IBM 8112 Central Processing Unit

Preliminary Manual of Operation
January 15, 1961

IBM CONFIDENTIAL

IBM Product Development Laboratory
Poughkeepsie, N. Y.

IBM 8112 Central Processing Unit

The IBM 8112 Central Processing Unit can be attached to an IBM 8106 Data Processing System. It greatly enhances the performance of the 8106 system in scientific computing applications.

The 8112 is basically a high-speed floating point arithmetic unit. It contains its own instruction-fetching, indexing and instruction sequencing mechanisms. These facilities allow it to execute extended portions of a program as a logically independent computer. During the time that the 8112 is in operation, the remaining facilities of the 8106 Data Processing System may continue to process other portions of the same program or other completely independent programs.

The 8112 has sufficient arithmetical, logical, and data-manipulative abilities to execute all statements of a Fortran-specified problem except those dealing with input or output. Input, output, and their associated editing are performed by the other parts of the 8106 system.

System Organization

The 8112 operates as an independent computer under control of the 8106. A typical 8106 system configuration incorporating the 8112 is shown in Figure 1.

The 8112 Central Processing Unit is connected directly to the same storage bus that connects the 8106 Central Processing Unit to the core storage units. Traffic on this bus is controlled by the 8106. Highest priority is given to storage requests originating from input-output operations. Next priority is given to requests from the 8112. Lowest priority is given to requests originating from processing operations of the 8106. An exception is made to these priorities if an interruption is waiting in the 8106 or if the 8106 is in the disabled mode. In this case, the 8112 is given lowest priority.

In the system shown in Figure 1, the processing rates of the 8106 and 8112 may be slowed somewhat by conflicting requests for the storage bus. The performance of the system can be increased by avoiding these conflicts through the use of a separate storage bus for the 8112, as shown in Figure 2. The second storage bus is made possible by the Shared Core Storage Unit.

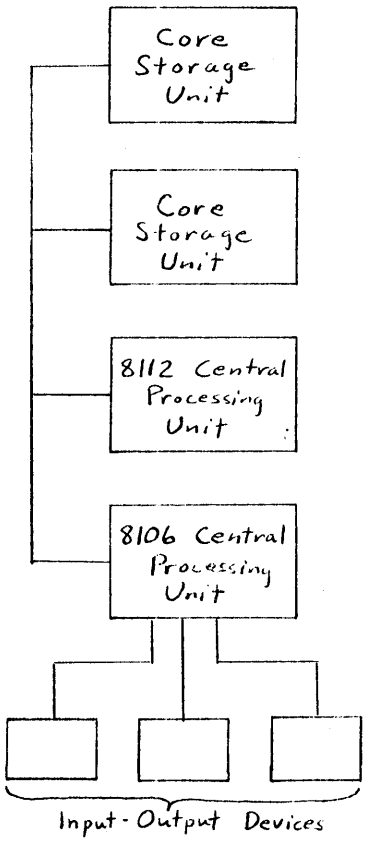


Figure 1

Typical 8106 System Including an 8112 Central Processing Unit

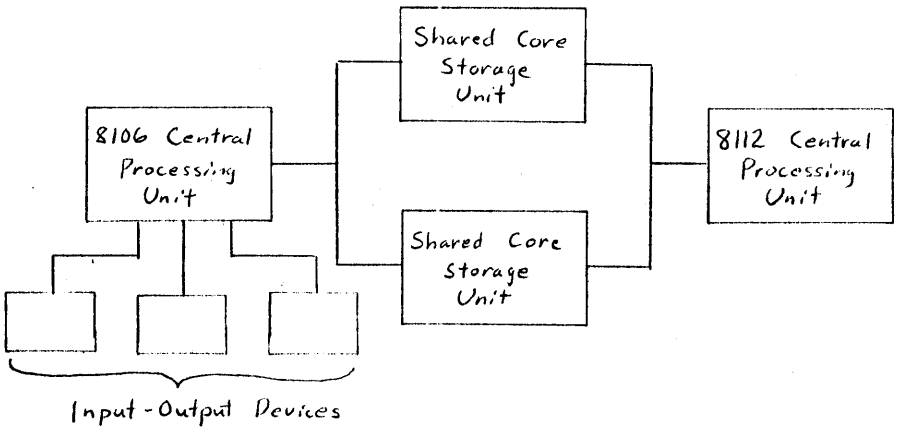


Figure 2

Typical 8106 System Using Two Storage Buses

Control of the 8112 is provided by the 8106 through an input-output channel. Thus the 8112 appears to the 8106 as an input-output device. The 8112 is started by a CONTROL instruction executed by the 8106. When the 8112 stops, it sends an interruption signal to the 8106, just as an input-output device would.

The 8112 has no input-output facilities. Thus, it must always be part of a larger system. Input and output are controlled by the 8106. Since both central processing units can refer to the same core storage locations, there is no need to transmit the input-output data from the 8106 to the 8112, or vice-versa. In addition to controlling the input-output devices, the 8106 also performs any necessary data editing and code translation, a function for which it is particularly well adapted.

Core Storage

Core storage is organized into words of 64 information bits and 8 check bits. The check bits are used as single-error-detecting parity bits on each group of eight bits. Such a group is called a byte. Multiple core storage units can be attached up to a maximum capacity of 65,536 (2^{16}) words.

The core storage units normally used in the system have a complete read-write cycle time of 2.0 microseconds. If higher system performance is desired, core storage units with a cycle time of 0.75 microseconds may be substituted for the slower units.

In addition to the main core storage described above, the 8112 contains an internal high speed core storage. This storage has a read or write time of 0.3 microseconds. It is used in such a way that in many cases only a read or a write is required, and not a complete regenerative read-write cycle. This core storage contains the 255 index registers and the 16 levels of last-in-first-out storage described below.

Central Processing Unit

Two characteristics of scientific calculations are that intermediate results of calculations are often used repeatedly as factors in later calculations, and that most of the elementary operations involved combine two factors to produce a single result . The last-in-first-out (LIFO) storage exploits these characteristics to increase performance by reducing the number of required references to the main core storage.

The LIFO storage is made up of 16 words of high speed core storage. Each word is referred to as a level; the levels are numbered from L_0 through L_{15}. (The LIFO storage is also known as a "pushdown accumulator".) It operates in the following way. Load-type operations cause the contents of each level

to be pushed down to the next higher-numbered level. The contents of the addressed storage location then are placed in the vacated level L_0. Store-type operations perform the inverse action, causing the contents of L_0 to be stored at the addressed storage location, and then pushing up the contents of each of the other levels into the next lower-numbered level. Arithmetic operations combine the contents of the two highest levels L_0 and L_1 to produce a result that is placed in L_1. The contents of each level are then pushed up to the next lower-numbered level, and the contents of L_0 are discarded.

Thus, if intermediate results are not stored, but left in L_0 before succeeding operands are loaded, these results will be pushed down to lower levels of the LIFO storage. When the intermediate results are needed as factors in later calculations, they often have come back up to the top as a natural consequence of the pushups involved in intervening calculations. If they have not, TOP or SWAP operations will rapidly bring them into the upper levels where they can take part in arithmetic operations.

The processes of pushing up and pushing down do not require actual transfers of information between the core storage locations that form the LIFO storage. Instead, a count is kept of the number of active levels. This count serves as a pointer to indicate the storage location that is currently the highest level, L_0. Other levels are addressed relative to this count. The count also serves to indicate whether all 16 levels are full. If they are, and a pushdown occurs, an alert is given to signal the loss of information from the lowest level. Sixteen levels are quite adequate for most computations.

Since the arithmetic instructions always imply operands in specific locations of the LIFO storage, they do not require addresses. On the other hand, load and store instructions must specify the addresses of locations in the main core storage. Therefore, a variable-length instruction format is used. Instructions may consist of from one to five eight-bit bytes. Arithmetic operations require only one byte. Manipulative operations require one or two bytes. Index operations require one, two, or three bytes. Load, store, and branch operations require from one to five bytes.

Sequencing of instructions in the 8112 Central Processing Unit is controlled by a program control word similar to that in the 8106 Central Processing Unit. This program control word contains the instruction address, instruction count, prefix, indicators, mask, condition register, and several control bits. The instruction address specifies the address of the next instruction to be executed. The prefix specifies a base address in main core storage. Several control locations in storage are located relative to this prefix. The prefix also determines the protected area of core storage. The other parts of the program control word will be described later.

Control for the 8112 is provided from the 8106 by connecting the 8112 to a standard input-output channel on the 8106. If the 8112 is not operating, it is started by the 8106 issuing a CONTROL instruction to this channel. This operation sends one byte of information to the 8112. If the byte is zero, the 8112 continues operating at the point where it previously stopped. If the byte is non-zero, it is treated as a prefix, and a new program control word is loaded from a location relative to this prefix. If bit 24 of this new program control word is one, the LIFO storage is reset (no active levels), bit 24 is set to zero, and the 8112 starts executing instructions at the address specified in the program control word. If bit 24 of the new program control word is zero, the LIFO storage is not reset, but otherwise the operation proceeds as just described.

The 8112 may stop executing instructions in three ways. It may be stopped by the completion of its program (signalled by a STOP instruction), by the occurrence of an exceptional condition in its program, or by a RELEASE instruction issued by the 8106. In each case, the 8112 will store its current program control word in a location relative to the prefix and then send an interruption signal to the 8106. In the first case this is a Channel End interruption. In the latter two cases it is a Channel Special Condition interruption.

Instruction Set

Only two instructions address data directly in main core storage. These are LOAD and STORE. A variety of addressing and indexing modes are provided for these instructions . LOAD performs a pushdown and then places the contents of L_0 in the addressed location and then performs a pushup. STORE places the contents of L_0 in the addressed location and then performs a pushup.

One other type of operation addresses a location in main core storage. This is the BRANCH operation. This operation specifies a test of the bits of the condition register, and then if the test is successful, the effective address of the instruction replaces the instruction address in the program control word. The bits of the condition register reflect the value of the current contents of L_0 .

Both single and double precision floating point operations are provided. The single precision operations operate on the contents of L_0 and L_1 to produce a result in L_1, followed by a pushup. The single precision operations are add, subtract, multiply, and divide. They produce results identical to those produced by the analogous operations in the 8106 Central Processing Unit. Double precision operations take two successive levels of the LIFO storage to hold a single factor. The high-order portion of the floating point fraction and the exponent of the quantity are in the lower-numbered level. The low-order portion of the fraction and an exponent 16 less than that of the entire

quantity are in the higher-numbered level. Add, subtract, and multiply operations are provided that operate on two single precision quantities to produce a double precision result. A divide operation divides a double precision quantity by a single precision quantity to produce single precision quotient and remainder. Add and subtract operations are also provided that add two double precision quantities to produce a double precision result. All floating point operations are available in both normalized and unnormalized modes and may be run in a noisy mode to check precision of results.

Addresses of LOAD, STORE, and BRANCH operations may be modified by the contents of any of 255 index registers. Instructions are provided to store the contents of L_0 in an index register or to load L_0 from an index register. Other instructions add or subtract the contents of an index register from the contents of L_0 or the contents of another index register. The instruction address of the program control word can also be placed in an index register.

A large number of instructions provide for manipulating the contents of the LIFO storage. The highest level may be loaded from any of the others. The contents of either of the two highest levels can be swapped with the contents of any of the other levels. A pushup or pushdown can be executed independent of any other operation. Conversely, the pushup or pushdown normally associated with an operation can be inhibited. These operations allow the intermediate results kept in the LIFO storage to be brought to the proper levels for subsequent calculations.

Several operations provide for manipulating the sign and exponent positions of L_0. The exponent of L_0 may also be brought into the fraction portion of L_1 for subsequent arithmetic or shifting operations. The inverse operation is also provided. A variety of shifting operations is available for normalization, editing, and logical operations.

For control of the computer, the condition register may be loaded with a four-bit byte from L_0 for testing. The program control word can be loaded into L_0 for testing. The mask and indicator bits of the program control word can be altered. The entire set of instructions is listed in the appendix.

Features

Four modes of addressing are provided for LOAD, STORE, and BRANCH operations. These are direct, direct indexed, indirect, and indirect indexed. In the direct mode, the effective address is taken directly from the instruction. In the direct indexed mode, the contents of a signed index quantity is added to the address in the instruction to form the effective address. In the indirect mode, the instruction contains no address. Instead, the contents of L_0 (high-order portion of fraction) is used as the effective address. In the indirect indexed mode the sum of the contents of L_0 and the contents of an index register is used as the effective address.

A program interruption system is provided to alert the 8112 to the occurrence of special circumstances during the execution of a program. Such circumstances include the production of infinitesimal or infinite results, invalid addresses or operation codes, and machine malfunctions. Such an occurrence turns on an indicator. There are two groups of indicators. Those in the program control word have corresponding mask bits. They cause an interruption only if the mask bit is on. There is also an additional group of indicators that cause an interruption whenever they are turned on. These are kept in a separate control word that is loaded from a location relative to the prefix when the 8112 is started and stored there when it is stopped, just as the program control word is loaded and stored. When an interruption occurs, the 8112 is stopped and a Channel Special Condition interruption signal is sent to the 8106 Central Processing Unit. It is the function of the 8106 to service the interruption, using the indicator bits placed in core storage by the 8112 to determine the cause.

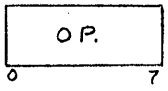
When a problem program is being executed under control of a supervisory control program, it is desirable to be able to protect the supervisory program and any other problem programs that may be in storage from inadvertent changes by the problem program being executed. Such protection is provided by the address monitoring feature. Each address that is used to store information in core storage is compared with the prefix address. If it is less than the prefix address, the storage is suppressed and an indicator is turned on. Thus, all storage locations below the prefix address are protected.

As an aid in checking out newly written programs, an instruction count feature is provided. The program control word contains a count field. Each time an instruction is executed, this count is decreased by one. When the count reaches zero, an indicator is turned on, causing an interruption. This feature may be used in a variety of ways. By setting the count to one (an operation performed by the 8106 before starting the 8112) a single instruction will be executed, and the 8112 will then stop and return control to the 8106. This technique is useful in tracing programs. Another technique is to set the count to a large number and then run the 8112 until trouble develops. The value of the count field at that point can be compared with the original value. By starting again with a count slightly less than the difference, the 8112 will then stop just before the trouble develops, and the program can be traced from that point.

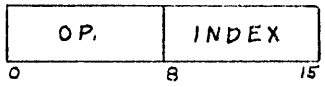
APPENDIX

Instruction Formats

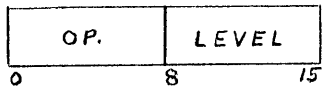
A



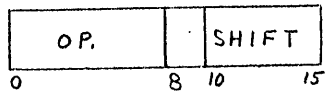
B



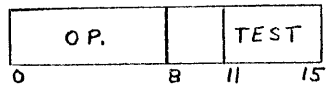
C



D

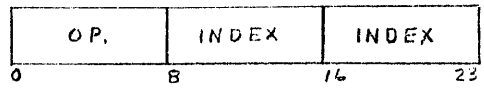


E

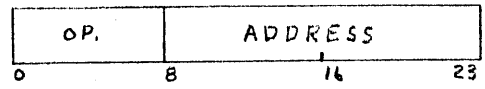


F

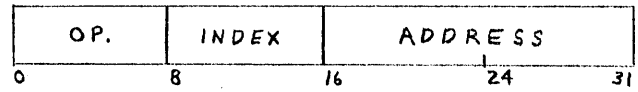
G



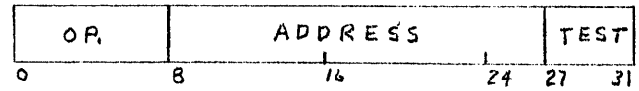
H



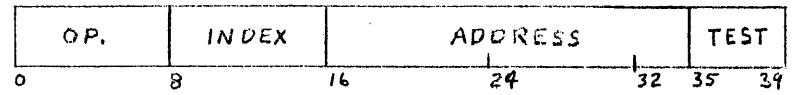
I



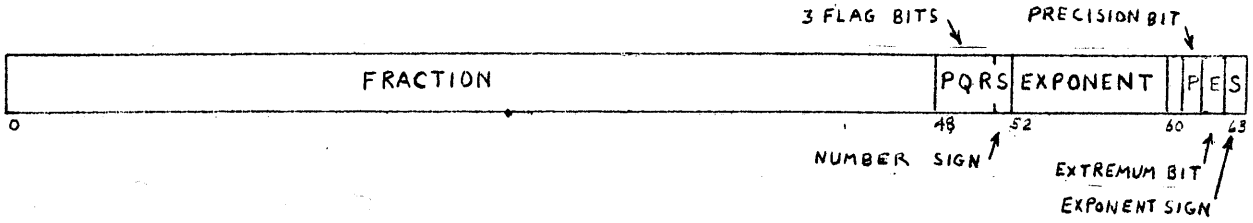
J



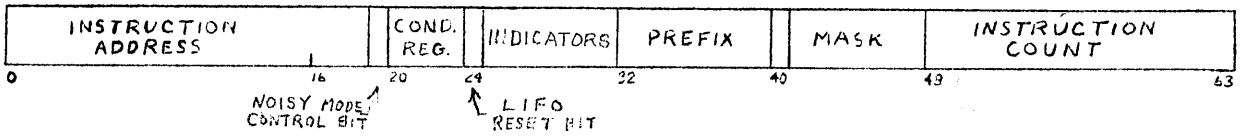
K



Floating Point Data Format



Program Control Word Format



Instruction List

There are five operations specifying addresses. These are as follows:

Instruction	Operation
LD	The full word at the location specified by the effective address is placed in L_0 following the pushdown ($L_i \rightarrow L_{i+1}$). The condition register is set but no change is made in the Maskable indicators.
LDF	This is the same as LD except that the maskable indicators are set if necessary.
ST	The full word in L_0 is stored in the location specified by the effective address. This is followed by a pushup ($L_{i+1} \rightarrow L_i$) and the condition register reflects the final contents of L_0.
BR	The five low-order bits of this instruction specify the type of branch. Four bits are a mask for the condition register. These bits are "anded" with the condition register and their outputs are "ored" together. The other bit specifies Any or None and is tested against the "ored" output to determine whether the branch is to be taken.
BRU	This is the same as BR except that after testing the condition register pushup ($L_{i+1} \rightarrow L_i$) occurs and the condition register is set by the new contents of L_0.

Four modes of addressing are available for these instructions:

D	Direct
DX	Direct Indexed
I	Indirect
IX	Indirect Indexed

These combine to form the following instructions:

Instructions	Format	Effective Address
LD, D	H	(Address)
LD, DX	I	(Address) + C(X)
LD,I	A	C (L_0)
LD, IX	B	C (L_0) + C(X)
LDF, D	H	(Address)
LDF, DX	I	(Address) + C(X)
LDF, I	A	C(L_0)
LDF, IX	B	C(L_0) + C(X)
ST,D	H	(Address)
ST, DX	I	(Address) + C(X)
ST, I	A	C(L_0)
ST, IX	B	C(L_0) + C(X)
BR, D	J	(Address)
BR, DX	K	(Address) + C(X)
BR, I	E	C(L_0)
BR, IX	F	C(L_0) + C(X)
BRU, D	J	(Address)
BRU, DX	K	(Address) + C(X)
BRU, I	E	C(L_0)
BRU, IX	F	C(L_0) + C(X)

Note that the I or IX modifier implies a pushup as C (L_0) or C (L_0) + C(X) are transferred to the effective address.

The remaining instructions contain no addresses. They may be divided into several groups.

Index Instructions

Instruction	Format	Operation
SIX	B	Store 0 - 21 and fraction sign of L_0 in specified index, $L_{i+1} \rightarrow L_i$.
LFX	B	$L_i \rightarrow L_{i+1}$, clear L_0, insert contents of index in 0 - 21 and fraction sign of L_0.

Instruction	Format	Operation
AFX	B	$L_0 + C(X) \rightarrow L_0$
SFX	B	$L_0 - C(X) \rightarrow L_0$
ICX	B	The 19 bit instruction counter (plus 1 in the Word Address part) is stored in the specified index.
AXX	G	The sum of the contents of the two indexes replaces the first index.
AXXL	G	The same as AXX followed by LFX from the first index.
SXX	G	The same as AXX except subtract.
SXXL	G	The same as AXXL except subtract.

Floating Point Instructions

A normalized or unnormalized modifier may be applied to any of the floating point operations.

Single Precision

FA	A	Floating Add: $L_0 + L_1 \rightarrow L_1$, $L_{i+1} \rightarrow L_i$
FS	A	Floating Subtract: $L_0 - L_1 \rightarrow L_1$, $L_{i+1} \rightarrow L_i$
FM	A	Floating Multiply: $L_0 * L_1 \rightarrow L_1$, $L_{i+1} \rightarrow L_i$
FD	A	Floating Divide: $L_0 / L_1 \rightarrow L_1$, $L_{i+1} \rightarrow L_i$

Double Precision

DFA	A	Double Floating Add: $L_0 + L_1 \rightarrow (L_0, L_1)$
DFS	A	Double Floating Subtract: $L_0 - L_1 \rightarrow (L_0, L_1)$
DFM	A	Double Floating Multiply: $L_0 * L_1 \rightarrow (L_0, L_1)$
DFD	A	Double Floating Divide: $(L_0, L_1) / L_2$ gives remainder in L_1, quotient in L_2 followed by $L_{i+1} \rightarrow L_i$
FAD	A	Floating Add Double: $(L_0, L_1) + (L_2, L_3) \rightarrow (L_2, L_3)$, then $L_{i+2} \rightarrow L_i$
FSD	A	Floating Subtract Double: $(L_0, L_1) - (L_2, L_3) \rightarrow (L_2, L_3)$, then $L_{i+2} \rightarrow L_i$

Manipulative Operations

Instruction	Format	Operation
TOP0	A	$L_i \rightarrow L_{i+1}$, $L_1 \rightarrow L_0$
TOP1	A	$L_i \rightarrow L_{i+1}$, $L_2 \rightarrow L_0$
TOP2	A	$L_i \rightarrow L_{i+1}$, $L_3 \rightarrow L_0$
TOPy	C	$L_i \rightarrow L_{i+1}$, $Ly_{+1} \rightarrow L_0$
SWP01	A	Interchange $L_0 + L_1$
SWP02	A	Interchange $L_0 + L_2$
SWP03	A	Interchange $L_0 + L_3$
SWP0y	C	Interchange $L_0 + Ly$
SWP12	A	Interchange $L_1 + L_2$
SWP13	A	Interchange $L_1 + L_3$
SWP14	A	Interchange $L_1 + L_4$
SWP1y	C	Interchange $L_1 + Ly$
UP	A	$L_{i+1} \rightarrow L_i$
FRZ	A	Do not perform a pushup or down if required by the next operation.
DNC	A	$L_i \rightarrow L_{i+1}$, $0 \rightarrow L_0$
IVS	A	Invert fraction sign of L_0
SSP	A	Set fraction sign in L_0 to plus
SSM	A	Set fraction sign in L_0 to minus
XPF	A	Store L_0 exponent into L_0 fraction
FXP	A	Store L_0 fraction into L_0 exponent
IXP	A	Interchange exponents of L_0 and L_1
ML	A	Make logical: Shift L_1 fraction right 16 bits and insert low order 16 bits of L_0 (exponent and signs) into the vacated bits.
MF	A	Store the 16 high order bits of L_1 fraction in the exponent and sign positions of L_0.
LCR	A	Load condition register with the 4 high order bits of L_0.
LFPW	A	Load the program control word into L_0.
LMK	A	The seven PCW mask bits are replaced by the corresponding bits of L_0.
LIN	A	The seven PCW indicator bits are replaced by the corresponding bits of L_0.
NOOP	A	No Operation.
STOP	A	The PCW is stored in memory at prefix plus two, additional indicators are stored at prefix plus three and the 8112 stops.

Instruction	Format	Operation
RSN	D	The fraction part of L_0 is shifted right as specified by the six low order bits of the second byte.
RSX	B	The fraction part of L_0 is shifted right as specified by the bit address part of the index specified in the second byte.
LSN	D	The same as RSN except left shift.
LSX	B	The same as RSX except left shift.
RDN	D	The fraction parts of L_0 and L_1 are shifted right as specified by the six low order bits of the second byte.
RDX	B	The fraction parts of L_0 and L_1 are shifted right as specified by the bit address of the index specified in the second byte.
LDN	D	The same as RDN except left shift.
LDX	B	The same as RDX except left shift.

Condition Register

The bits of the condition register are set as follows to reflect the value of the result in L_0.

1. Result less than zero
2. Result zero
3. Result greater than zero
4. Result infinite

Immediately following an LCR instruction, the condition register reflects the value of the byte loaded, rather than the entire contents of L_0.

Indicators

Maskable indicators

PF	Data Flag P
QF	Data Flag Q
RF	Data Flag R
GEP	Generated Extremum Positive
GEN	Generated Extremum Negative
OR	Oversized Result
ZD	Zero divisor

Non-maskable indicators

ICS	Instruction Count Signal
AD	Address Invalid
DS	Data Store
OP	Operation Code Invalid
MI	Maskable indicator
DE	Detected Parity Error
IA	Improper LIFO Operation
RL	Release Instruction from 8106