Unchanged. |
| 9 | MAC I CB | (Model 165) Current E80 operand. |
| 9 | MAC I CB | (Model 165 II or 168) Original contents restored after use as a work register. |
| 11 | MAC II | Unchanged. |
| *FPR* | | |
| 0 | Status | Compare status (in byte 3). |
| 6 | Work register | Unpredictable. |

### ELDA (Emulated Load Address)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ELDA emulates the 7080 LDA instruction.

*Description*

The four-character instruction address in E80 memory at the MAC I CB (GR 9) is placed in E80 storage at the address in the SAR (GR 5) as a five-digit numeric field in 705 III mode or as a six-digit numeric field in 7080 mode.

ELDA follows the rules for the 7080 LDA instruction, with one exception: If the E80 address does not end in 4/9, neither the 900 check bit nor the ANY check bit is turned on, but a System/370 branch is forced to the address hex 20 greater than the one in FPR 2.

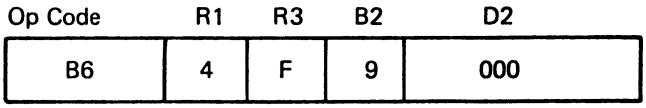
This instruction is completely retryable in a Model 165, and is retryable up to the thirteenth machine cycle in a Model 165 II or 168.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-15 | – | Unchanged. |

### ELFC (Emulated Load Four Characters)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ELFC emulates the 7080 LFC instruction.

*Description*

Four characters at the MAC I CB (GR 9) in E80 memory are moved to E80 storage at the address of the SAR (GR 5).

ELFC follows the rules of the 7080 LFC instruction, with the following exception: If ELFC is given in 705 I/II or 705 III mode, the 900 check bit (in FPR 0) is not turned on, but a System/370 branch is forced to the address hex 20 greater than the one in FPR 2.

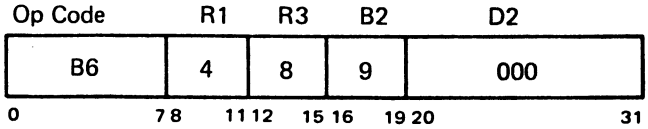
This instruction is completely retryable in a Model 165, and is retryable in a Model 165 II or 168 up to the point of storing the first result field.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-8 | – | Unchanged. |
| 9 | – | (Model 165) Unchanged. |
| 9 | Work register | (Model 165 II or 168) Unpredictable. |
| 10-15 | – | Unchanged. |

### ELOD (Emulated Load)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ELOD emulates the 7080 LOD instruction.

*Description*

The field in E80 memory at the address in the MAC I CB (GR 9) is loaded into E80 storage at the address in the SAR (GR 5).

ELOD follows the rules of the 7080 instruction, with two exceptions:

1. The microprogram samples for a pending interruption after every fifth doubleword loaded into E80 storage.
2. If no terminating storage mark is found, the microprogram exits to the address hex 80 greater than the one in FPR 2.

An access exception can occur on a reference to 7080 memory.

This instruction is retryable in a Model 165 up to the ending routine, when status is changed,and is retryable in a Model 165 II or 168 up to the point of storing the first result field.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-8 | – | Unchanged. |
| 9 | – | (Model 165) Unchanged. |
| 9 | Work register | (Model 165 II or 168) Unpredictable. |
| 10-15 | – | Unchanged. |

**ELSB (Emulated Load Storage Bank)**

*RS Format*

| Op Code | R1 | R3 | B2 | D2 |
|---|---|---|---|---|
| B6 | 4 | E | 9 | 000 |

0 7 8 11 12 15 16 19 20 31

(R3 contains the op-code modifier.)

*Function*

ELSB emulates the 7080 LSB instruction.

*Description*

The 256 characters in E80 memory, with the low-order character addressed by MAC I CB, are loaded with E80 storage starting at the address in SAR (GR 5).

ELSB follows the rules of the 7080 LSB instruction, with two exceptions:

1. If ELSB is given in non-7080 mode, a System/370 branch is forced to the address hex 20 greater than the one in FPR 2.
2. The 900 check bit in FPR 0 is not set on. This action identifies an E80 comma instruction given in non-7080 mode.

An access exception can occur on a reference to 7080 memory.

This instruction is completely retryable in a Model 165, and is retryable in a Model 165 II or 168 up to the point of storing the first result field.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-8 | – | Unchanged. |
| 9 | – | (Model 165) Unchanged. |
| 9 | Work register | (Model 165 II or 168) Unpredictable. |
| 10-15 | – | Unchanged. |

**EMPY (Emulated Multiply)**

*RS Format*

| Op Code | R1 | R3 | B2 | D2 |
|---|---|---|---|---|
| B6 | 0 | 0 | 9 | 000 |

0 7 8 11 12 15 16 19 20 31

(R3 contains the op-code modifier.)

*Function*

EMPY emulates the 7080 MPY instruction.

*Description*

The multiplicand at the address in the MAC I CB in E80 memory is multiplied by the value in E80 storage at the address in the SAR (GR 5). Multiplication is accomplished by addition of partial products previously stored in a table (Figure 2) in System/370 storage.

| Multiplier | Multiplicand 8 | 9 | 2 | 3 | 4 | 5 | 6 | 7 | Table Decrement (Hex) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 08 | 09 | 02 | 03 | 04 | 05 | 06 | 07 | -08 |
| 2 | 16 | 18 | 04 | 06 | 08 | 10 | 12 | 14 | -10 |
| 3 | 24 | 27 | 06 | 09 | 12 | 15 | 18 | 21 | -18 |
| 4 | 32 | 36 | 08 | 12 | 16 | 20 | 24 | 28 | -20 |
| 5 | 40 | 45 | 10 | 15 | 20 | 25 | 30 | 35 | -28 |
| 6 | 48 | 54 | 12 | 18 | 24 | 30 | 36 | 42 | -30 |
| 7 | 56 | 63 | 14 | 21 | 28 | 35 | 42 | 49 | -38 |
| 8 | 64 | 72 | 16 | 24 | 32 | 40 | 48 | 56 | -40 |
| 9 | 72 | 81 | 18 | 27 | 36 | 45 | 54 | 63 | -48 |
| A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | -50 |

Notes:

1. The 7080 multiplicand (addressed by MAC I CB) is loaded from E80 memory into E80 working storage bank 5 prior to the actual multiply operation in order to facilitate addressing the product table during multiply loops. The multiplier is addressed by the storage address register (SAR).
2. A = E80 BCD zero.
3. Multiplication by B, C, D, E, or F is equivalent to multiplication by 1, 2, 3, 4, or 5, respectively. (Arithmetic operations involving special characters do not necessarily yield the same inconsistent results as in the 7080.)
4. The table arguments are the hexadecimal representations of the numeric portion (bits 4-7) of the E80 characters; the functions are one-byte fields of the product in packed-decimal format.
5. The table starts at the location eight less than the one specified by FPR 4 bytes 4-7. The search argument is developed by decrementing the table address by the hex value corresponding to a particular multiplier digit and by fetching the doubleword at that location. The multiplicand digit is referenced and the correct product is selected.

Figure 2. Product Table

EMPY follows the rules of the 7080 MPY instruction, with the following exceptions:

1. If the multiplier is a storage mark, EMPY does not replace the storage mark with a zero. EMPY steps SPC plus hex 128 and places a zero and a storage mark at that location.
2. If the units position of the multiplier is a special character, the product obtained from the EMPY does not match the product obtained from the 7080 MPY in all cases.
3. If a 905 check condition is detected and the corresponding check-switch trigger is on, the appropriate check bit in FPR 0 is turned on and a System/370 branch is forced to the address hex 10 greater than the one in FPR 2.
4. The microprogram samples for an interruption at the completion of every multiplier character operation.
5. If EMPY goes into an unending loop, an exit is taken to the address hex 80 greater than the one in FPR 2.

An access exception can occur on a reference to 7080 memory or to a storage bank 5 address.

This instruction is retryable until the first word of the multiplicand is stored in storage bank 5.

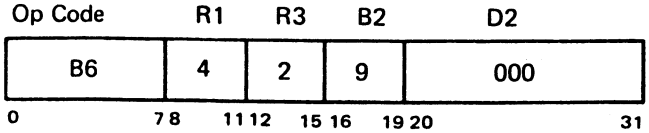
E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6 | IAR | Unchanged. |
| 7 | SPC | Pointer to the units position of the product (the initial SPC plus hex 128). |
| 9 | MAC I CB | (Model 165) Unchanged. |
| 9 | MAC I CB | (Model 165 II or 168) Original contents restored after use as a work register. |
| 11 | MAC II | Unchanged. |
| *FPR* | | |
| 6 | Work register | Unpredictable. |

If EMPY is given with ASU ≠ 0 or if the product is longer than 128 bytes, the instruction is not valid, and a branch is taken to the address hex 70 greater than the one in FPR 2 bytes 1-3.

### ERAD (Emulated Reset and Add)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ERAD emulates the 7080 RAD instruction.

*Description*

The numeric field in E80 memory at the address in the MAC I CB (GR 9) is entered into E80 storage starting at the address in the SAR (GR 5).

ERAD follows the rules of the 7080 RAD instruction, with the following exceptions:

1. If a 905 condition is detected and the 905 check-switch trigger is on, a forced System/370 branch is taken to the address hex 10 greater than the one in FPR 2.
2. The microprogram samples for a pending interruption at the completion of every fifth doubleword going into E80 storage.
3. A field length exceeding 640 bytes may cause an exit to the address hex 80 greater than the one in FPR 2.

An access exception can occur on a reference to 7080 memory.

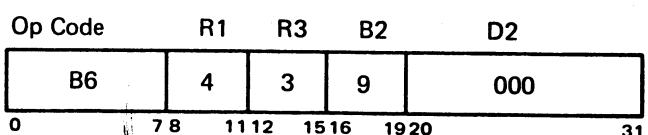
This instruction is retryable until the first word of the sum is stored, which could be the last machine cycle if the sum is contained within a doubleword.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-8 | – | Unchanged. |
| 9 | – | (Model 165) Unchanged. |
| 9 | MAC I CB | (Model 165 II or 168) Original contents restored after use as a work register. |
| 10-15 | – | Unchanged. |
| *FPR* | | |
| 6 | Work register | Unpredictable. |

### ERSU (Emulated Reset and Subtract)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ERSU emulates the 7080 RSU instruction.

*Description*

The numeric field in E80 memory at the address in the MAC I CB (GR 9) is entered into E80 storage starting at the address in the SAR (GR 5).

The ERSU instruction follows the rules of the 7080 RSU instruction, with the following exceptions:

1. If a 905 condition is detected and the 905 check-switch trigger is on, a forced System/370 branch is executed to the address obtained from FPR 2 plus hex 10.
2. The microprogram samples for a pending interruption at the completion of every fifth doubleword going into E80 storage.

An access exception can occur on a reference to 7080 memory.

This instruction is retryable until the first word of the sum is stored, which could be the last machine cycle if the sum is contained within a doubleword.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-8 | - | Unchanged. |
| 9 | – | (Model 165) Unchanged. |
| 9 | MAC I CB | (Model 165 II or 168) Original contents restored after use as a work register. |
| 10-15 | – | Unchanged. |
| *FPR* | | |
| 6 | Work register | Unpredictable. |

check bit in FPR 0 is turned on and a forced System/370 branch is taken to the address hex 10 greater than the one in FPR 2.

4. The microprogram samples for a pending interruption after every fifth doubleword of E80 storage used.
5. If no terminating storage mark is found, the microprogram exits to the address hex 80 greater than the one in FPR 2.

An access exception can occur on a reference to 7080 memory.

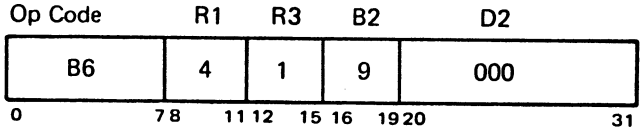
This instruction is retryable until the first word of the sum is stored, which could be the last cycle if the sum is contained within a doubleword.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-8 | – | Unchanged. |
| 9 | - | (Model 165) Unchanged. |
| 9 | MAC I CB | (Model 165 II or 168) Original contents restored after use as a work register. |
| 10-15 | – | Unchanged. |
| *FPR* | | |
| 6 | Work register | Unpredictable. |

## ESUB (Emulated Subtract)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ESUB emulates the 7080 SUB instruction.

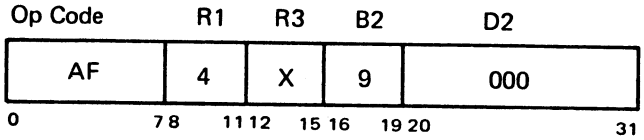
*Description*

ESUB follows the rules of the 7080 SUB instruction with the following exceptions:

1. Inclusion of special characters in the arithmetic fields may not yield the same inconsistent results as in the 7080.
2. A 904 check condition does not terminate the operation, which continues as if the 904 switch were not on. This allows emulator program handling of overflow conditions.
3. If a 904 or 905 check condition is detected and the corresponding check-switch trigger is on, the appropriate

## EAAM (Emulated Add Address to Memory)

*RS Format*




X = Not important

*Function*

EAAM emulates the 7080 AAM instruction.

*Description*

Up to five digits (705 III mode) or six digits (7080 mode) in E80 storage are added to a four-character address in E80 memory to develop an incremented memory address.

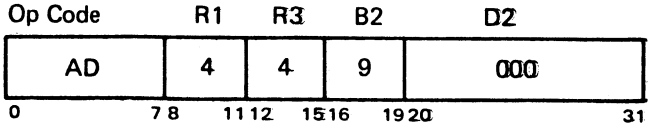
The EAAM instruction follows the rules of the 7080 AAM instruction with one exception: for E80 memory addresses not ending in 4/9, no error check is forced, and the operation goes to completion by treating the address as if it ends in 4/9 in the same memory word.

This instruction is completely retryable.

E80 register contents at the end of operation are unchanged.

### EADM (Emulated Add to Memory)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

EADM, an interruptible instruction, emulates the 7080 ADM instruction.

*Description*

A field in E80 storage is added to a field in E80 memory. The addition is carried out in two different ways, depending on whether the field in E80 memory is signed or unsigned. In either case, the result replaces the memory field.

The EADM instruction follows the rules of the 7080 ADM instruction.

An access exception can occur on a reference to 7080 memory.

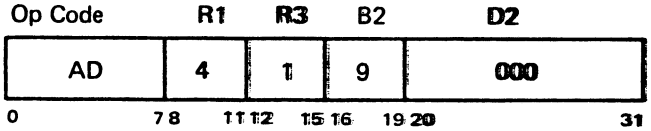
This instruction is retryable in a Model 165 up until the first word of the sum is stored or until a source data refill is needed, and is retryable in a Model 165 II or 168 up to the seventh machine cycle.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-3 | Work registers | Unpredictable. |
| 4-15 | – | Unchanged. |

### ESPR (Emulated Store for Print)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ESPR, an interruptible instruction, emulates the 7080 SPR instruction.

*Description*

A field in E80 storage is placed in E80 memory with discriminative action by the instruction. As with the 7080 SPR instruction, ESPR is normally used to arrange data for printing. The rules followed by ESPR differ from those of the 7080 SPR instruction only in that the microprogram exits to the address hex 80 greater than the one in FPR 2 if no terminating storage mark is found.

An access exception can occur on a reference to 7080 memory.

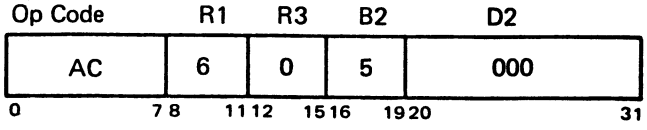
This instruction is completely retryable in a Model 165, and is retryable up to the seventh machine cycle in a Model 165 II or 168. This instruction can be interrupted at any time other than when insignificant characters are being replaced with blanks.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-3 | Work registers | Unpredictable. |
| 4-15 | – | Unchanged. |

### EST (Emulated Store)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

EST emulates the 7080 ST instruction.

*Description*

A numeric field in E80 storage is placed in E80 memory.

The rules followed by EST differ from those of the 7080 ST instruction only in that the microprogram exits to the address hex 80 greater than the one in FPR 2 if no terminating storage mark is found.

An access exception can occur on a reference to 7080 memory.

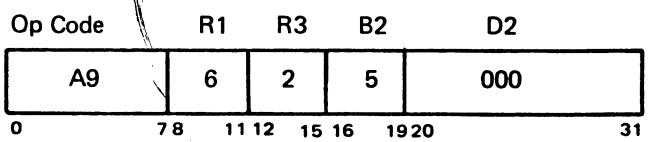
This instruction is retryable up to the storing of the first result field.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-3 | Work registers | Not significant. |
| 4-15 | – | Unchanged. |

### EUFC (Emulated Unload Four Characters)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

EUFC emulates the 7080 UFC instruction.

*Description*

Four characters at the SAR (GR 5) in E80 storage are moved to E80 memory at the location addressed by MAC I CB (GR 9).

EUFC follows the rules of the 7080 UFC instruction with two exceptions:

1. If EUFC is given in 705 I/II or 705 III mode, a System/370 branch is forced to the location of the address hex 20 greater than the contents of FPR 2.
2. The 900 check bit in FPR 0 is not set on. This action identifies an E80 comma instruction given in non-7080 mode.

An access exception can occur on a reference to 7080 memory.

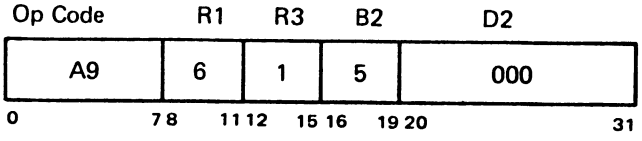
This instruction is completely retryable.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-3 | Work registers | Not significant. |
| 4-15 | – | Unchanged. |

**EULA (Emulated Unload Address)**

*RS Format*




(R3 contains the op-code modifier.)

*Function*

EULA emulates the 7080 ULA instruction.

*Description*

A five-digit (705 III mode) or a six-digit (7080 mode) numeric field in E80 storage is moved to E80 memory as a four-character address field. The rules of the 7080 ULA instruction are followed except that no 4/9 check is performed. The address is treated as if it ended in 4/9 of the same memory word.

An access exception can occur on a reference to 7080 memory.

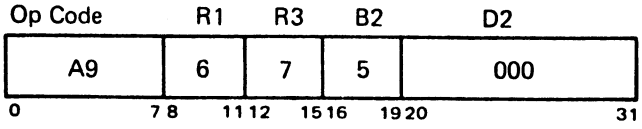
This instruction is completely retryable.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 2-3 | Work registers | Not significant. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6-15 | | Unchanged. |

**EUNL (Emulated Unload)**

*RS Format*




(R3 contains the op-code modifier.)

*Function*

EUNL, an interruptible instruction, emulates the 7080 UNL instruction.

*Description*

The field in E80 storage at the address in the SAR (GR 5) is placed in E80 memory at the address in the MAC I CB (GR 9). The rules of the 7080 UNL instruction are followed, with one exception: the microprogram exits to the address hex 80 greater than the one in FPR 2 if no terminating storage mark is found.

An access exception can occur on reference to 7080 memory.

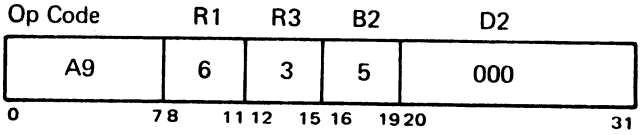
This instruction is completely retryable.

E80 register contents at the end of operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-3 | Work registers | Unpredictable. |
| 4-15 | | Unchanged. |

**EUSB (Emulated Unload Storage Bank)**

*RS Format*




(R3 contains the op-code modifier.)

*Function*

EUSB emulates the 7080 USB instruction.

*Description*

The 256 characters in E80 storage starting at the address in the SAR (GR 5) are placed in E80 memory starting at the address in the MAC I CB (GR 9).

EUSB follows the rules of the 7080 USB instruction with two exceptions:

1. If EUSB is given in 705 I/II or 705 III mode, a System/370 branch is forced to the address hex 20 greater than the one in FPR 2.
2. The 900 check bit in FPR 0 is not set on. This action identifies an E80 comma instruction given in non-7080 mode.

An access exception can occur on a reference to 7080 memory.

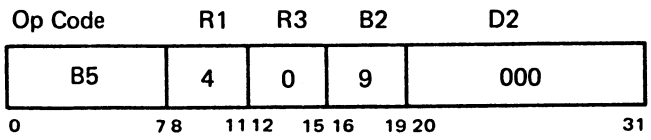
This instruction is completely retryable in a Model 165, and is retryable in a Model 165 II or 168 up to the storing of the first result field.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-3 | Work registers | Unpredictable. |
| 4-15 | – | Unchanged. |

### EBLM (Emulated Blank Memory)
### EBLMS (Emulated Blank Memory Serial)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

EBLM and EBLMS, both interruptible instructions, emulate the 7080 BLM and BLM 01 (BLMS) instructions, respectively.

*Description*

Blank characters are placed in E80 memory, starting at the coded-binary equivalent of the zoned-address in MAC II (GR 11). The number of five-character groups to be blanked is indicated by the numerical value of the E80 address in the IAR (GR 6), modulo 20,000.

EBLM follows the rules of the 7080 BLM with one exception: If the MAC II address ends in other than 4 or 9, the characters contained are blanked, and the MAC II address is increased to the nearest address ending in 4 or 9. EBLMS follows the rules of the 7080 BLM instruction.

An access exception can occur on a reference to 7080 memory.

This instruction is completely retryable in a Model 165, and is retryable in a Model 165 II or 168 up to the fourteenth machine cycle.

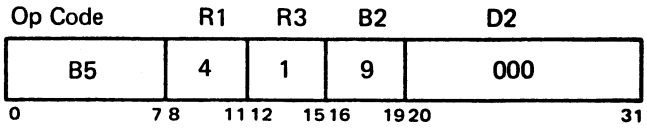
E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unchanged. |
| 5 | SAR | Initial E80 storage address. |
| 6 | IAR | Original IAR converted to zoned-address type B with bits 0, 24, and 25 removed. |
| 7 | SPC | Unchanged. |
| 9 | MAC I CB | Current E80 operand in coded binary format. |
| 11 | MAC II | In zoned-address type B format, the first 4/9 address following the blanked area. |

EBLMS differs from EBLM in that in EBLMS the MAC II address (in GR 11) is one greater than the last E80 memory location blanked.

### ESND (Emulated Send)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ESND, an interruptible instruction, emulates the 7080 SND instruction.

*Description*

Characters are transmitted, five at a time, from the E80 memory location specified by IAR (GR 6) to the E80 address specified by MAC II (GR 11). The number of five-character blocks transmitted will be equivalent to the length of the E80 storage field whose units position is addressed by SAR (GR 5).

ESND follows the rules of the 7080 SND instruction with the following exceptions:

1. The IAR or MAC II address ending in other than 4/9 is increased to the nearest address ending in 4 or 9.
2. If the instruction termination conditions do not exist, the microprogram exits to the address hex 80 greater than the one in FPR 2.

An access exception can occur on a reference to 7080 memory.

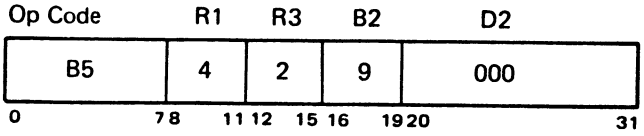
This instruction is retryable in a Model 165 up until the first word is stored, and is retryable in a Model 165 II or 168 up to the fourth machine cycle.

E80 register contents at the end of the operation are:

| GR | Function | Contents |
|---|---|---|
| 2 | Work register | Unpredictable. |
| 3 | Work register | E80 base. |
| 4 | MAC I | Unchanged. |
| 5 | SAR | One greater than the location of the storage address. |
| 6 | IAR | In zoned-address type B format, the first 4/9 address following the last block of five characters sent. |
| 7 | SPC | Unchanged. |
| 8 | | (Model 165) Unchanged. |
| 8 | | (Model 165 II or 168) Original contents restored after GR 8 used as a work register. |
| 9 | MAC I CB | (Model 165) Current E80 operand in coded binary format. |
| 9 | | (Model 165 II or 168) Original contents restored after GR 9 used as a work register. |
| 11 | MAC II | In zoned-address type B format, the first 4/9 address following the last block of five characters received. |

## ETMT (Emulated Transmit)
## ETMTS (Emulated Transmit Serial)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ETMT and ETMTS, both interruptible instructions, emulate the 7080 TMT and TMT 01 (TMTS) instructions, respectively

*Description*

Characters are transmitted from the E80 memory location specified by IAR (GR 6) to the E80 location addressed by MAC II (GR 11).

ETMT and ETMTS follow the 7080 TMT and TMTS instructions respectively, with the following exceptions:

1. The IAR or MAC II address ending in other than 4/9 is increased to the nearest address ending in 4 or 9.
2. The ETMT instruction terminates if a storage mark is found in E80 memory.

An access exception can occur on a reference to 7080 memory

Both instructions are retryable in a Model 165 up to the storing of the first word, and are retryable in a Model 165 II or 168 up to the fourth machine cycle.

The difference between ETMT and ETMTS instructions is recognized via ASU zoning in the previous DIL instruction.

E80 register contents at the end of ETMT operation are:

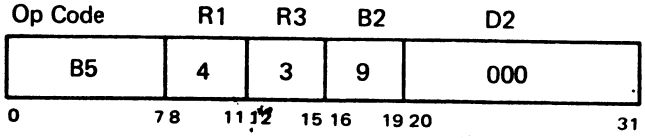
| GR | Function | Contents |
|---|---|---|
| 3 | Work register | E80 base. |
| 4 | MAC I | Unchanged. |
| 5 | SAR | Initial E80 storage address. |
| 6 | IAR | In zoned-address type B format, the first 4/9 address following the last block of five characters transmitted. |
| 7 | SPC | Unchanged. |
| 8 | – | (Model 165) Unchanged. |
| 8 | – | (Model 165 II or 168) Original contents restored after GR 8 used as a work register. |
| 9 | MAC I CB | (Model 165) Current E80 operand in coded binary format. |
| 9 | – | (Model 165 II or 168) Original contents restored after GR 9 used as a work register. |
| 11 | MAC II | In zoned-address type B format, the first 4/9 address following the last block of five characters received. |

E80 register contents at the end of ETMTS operation are:

| GR | Function | Contents |
|---|---|---|
| 0 | Work register | Buffer MAC II data. |
| 1 | Work register | Buffer MAC II data. |
| 2 | Work register | E80 base. |
| 3 | Work register | IAR zoned-address type A of the character one higher than the last character transmitted. |
| 4 | MAC I | Unchanged. |
| 5 | SAR | Storage address one character past storage mark. |
| 6 | IAR | Unpredictable. |
| 7 | SPC | Unchanged. |
| 8 | – | (Model 165) Unchanged. |
| 8 | – | (Model 165 II or 168) Original contents restored after GR 8 used as a work register. |
| 9 | MAC I CB | (Model 165) Current E80 operand in coded binary format. |
| 9 | – | (Model 165 II or 168) Original contents restored after GR 9 used as a work register. |
| 11 | MAC II | In zoned-address type B format, the character one higher than the last character received. |
| *FPR* | | |
| 6 | Work register | Last IAR data field. |

## ETCT (Emulated Ten Character Transmit)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ETCT, an interruptible instruction, emulates the 7080 TCT instruction.

*Description*

Data is moved, in blocks of ten characters, from the E80 memory location specified by IAR (GR 6) to the E80 location addressed by MAC II (GR 11). Transmission continues until a record mark (0001 1010) has been moved from one memory address ending in 9 to another memory address ending in 9.

ETCT follows the rules of the 7080 TCT instruction with the following exceptions:

1. ETCT always starts to transmit with the E80 character at the low-order position of the ten-character block specified by the IAR address and depends on sensing a record mark in byte 4 of every second block of five blocks transmitted.
2. If ETCT is given in 705 I/II or 705 III mode, a System/370 branch is forced to the location of the address specified by hex 20 plus the contents of FPR 2, and the 900 check bit is not set on. Thus an E80 comma instruction is identified in 705 I/II or 705 III mode.
3. If the instruction termination conditions do not exist, the microprogram exits to the address hex 80 greater than the one in FPR 2.
4. ETCT terminates on a storage mark in E80 memory.

An access exception can occur on a reference to 7080 memory.

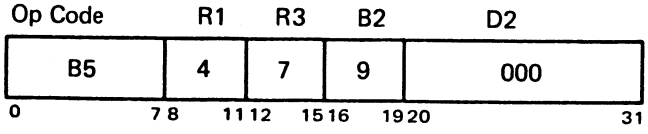
This instruction is retryable in a Model 165 up to the storing of the first word, and retryable in a Model 165 II or 168 up to the fourth machine cycle.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 3 | Work register | E80 base. |
| 4 | MAC I | Unchanged. |
| 5 | SAR | Initial E80 storage address. |
| 6 | IAR | In zoned-address type B format, the first address ending in 9 following the last block of ten characters transmitted. |
| 7 | SPC | Unchanged. |
| 8 | | (Model 165) Unchanged. |
| 8 | - | (Model 165 II or 168) Original contents restored after GR 8 used as a work register. |
| 9 | MAC I CB | (Model 165) Current E80 operand in coded binary format. |
| 9 | - | (Model 165 II or 168) Original contents restored after GR 9 used as a work register. |
| 11 | MAC II | In zoned-address type B format, the first address ending in 9 following the last block of ten characters received. |

## ETCR (Emulated Ten Characters Receive)

*RS Format*




*Function*

ETCR, an interruptible instruction, emulates the 7080 TCR instruction.

*Description*

Data is moved from the IAR address (in GR 6) to the MAC II address (in GR 11). Via programming, the contents of GR 6 and 11 are interchanged before and after execution of the ETCR instruction. Addresses in GR 6 and 11 are not required to have a units digit of 9.

The 903 indicator is not turned on if the instruction is terminated by a group mark (this indicator is set via programming). The terminating character is placed in the rightmost byte of GR 3 so that, via programming, this byte can be analyzed and the 903 indicator can be turned on, if required.

ETCR can be executed in any mode.

An access exception can occur on a reference to 7080 memory.

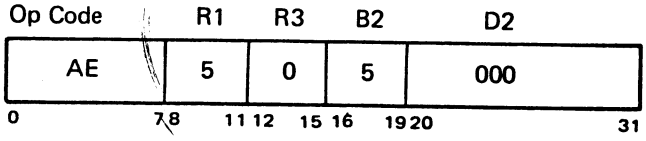
This instruction is retryable in a Model 165 up to the storing of the first word, and retryable in a Model 165 II or 168 up to the fourth machine cycle.

E80 contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 3 | Work register | Terminating character in rightmost byte. |
| 4 | MAC I | Unchanged. |
| 5 | SAR | Unchanged. |
| 6 | IAR | In zoned-address type B format, the first address ending in 9 that follows the last block of ten characters transmitted. |
| 7 | SPC | Unchanged. |
| 9 | MAC I CB | (Model 165) Current E80 operand in coded binary format. |
| 9 | - | (Model 165 II or 168) Original contents restored after GR9 used as a work register. |
| 11 | MAC II | In zoned-address type B format, the first address ending in 9 that follows the last block of ten characters received. |

## ESET (Emulated Set Left)

*RS Format*




(R3 contains the op-code modifer.)

*Function*

ESET, an interruptible instruction, emulates the 7080 SET instruction.

*Description*

ESET establishes the length of a field in E80 storage equal to the value of the address in IAR (GR 6), modulo 256. The units position of the field in storage is addressed by SAR (GR 5).

ESET follows the rules of the 7080 SET instruction with the following exception: interruptions are allowed every eight characters until a storage mark is found. While zeros are being stored, interruptions are allowed every 128 characters.

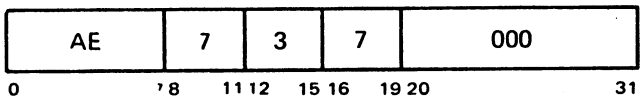
This instruction is retryable up to the ending routine, if a storage mark is not found before the count runs out. Otherwise this instruction is retryable up to the storing of the first word of zeros.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-2 | Work register | Unpredictable. |
| 3 | Work register | Unpredictable (used for interruptions). |
| 4 | MAC I | Unchanged. |
| 5 | SAR | Unchanged. |
| 6 | IAR | Unchanged. |
| 7 | SPC | Unchanged. |
| 8 | | (Model 165) Unchanged. |
| 8 | | (Model 165 II or 168) Original contents restored after GR 8 used as a work register. |
| 9 | MAC I CB | (Model 165) Current E80 operand in coded binary format. |
| 9 | | (Model 165 II or 168) Original contents restored after GR 9 used as a work register. |
| 11 | MAC II | Unchanged. |
| *FPR* | | |
| 6 | Work register | Unpredictable |

**ESHR (Emulated Shorten)**

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ESHR an interruptible instruction, emulates the 7080 SHR instruction

*Description*

ESHR shifts the starting point counter of storage toward higher storage addresses. The number of positions shortened is the value of the E80 address in IAR (GR 6), modulo 256. ESHR follows the rules of the 7080 SHR instruction except the microprogram exits to the address hex 80 greater than the one in FPR 2 if no terminating storage mark exists.

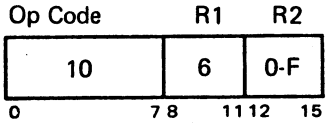
This instruction is retryable up until the new value of SPC is stored in GR 7.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-2 | Work register | Unpredictable. |
| 3 | Work register | Unpredictable for interruptions. |
| 4 | MAC I | Unpredictable. |
| 5 | SAR | Unpredictable. |
| 6 | IAR | Unchanged. |
| 7 | SPC | Current SPC plus number of positions shortened. |
| 9 | MAC I CB | Current E80 operand in coded binary format. |
| 11 | MAC II | Unchanged. |
| *FPR* | | |
| 6 | Work register | Unpredictable. |

**ESPC (Emulated Set Starting Point Counter)**

*RR Format*



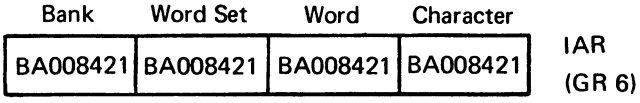

(R2 contents are insignificant.)

*Function*

ESPC emulates the 7080 SPC instruction.

*Description*

ESPC sets the starting point counter to the position of storage addressed by the IAR (GR 6). The four bytes of the IAR address are (from left to right): storage bank, word set, word, and character. The value for bank ranges from 0 to 3, word set from 0 to 7, word from 0 to 3, and character from 0 to 7.




ESPC follows the rules of the 7080 SPC instruction with two exceptions:

1. If ESPC is given in 705 I/II or 705 III mode, a System/370 branch is forced to the address specified by hex 20 plus the contents of FPR 2.
2. The 900 check bit in FPR 0 is not set on.

Either action identifies an E80 comma instruction given in non-7080 mode.

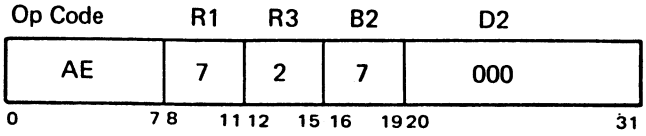
This instruction is completely retryable.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 4 | MAC I | Unchanged. |
| 5 | SAR | Initial E80 storage address. |
| 6 | IAR | Current E80 operand. |
| 7 | SPC | Set to the value of the operand interpreted by the SPC decoder. |
| 9 | MAC I CB | Current E80 operand in coded binary format. |
| 11 | MAC II | Unchanged. |

### ERND (Emulated Round)

*RS Format*




(R3 contains the op-code modifier.)

*Function*

ERND emulates the 7080 RND instruction.

*Description*

ERND adds the binary equivalent, modulo 256, of the E80 address in the IAR (GR 6) to byte 3 of SPC (GR 7) and blocks any carry out of byte 3. In addition, an E80 BCD five is added to the storage character whose address is one byte lower than the final address in SPC (GR 7).

ERND follows the rules of the 7080 RND instruction with the following exceptions:

1. On an overflow condition, two different branches are forced.
   a. For carry out of the high-order position and for the 904 check switch being on automatic, a branch to the base interruption address in FPR 2 plus hex 10 is forced.
   b. For adding five to the storage mark, a branch is forced to the base interruption address plus hex 50. The 904 and ANY indicators are not set.
2. If no terminating storage mark exists, the microprogram exits to the address hex 80 greater than the one in FPR 2.

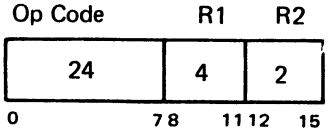
This instruction is retryable up until the new value of SPC is stored in GR 7.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0-3 | Work registers | Unpredictable. |
| 4 | MAC I | Unchanged. |
| 5 | SAR | Unpredictable. |
| 6 | IAR | Unchanged. |
| 7 | SPC | Current SPC plus the number of positions rounded. |
| 9 | MAC I CB | Current E80 operand in coded binary format. |
| 11 | MAC II | Unchanged. |
| *FPR* | | |
| 6 | Work register | Unpredictable. |

### EEIA (Emulated Enable Indirect Address)

*RR Format*




*Function*

EEIA emulates the 7080 EIA instruction.

*Description*

This instruction sets an internal one-bit register (which indicates indirect addressing is required) and links directly to the DIL microprogram routine.

EEIA follows the rules of the 7080 EIA instruction with two exceptions:

1. If EEIA is given in 705 I/II or 705 III mode, a System/370 branch is forced to the address specified by hex 20 plus the contents of FPR 2.
2. The 900 check bit in FPR 0 is not set on.

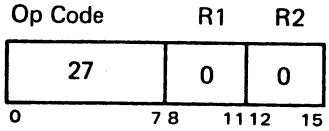
Either exception identifies an E80 comma instruction given in non-7080 mode.

An access exception can occur on a reference to 7080 memory.

This instruction is retryable in a Model 165 up to the third machine cycle, but not retryable in a Model 165 II or 168.

### ETRP (Emulated Transfer on Plus)

*RR Format*




*Function*

ETRP emulates the 7080 TRP instruction.

*Description*

ETRP tests the accumulator (ACC) or auxiliary storage (ASU) sign (FPR 0 bit 6 or 7), the choice depending on the ASU setting. If the sign is +, the branch is made by moving the contents of the IAR to the 7080 IC.

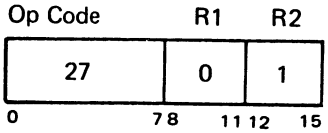
This instruction follows the rules of the 7080 TRP instruction.

ETRP changes the value of the 7080 IC in FPR 4 if a branch is taken; otherwise the instruction terminates.

This instruction is completely retryable in a Model 165, and is retryable in a Model 165 II or 168 up to the seventh machine cycle.

### ETRZ (Emulated Transfer on Zero)

*RR Format*




*Function*

ETRZ emulates the 7080 TRZ instruction.

*Description*

ETRZ tests the accumulator zero indicator (ACC 0) or the auxiliary storage indicator (ASU 0) (FPR 0 bit 1 or 5), the choice depending on the ASU setting. If the zero indicator is on, the branch is made by moving the contents of the IAR to the 7080 IC.

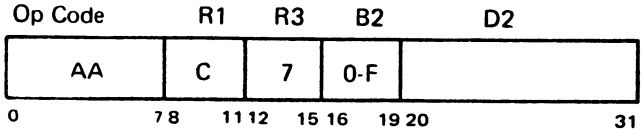
This instruction follows the rules of the 7080 TRZ instruction.

ETRZ changes the value of the 7080 IC in FPR 4 if a branch is taken, otherwise the instruction terminates.

This instruction is completely retryable in a Model 165, and is retryable in a Model 165 II or 168 up to the seventh machine cycle.

## ETR (Emulated Transfer)

*RS Format*




D2 contains a variable whose value depends on programming requirements

B2 + D2 must begin on a word boundary.

*Function*

ETR emulates the 7080 TR instruction.

*Description*

The word specified by C(B2)+D2 is placed in the System/370 IC and in GR 8 if a 7080 TSL condition exists (the emulator not in 705 I/II mode and the SSR in GR 12 equal to 1). This operation causes a branch to a subroutine that simulates a 7080 TSL instruction. If a TSL condition does not exist, the IAR is converted to zoned-address type B and is placed in the 7080 IC.

The ETR instruction is completely retryable. The simulated 7080 TSL instruction is completely retryable in a Model 165 and is retryable in a Model 165 II or 168 up to the fourth machine cycle.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 8 | Subroutine branch address | Address of a TSL routine if a System/370 branch is made; otherwise, the contents are unchanged. |

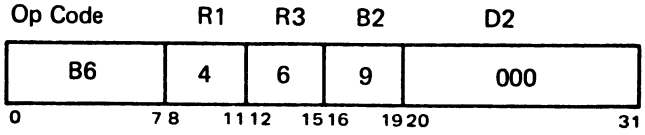
The other GR's remain unchanged.

*FPR*

| | | |
|---|---|---|
| 4 | – | Current IAR contents (replacing the 7080 IC), if a System/370 branch is not made. |

## ETLH (Emulated Table Lookup High or Equal)

*RS Format*




*Function*

ETLH, an interruptible instruction, compares the storage field (addressed by GR 5) with the memory field (addressed by GR 6).

*Description*

Comparison is terminated by a storage mark (located in storage), at which time the result of the compare is placed into bits 6 and 7 of FPR 0 byte 3. If the result of the compare is equal or high, the address of the function (the original GR 6 address less the length of the storage field) is placed in GR 0. If otherwise, the memory field is searched for a group mark or record mark. If a group mark is found, GR 2 is set to 00000001 and the address of the group mark is placed in GR 0. If, instead, a record mark is found, GR 2 is set to FFFFFFFF, and the address one less than the address of the record mark is placed into GR 0.

This instruction can be executed in any mode. Programming tests for 7080 mode before the ETLH instruction is executed. Also, this instruction can be retried up to the ending routine.

An access exception can occur on a reference to 7080 memory.

This instruction is retryable up to the second fetch for storage data or until the search of the function is begun.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0 | Work register | If compare is low, address of group mark if group mark is found, or one less than the address of record mark if a record mark is found. |
| | | If compare is not low, the address of the function. |
| 2 | Work register | 00000001 if the instruction terminates on a group mark. |
| | | FFFFFFFF if the instruction terminates on a record mark. |
| 1 and 3 | Work register | Unpredictable. |
| 4 | MAC I | Zoned-address type B which is five less than the last block of memory referenced. |
| 5 | SAR | Eight greater than the address of the last doubleword of E80 storage to be used. |
| 6 | IAR | Current E80 operand. |
| 7 | SPC | Unchanged. |
| 9 | MAC I CB | (Model 165) Current E80 operand |
| 9 | – | in coded binary format. (Model 165 II or 168) Original contents restored after GR 9 used as a work register. |
| 11 | MAC II | Unchanged. |
| *FPR* | | |
| 0 | Status | Byte 3 bits 6 and 7 are set to the results of the compare. |
| 6 | Work register | Unpredictable. |

## ETLE (Emulated Table Lookup Equal)

*RS Format*

| Op Code | R1 | R3 | B2 | D2 |
|---|---|---|---|---|
| B6 | 4 | 4 | 9 | 000 |
| 0 7 8 | 11 12 | 15 16 | 19 20 | 31 |

*Function*

ETLE, an interruptible instruction, compares the storage field (addressed by GR 5) with the memory field (addressed by GR 6).

*Description*

Comparison is terminated by a storage mark (located in storage), at which time the result of the compare is placed into bits 6 and 7 of FPR 0 byte 3. If the result of the compare is equal, the address of the function (the original GR 6 address less the length of the storage field) is placed in GR 0. If otherwise, the memory field is searched for a group mark or record mark. If a group mark is found, GR 2 is set to 00000001, and the address of the group mark is placed into GR 0. If, instead, a record mark is found, GR 2 is set to FFFFFFFF, and the address one less than the address of the record mark is placed into GR 0.

This instruction can be executed in any mode. Programming tests for 7080 mode before the ETLE instruction is executed. Also, this instruction can be retried up to the ending routine.

An access exception can occur on a reference to 7080 memory.

This instruction is retryable up to the second fetch for storage data or until the search of the function is begun.

E80 register contents at the end of the operation are:

| *GR* | *Function* | *Contents* |
|---|---|---|
| 0 | Work register | If compare is equal, address of the function. If compare is not equal, address of the group mark if the instruction terminates on a group mark, or one less than the address of the record mark if the instruction terminates on a record mark. |
| 2 | Work register | 00000001 if the instruction terminates on a group mark. |
| | | FFFFFFFF if the instruction terminates on a record mark. |
| I and 3 | Work registers | Unpredictable. |
| 4 | MAC I | Zoned-address type B which is five less than the last block of memory referenced. |
| 5 | SAR | Eight greater than the address of the last doubleword of E80 storage to be used. |
| 6 | IAR | Current E80 operand. |
| 7 | SPC | Unchanged. |
| 9 | MAC I CB | (Model 165) Current E80 operand in coded binary format. |
| 9 | – | (Model 165 II or 168) Original contents restored after GR 9 used as a work register. |
| 11 | MAC II | Unchanged. |
| *FPR* | | |
| 0 | Status | Byte 3 bits 6 and 7 are set to the result of the compare. |
| 6 | Work register | Unpredictable. |

# Appendix A. Glossary of Special Terms

Three classes of special terms are used to describe the functions of the emulator: standard 7080 terms, standard System/370 terms, and those terms created solely for the emulator.

## STANDARD 7080 TERMS

Accumulator (ACC)–Storage bank 0.

Address zones–The zone bits over the units and thousands positions of the operand.

Arithmetic and logic unit (ALU)–The central processing unit (CPU).

ASU zones–The zone bits over the tens and hundreds positions of the operand.

Auxiliary storage unit (ASU)–Storage bank 1.

Central storage–Storage which consists of four or five 7080 storage banks, each containing 256 characters.

Channel auxiliary storage unit (CASU)–Storage bank 3.

Character–The 7-bit even-parity BCD combination (six data bits and one parity bit).

Communication storage bank–Storage bank 2.

Instruction–A memory word fetched from memory during I-time.

Internal 7080 registers–The instruction counter (IC), the memory address counter (MAC I), the second memory address counter (MAC II), the select register (SR), the storage address register (SAR), and the starting point counter (SPC).

Memory–7080 main storage.

Memory word–Five characters occupying locations ending in 0/5 through 4/9.

Mode–The operating state of the 7080 ALU, which may be 7080 mode, 705 I/II mode. or 705 III mode.

Operand–The characters located in 1/6 through 4/9 locations of the instruction.

Operation code (op code)–A character located at a 0/5 location in the instruction.

## STANDARD SYSTEM/370 TERMS

Floating-point registers (FPR's)–Four System/370 registers assigned for special use during emulation.

General registers (GR's)–Sixteen System/370 registers, some of which are used to emulate the 7080 internal registers.

Storage–Main storage.

## EMULATOR TERMS

Coded binary–A 24-bit binary address that maps a 7080 address to a unique System/370 address in E80 memory. When a system is operating in relocate mode, any System/370 address generated by the emulator is a virtual address.

Emulation routine–Performs the function of a 7080 operation. The op-code branch executed by the DIL instruction takes each unique 7080 operation to a specific System/370 routine. In this routine the function of this particular 7080 operation is performed.

E80–This terms refers to anything in System/370 that is made to correspond to the 7080 system. For example, System/370 storage used to emulate 7080 memory is called E80 memory.

Op-code branch–A branch taken when a 7080 op code is decoded by special hardware provided by the emulator. This hardware creates a System/370 address which points to the first instruction of the routine that emulates this particular 7080 operation. The DIL instruction provides such linkage.

Work bank–A 256-byte System/370 storage block that has the properties of an E80 storage bank when referred to by the special emulator operations. The four banks available are designated banks 4 through 7.

Zoned-address type A–Bytes 1-4 of the E80 memory word (a System/370 doubleword) with E80 ASU zones set to zeros. All bytes representing zeros have the numeric portion (bits 4-7) set to binary 1010.

Zoned-address type B–Similar to zoned-address type A, except that zeros are represented by the numeric portion of the byte set to binary 0000.

## Appendix B. Emulator Instruction Summary

| *Instruction* | *Mnemonic* | *Op Code* | *R1* | *R2 or R3* | *B2* |
|---|---|---|---|---|---|
| Load Buffer | LB | 0E | 2 | 0 | |
| Load Buffer to Memory Boundary | LBM | 0E | 2 | 1 | |
| Unload Buffer Serial | UBS | 0F | 2 | 0 | – |
| Unload Buffer Parallel | UBP | 0F | 2 | 1 | – |
| Zoned Address Plus Zero | ZAPZ | 09 | R | R | |
| Emulated Set Starting Point Counter | ESPC | 10 | 6 | X | |
| Set Mode | SETMD | 11 | M | M | |
| Branch If | BRIF | 12 | M | 6 | |
| Convert Eighty Register Zeros | CERZ | 13 | R | 3 | |
| Zoned Address to Coded Binary | ZACB | 14 | R | 0 | – |
| Zoned Address Plus One | ZAPON | 14 | R | 4 | |
| Zoned Address Plus Five | ZAPF | 14 | R | 8 | |
| Zoned Address Minus Five | ZAMF | 14 | R | C | |
| Modulo 256 Addition | ATFS | 15 | R | 0 | |
| Convert Select Register | CSELR | 15 | 2 | 1 | |
| Convert and Store SPC | UNSPC | 15 | 7 | 2 | |
| Convert Zoned Address | CINZA | 15 | R | 3 | |
| Branch on 705 I/II Mode | BSOF | 16 | R | 0 | |
| Branch on Indicator | BRIND | 17 | M | M | |
| Emulated Enable Indirect Address | EEIA | 24 | 4 | 2 | |
| Do Interpretive Loop | DIL | 25 | 4 | 2 | |
| Emulated Transfer on Zero | ETRZ | 27 | 0 | 1 | |
| Emulated Transfer on Plus | ETRP | 27 | 0 | 0 | |
| Set Mode Bits Off | MDOFF | A4 | M | M | 7 |
| Set Mode Bits On | MDON | A6 | M | M | 7 |
| Emulated Unload Four Characters | EUFC | A9 | 6 | 2 | 5 |
| Emulated Unload Address | EULA | A9 | 6 | 1 | 5 |
| Emulated Unload | EUNL | A9 | 6 | 7 | 5 |
| Emulated Unload Storage Bank | EUSB | A9 | 6 | 3 | 5 |
| Emulated Transfer | ETR | AA | C | 7 | R |
| DIL Extention | DILEX | AB | C | 7 | R |
| Emulated Store | EST | AC | 6 | 0 | 5 |
| Emulated Store for Print | ESPR | AD | 4 | 1 | 9 |
| Emulated Add to Memory | EADM | AD | 4 | 4 | 9 |
| Emulated Add Address to Memory | EAAM | AF | 4 | X | 9 |
| Emulated Set Left | ESET | AE | 5 | 0 | 5 |
| Emulated Shorten | ESHR | AE | 7 | 3 | 7 |
| Emulated Round | ERND | AE | 7 | 2 | 7 |
| Search and Locate Storage Mark | SALSM | AE | 5 | 5 | 5 |
| Restore E80 Register | RSTER | B3 | R | 3 | 3 |
| Emulated Blank Memory | EBLM | B5 | 4 | 0 | 9 |
| Emulated Send | ESND | B5 | 4 | 1 | 9 |
| Emulated Transmit | ETMT | B5 | 4 | 2 | 9 |
| Emulated Ten Character Transmit | ETCT | B5 | 4 | 3 | 9 |
| Emulated Multiply | EMPY | B6 | 0 | 0 | 9 |
| Emulated Add | EADD | B6 | 4 | 0 | 9 |
| Emulated Subtract | ESUB | B6 | 4 | 1 | 9 |
| Emulated Reset and Add | ERAD | B6 | 4 | 2 | 9 |
| Emulated Reset and Subtract | ERSU | B6 | 4 | 3 | 9 |
| Emulated Compare | ECMP | B6 | 4 | 9 | 9 |
| Emulated Load | ELOD | B6 | 4 | 8 | 9 |
| Emulated Load Address | ELDA | B6 | 4 | D | 9 |
| Emulated Load Storage Bank | ELSB | B6 | 4 | E | 9 |

| *Instruction* | *Mnemonic* | *Op Code* | *R1* | *R2 or R3* | *B2* |
|---|---|---|---|---|---|
| Emulated Load Four Characters | ELFC | B6 | 4 | F | 9 |
| Emulated Ten Character Receive | ETCR | B5 | 4 | 7 | 9 |
| Emulated Table Lookup Equal | ETLE | B6 | 4 | 4 | 9 |
| Emulated Table Lookup High or Equal | ETLH | B6 | 4 | 6 | 9 |

*Notes:*

R = Any general register
M = Mask
X = Unimportant
All other numbers and letters are in hexadecimal

The D2 field is zero for all RS instructions other than ETR and DILEX, in which both B2 and D2 may be different because of programming requirements.

# Appendix C. Indicator and Logout Addresses for Model 165

| *7080 Indicator* | *Indicator Position on Microfiche (Frame B6)* | *Logout Address (Decimal)* | *Bit* |
|---|---|---|---|
| Stat Group 1 Bit 0 | B04 | P*+952 | 03 |
| Stat Group 1 Bit 1 | B05 | P*+952 | 04 |
| Stat Group 1 Bit 2 | B06 | P*+952 | 05 |
| Stat Group 1 Bit 3 | B07 | P*+952 | 08 |
| Stat Group 1 Bit 4 | B08 | P*+952 | 09 |
| Stat Group 1 Bit 5 | B09 | P*+952 | 10 |
| Stat Group 1 Bit 6 | B10 | P*+952 | 11 |
| Stat Group 1 Bit 7 | B11 | P*+952 | 12 |
| WCS Data Register 108 | B15 | P*+952 | 18 |
| WCS Data Register 109 | B16 | P*+952 | 19 |
| WCS Data Register 110 | B17 | P*+952 | 20 |
| WCS Data Register 111 | B18 | P*+952 | 21 |
| WCS Data Register 112 | B19 | P*+952 | 24 |
| WCS Data Register 113 | B20 | P*+952 | 25 |
| Stat Group 2 Bit 0 | C04 | P*+952 | 35 |
| Stat Group 2 Bit 1 | C05 | P*+952 | 36 |
| Stat Group 2 Bit 2 | C06 | P*+952 | 37 |
| Stat Group 2 Bit 3 | C07 | P*+952 | 40 |
| Stat Group 2 Bit 4 | C08 | P*+952 | 41 |
| Stat Group 2 Bit 5 | C09 | P*+952 | 42 |
| Stat Group 2 Bit 6 | C10 | P*+952 | 43 |
| Stat Group 2 Bit 7 | C11 | P*+952 | 44 |
| WCS Data Register 114 | C15 | P*+952 | 50 |
| WCS Data Register 115 | C16 | P*+952 | 51 |
| WCS Data Register 116 | C17 | P*+952 | 52 |
| WCS Data Register 117 | C18 | P*+952 | 53 |
| WCS Data Register 118 | C19 | P*+952 | 56 |
| WCS Data Register 119 | C20 | P*+952 | 57 |
| EMU in Execution | C22 | P*+952 | 59 |
| Stat Group 3 Bit 0 | D04 | P*+960 | 03 |
| Stat Group 3 Bit 1 | D05 | P*+960 | 04 |
| Stat Group 3 Bit 2 | D06 | P*+960 | 05 |
| Stat Group 3 Bit 3 | D07 | P*+960 | 08 |
| Stat Group 3 Bit 4 | D08 | P*+960 | 09 |
| Stat Group 3 Bit 5 | D09 | P*+960 | 10 |
| Stat Group 3 Bit 6 | D10 | P*+960 | 11 |
| Stat Group 3 Bit 7 | D11 | P*+960 | 12 |
| WCS Data Register 120 | D15 | P*+960 | 18 |
| WCS Data Register 121 | D16 | P*+960 | 19 |
| WCS Data Register 122 | D17 | P*+960 | 20 |
| WCS Data Register 123 | D18 | P*+960 | 21 |
| WCS Data Register 124 | D19 | P*+960 | 24 |
| WCS Data Register 125 (Parity) | D20 | P*+960 | 25 |
| Status Valid | E18 | P*+960 | 45 |
| EMLAR Bit 0 | E21 | P*+960 | 56 |
| EMLAR Bit 1 | E22 | P*+960 | 57 |
| EMLAR Bit 2 | E23 | P*+960 | 58 |
| EMLAR Bit 3 | E24 | P*+960 | 59 |
| Adder 80 Arith | F04 | P*+968 | 03 |
| A-side Terminate Latch | F05 | P*+968 | 04 |
| B-side Terminate Latch | F06 | P*+968 | 05 |
| 20K Wrap | F07 | P*+968 | 08 |
| 40K Wrap | F08 | P*+968 | 09 |
| 80K Wrap | F09 | P*+968 | 10 |
| Inc Factor (Inc) | F10 | P*+968 | 11 |
| Inc Factor (±5) | F11 | P*+968 | 12 |
| Gate EMU to W | F15 | P*+968 | 18 |
| Gate EMU to PB | F16 | P*+968 | 19 |
| Gate W to Branch | F17 | P*+968 | 20 |
| EMT to LAL | F18 | P*+968 | 21 |
| EMU WCS Control | G19 | P*+968 | 53 |
| EMT 0 | G21 | P*+968 | 56 |
| EMT 1 | G22 | P*+968 | 57 |
| EMT 2 | G23 | P*+968 | 58 |
| EMT 3 | G24 | P*+968 | 59 |
| WCS Data Error | K15 | P*+982 | 18 |
| WIA Error | K16 | P*+982 | 19 |
| WIB Error | K17 | P*+982 | 20 |

P* Contents of control register 15 (8-31) which is logged out at decimal address 508.

# Appendix D. Indicator and Logout Addresses for Models 165 II and 168

| *7080 Indicator* | *Indicator Position on Microfiche (Frame B6)* | *Logout Address (Decimal)* | *Bit* | *7080 Indicator* | *Indicator Position on Microfiche (Frame B6)* | *Logout Address (Decimal)* | *Bit* |
|---|---|---|---|---|---|---|---|
| Stat Group 1 Bit 0 | B04 | P*+448 | 03 | CS Data Register 123 | D18 | P*+450 | 21 |
| Stat Group 1 Bit 1 | B05 | P*+448 | 04 | CS Data Register 124 | D19 | P*+450 | 24 |
| Stat Group 1 Bit 2 | B06 | P*+448 | 05 | CS Data Register 125 | D20 | P*+450 | 25 |
| Stat Group 1 Bit 3 | B07 | P*+448 | 08 | (Parity) | | | |
| Stat Group 1 Bit 4 | B08 | P*+448 | 09 | Status Valid | E18 | P*+450 | 45 |
| Stat Group 1 Bit 5 | B09 | P*+448 | 10 | EMLAR Bit 0 | E21 | P*+450 | 56 |
| Stat Group 1 Bit 6 | B10 | P*+448 | 11 | EMLAR Bit 1 | E22 | P*+450 | 57 |
| Stat Group 1 Bit 7 | B11 | P*+448 | 12 | EMLAR Bit 2 | E23 | P*+450 | 58 |
| CS Data Register 108 | B15 | P*+448 | 18 | EMLAR Bit 3 | E24 | P*+450 | 59 |
| CS Data Register 109 | B16 | P*+448 | 19 | Adder 80 Arith | F04 | P*+458 | 03 |
| CS Data Register 110 | B17 | P*+448 | 20 | A-side Terminate Latch | F05 | P*+458 | 04 |
| CS Data Register 111 | B18 | P*+448 | 21 | B-side Terminate Latch | F06 | P*+458 | 05 |
| CS Data Register 112 | B19 | P*+448 | 24 | 20K Wrap | F07 | P*+458 | 08 |
| CS Data Register 113 | B20 | P*+448 | 25 | 40K Wrap | F08 | P*+458 | 09 |
| Stat Group 2 Bit 0 | C04 | P*+448 | 35 | 80K Wrap | F09 | P*+458 | 10 |
| Stat Group 2 Bit 1 | C05 | P*+448 | 36 | Inc Factor (Inc) | F10 | P*+458 | 11 |
| Stat Group 2 Bit 2 | C06 | P*+448 | 37 | Inc Factor (±5) | F11 | P*+458 | 12 |
| Stat Group 2 Bit 3 | C07 | P*+448 | 40 | Gate EMU to W | F15 | P*+458 | 18 |
| Stat Group 2 Bit 4 | C08 | P*+448 | 41 | Gate EMU to PB | F16 | P*+458 | 19 |
| Stat Group 2 Bit 5 | C09 | P*+448 | 42 | Gate W to Branch | F17 | P*+458 | 20 |
| Stat Group 2 Bit 6 | C10 | P*+448 | 43 | EMT to LAL | F18 | P*+458 | 21 |
| Stat Group 2 Bit 7 | C11 | P*+448 | 44 | EMU CS | G19 | P*+458 | 53 |
| CS Data Register 114 | C15 | P*+488 | 50 | EMT 0 | G21 | P*+458 | 56 |
| CS Data Register 115 | C16 | P*+448 | 51 | EMT 1 | G22 | P*+458 | 57 |
| CS Data Register 116 | C17 | P*+448 | 52 | EMT 2 | G23 | P*+458 | 58 |
| CS Data Register 117 | C18 | P*+448 | 53 | EMT 3 | G24 | P*+458 | 59 |
| CS Data Register 118 | C19 | P*+448 | 56 | Page Check Addr Bit 20 | H04 | P*+460 | 03 |
| CS Data Register 119 | C20 | P*+448 | 57 | Page Check Page 00 | H05 | P*+460 | 04 |
| EMU in Execution | C22 | P*+448 | 59 | Page Check Page 01 | H06 | P*+460 | 05 |
| Stat Group 3 Bit 0 | D04 | P*+450 | 03 | Page Check Page 10 | H07 | P*+460 | 08 |
| Stat Group 3 Bit 1 | D05 | P*+450 | 04 | Page Check Page 11 | H08 | P*+460 | 09 |
| Stat Group 3 Bit 2 | D06 | P*+450 | 05 | Page Remember Addr | | | |
| Stat Group 3 Bit 3 | D07 | P*+450 | 08 | Bit 20 | J04 | P*+460 | 35 |
| Stat Group 3 Bit 4 | D08 | P*+450 | 09 | Page Remember Page 00 | J05 | P*+460 | 36 |
| Stat Group 3 Bit 5 | D09 | P*+450 | 10 | Page Remember Page 01 | J06 | P*+460 | 37 |
| Stat Group 3 Bit 6 | D10 | P*+450 | 11 | Page Remember Page 10 | J07 | P*+460 | 40 |
| Stat Group 3 Bit 7 | D11 | P*+450 | 12 | Page Remember Page 11 | J08 | P*+460 | 41 |
| CS Data Register 120 | D15 | P*+450 | 18 | CS Data Error | K15 | P*+468 | 18 |
| CS Data Register 121 | D16 | P*+450 | 19 | WIA Error | K16 | P*+468 | 19 |
| CS Data Register 122 | D17 | P*+450 | 20 | WIB Error | K17 | P*+468 | 20 |

P* = Contents of control register 15 (8-31) which is logged out at decimal address 508.

# Appendix E. EBCDIC and E80 BCD Conversion Tables

Bit Positions 0123 ↓ / Bit Positions 4567

| | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | # | @ | ' | = | " |
| 0100 | Blank | / | S | T | U | V | W | X | Y | Z | : | , | % | _ | > | ? |
| 1000 | - | J | K | L | M | N | O | P | Q | R | ! | $ | * | ) | ; | ¬ |
| 1100 | & | A | B | C | D | E | F | G | H | I | ¢ | . | < | ( | + | \| |

E80 BCD Character Set

EBCDIC Bit Order 0 1 2 3 4 5 6 7

Bit Positions 0123 ↓ / Bit Positions 4567

| | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | Blank | A | B | C | D | E | F | G | H | I | ¢ | . | < | ( | + | \| |
| 0001 | & | J | K | L | M | N | O | P | Q | R | ! | $ | * | ) | ; | ¬ |
| 0010 | - | / | S | T | U | V | W | X | Y | Z | : | , | % | _ | > | ? |
| 0011 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Blank | # | @ | ' | = | " |
| 0100 | | A | B | C | D | E | F | G | H | I | ¢ | | | | | |
| 0101 | | J | K | L | M | N | O | P | Q | R | ! | | | | | |
| 0110 | | | S | T | U | V | W | X | Y | Z | : | | | | | |
| 0111 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Blank | | | | | |
| 1000 | ¢ | A | B | C | D | E | F | G | H | I | Blank | . | < | ( | + | \| |
| 1001 | ! | J | K | L | M | N | O | P | Q | R | & | $ | * | ) | ; | ¬ |
| 1010 | : | / | S | T | U | V | W | X | Y | Z | - | , | % | _ | > | ? |
| 1011 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Blank | # | @ | ' | = | " |
| 1100 | | | | | | | | | | | Blank | . | < | ( | + | \| |
| 1101 | | | | | | | | | | | & | $ | * | ) | ; | ¬ |
| 1110 | | / | | | | | | | | | - | , | % | _ | > | ? |
| 1111 | | | | | | | | | | | Blank | # | @ | ' | = | " |

7080 emulation uses the preferred character set (shown in the shaded areas).

EBCDIC-to-E80-BCD Conversion

GA22-6963-1

7080 Compatibility Feature for IBM System/370 Models 165, 165 II, and 168 (S/370-13) Printed in U.S.A. GA22-6963-1

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)**

7080 Compatibility Feature for IBM System/370
Models 165, 165 II, and 168

**READER'S COMMENT FORM**

Order No. GA22-6963-1

*Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action.* Using this form to request system assistance or additional publications, however, will delay response. *For more direct handling of such request, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

What is your occupation? _____

Number of latest Technical Newsletter (if any) concerning this publication: _____

Please indicate in the space below if you wish a reply.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Cut or Fold Along Line

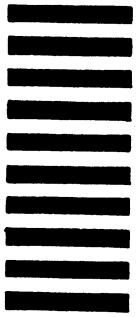
## Your comments, please . . .

This manual is a reference source for systems analysts, programmers and operators of IBM systems. Your comments will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold Fold

FIRST CLASS
PERMIT NO. 419
POUGHKEEPSIE, N.Y.

**Business Reply Mail**
**No postage stamp necessary if mailed in U.S.A.**



**Postage will be paid by:**

**International Business Machines Corporation**
**Department B98**
**P.O. Box 390**
**Poughkeepsie, New York 12602**

Fold Fold

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)**

Cut or Fold Along Line

7080 Compatibility Feature for IBM System/370 Models 165, 165 II, and 168 (S/370-13) Printed in U.S.A. GA22-6963-1