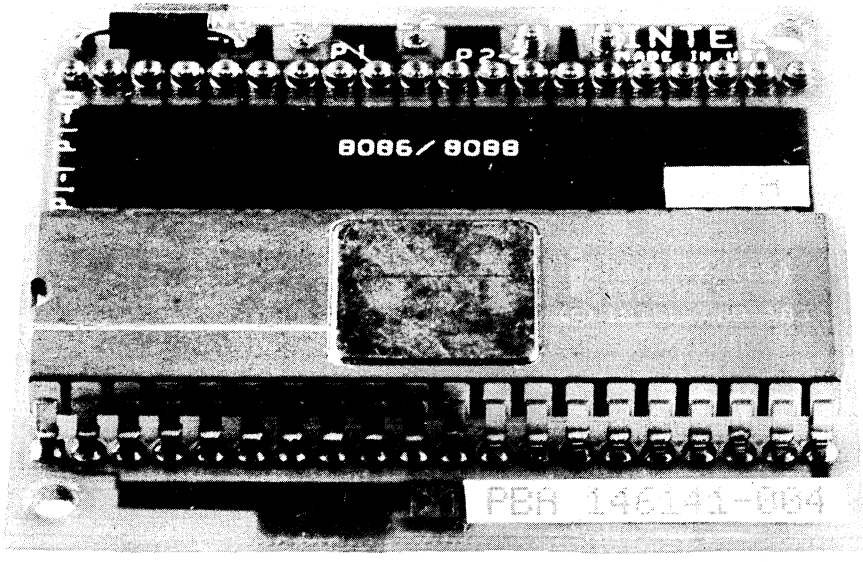
# iSBC® 337A and iSBC® 337 MULTIMODULE™ NUMERIC DATA PROCESSOR

- **High speed fixed and floating point functions for 8 or 5 MHz iSBC® 86, 88, and iAPX 86, 88 systems**
- **Extends host CPU instruction set with arithmetic, logarithmic, transcendental and trigonometric instructions**
- **MULTIMODULE™ option containing 8087 Numeric Data Processor**
- **Up to 80X performance improvement in Whetstone benchmarks over 8MHz iAPX-86/10 performance**
- **Supports seven data types including single and double precision integer and floating point**
- **Software support through ASM 86/88 Assembly Language and High Level Languages**
- **Fully supported in the multi-tasking environment of the iRMX™ 86 Operating System**

The Intel iSBC® 337A/337 MULTIMODULE™ Numeric Data Processor offers high performance numerics support for iSBC 86 and iSBC 88 Single Board Computer users, for applications including simulation, instrument automation, graphics, signal processing and business systems. The coprocessor interface between the 8087 and the host CPU provides a simple means of extending the instruction set with over 60 additional numeric instructions supporting six additional data types. The MULTIMODULE implementation allows the iSBC 337A module to be used on all iSBC 86 and iAPX 88 board designs.

The coprocessor interface between the 8087 Numeric Data Processor and the host CPU provides a simple means of extending the instruction set with over 60 additional numeric instructions supporting seven data types. The MULTIMODULE implementation allows the iSBC 337A/337 module to be used on all iSBC 86/88" single board computers and can be added as an option to custom iAPX board designs.








October, 1984
ORDER NUMBER: 280077-001

## OVERVIEW

The iSBC 337A/337 MULTIMODULE Numeric Data Processor (also called NDP) provides arithmetic and logical instruction extensions to the 86/88 of the iAPX 86/88 families. The instruction set consists of arithmetic, transcendental, logical, trigonometric and exponential instructions which can all operate on seven different data types. The data types are 16, 32, and 64 bit integer, 32 and 64 bit floating point, 18 digit packed BCD and 80 bit temporary.

### Coprocessor Interface

The coprocessor interface between the host CPU and the iSBC 337A/337 processor provides easy to use and high performance math processing. Installation of the iSBC 337A/337 processor is simply a matter of removing the host CPU from its socket, installing the iSBC 337A/337 processor into the host's CPU socket, and reinstalling the host CPU chip into the socket provided for it on the iSBC 337A/337 processor (see Figure 1).

All synchronization and timing signals are provided via the coprocessor interface with the host CPU. The two processors also share a common address/data bus. (See Figure 2). The NDP component is capable of recognizing and executing NDP numeric instructions as they are fetched by the host CPU. This interface allows concurrent processing by the host CPU and the NDP. It also allows NDP and host CPU instructions to be intermixed in any fashion to provide the maximum overlapped operation and the highest aggregate performance.

### High Performance and Accuracy

The 80-bit wide internal registers and data paths contribute significantly to high performance and minimize the execution time difference between single and double precision floating point formats. This 80-bit architecture provides very high resolution and accuracy.

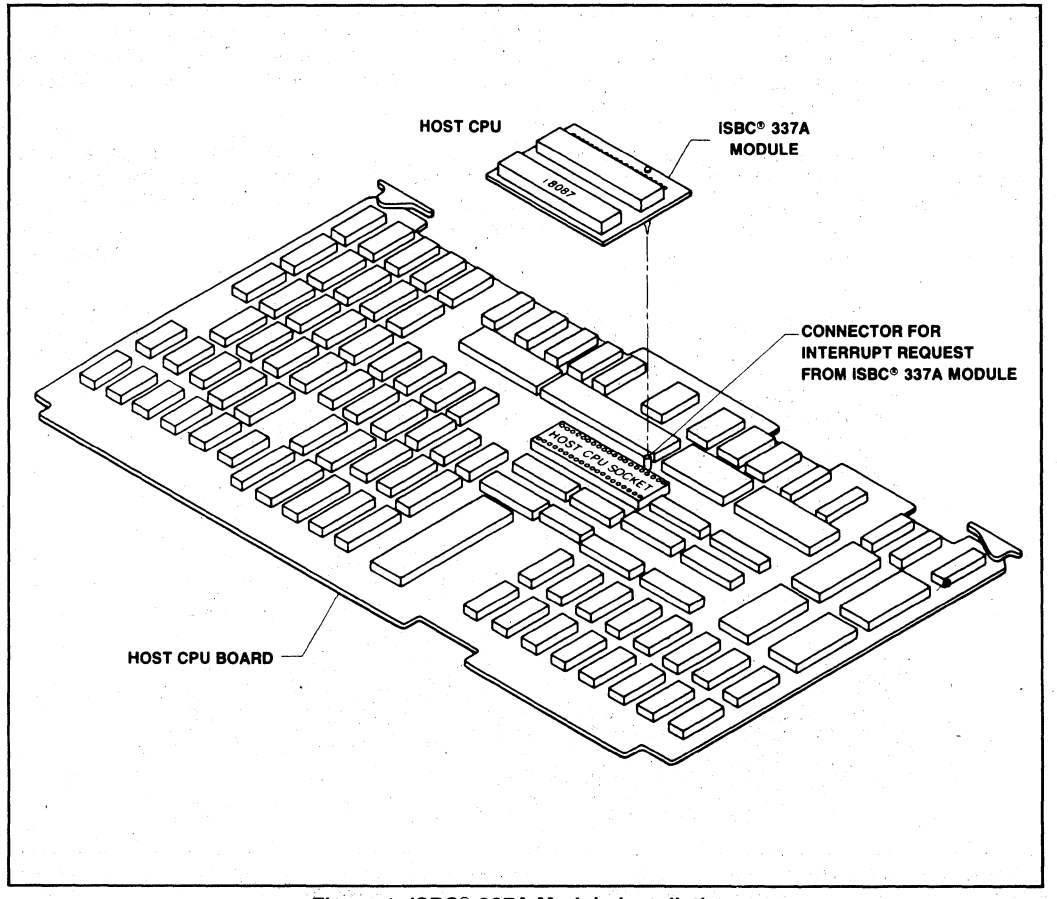



**Figure 1. iSBC® 337A Module Installation**

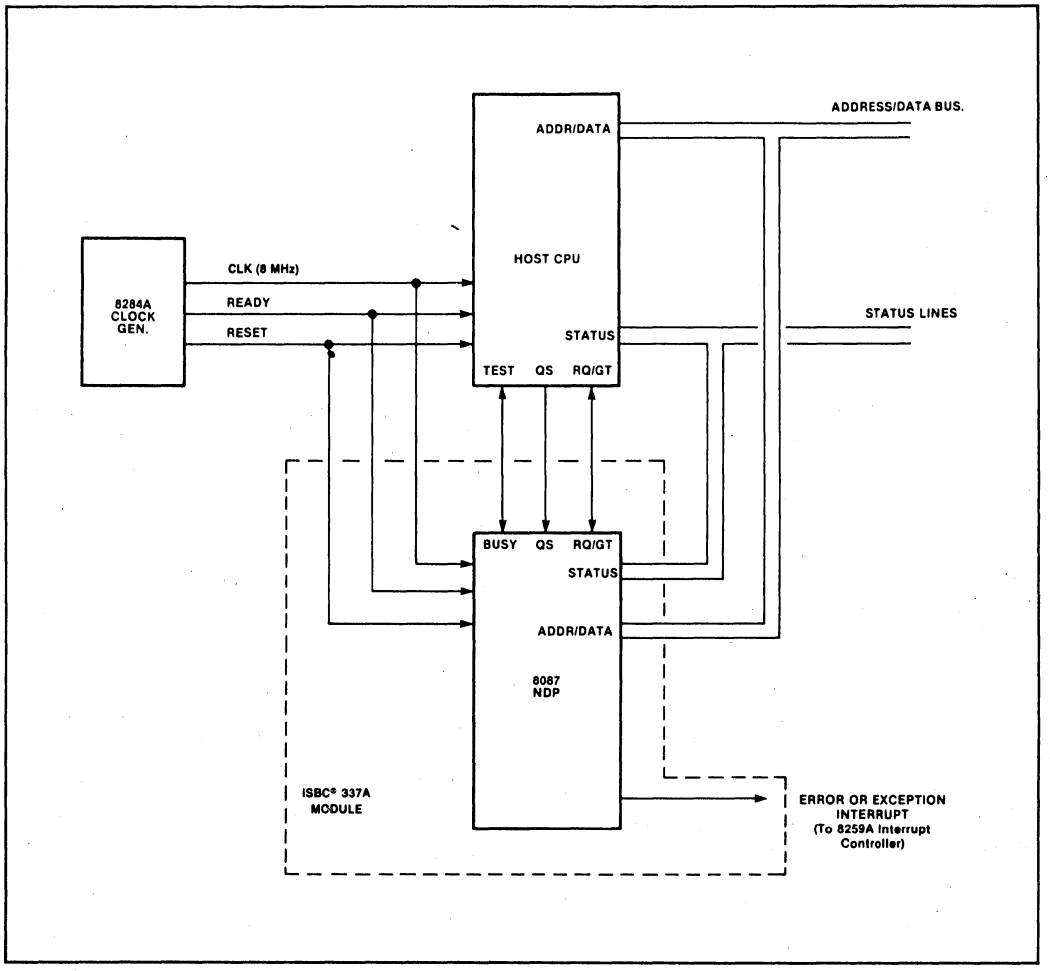



**Figure 2. iSBC® 337A System Configuration**

This precision is complemented by extensive exception detection and handling. Six different types of exceptions can be reported and handled by the NDP. The user also has control over internal precision, infinity control and rounding control.

## SYSTEM CONFIGURATION

As a coprocessor to the Host CPU, the NDP is wired in parallel with the CPU as shown in Figure 2. The CPU's status and queue status lines enable the NDP to monitor and decode instructions in synchronization with the CPU and without any CPU overhead. Once started, the NDP can process in parallel with and independent of the host CPU. For resynchronization, the NDP's BUSY signal informs the CPU that the NDP is executing an instruction and the CPU WAIT instruction tests this signal to insure that the NDP is ready to execute subsequent instructions.

The NDP can interrupt the CPU when it detects an error or exception. The interrupt request line is routed to the CPU through an 8259A Programmable Interrupt Controller. This interrupt request signal is brought down from the iSBC 337A/337 module to the single board computer through a single pin connector (see Figure 1). The signal is then routed to the interrupt matrix for jumper connection to the 8259A Interrupt Controller. Other iAPX designs may use a similar arrangement, or by masking off the CPU "READ" pin from the iSBC 337A/337 socket, provisions are made to allow the now vacated pin of the host's CPU socket to be used to bring down the interrupt request signal for connection to the base board and then to the 8259A. Another alternative is to use a wire to establish this connection.

## PROGRAMMABLE INTERFACE

Table 1 lists the seven data types the NDP supports and presents the format for each type. Internally, the NDP holds all numbers in the temporary real format. Load and store instructions automatically convert operands represented in memory as 16-, 32-, or 64-bit integers, 32- or 64-bit floating point numbers or 18-digit packed BCD numbers into temporary real format and vice versa.

Computations in the NDP use the processor's register stack. These eight 80-bit registers provide the equivalent capacity of 40 16-bit registers. The NDP register set can be accessed as a stack, with instructions operating on the top stack element, or as a fixed register set with instructions operating on explicitly designated registers.

Table 2 lists the NDP instructions by class. Assembly language programs are written in ASM 86/88, the iAPX family assembly language.

Table 3 gives the execution times of some typical numeric instructions and their equivalent time on a 8 MHz 8086-2.

## FUNCTIONAL DESCRIPTION

The NDP is internally divided into two processing elements, the control unit (CU) and the numeric execution unit (NEU), providing concurrent operation of the two units. The NEU executes all numeric instructions, while the CU receives and decodes instructions, reads and writes memory operands and executes processor control instructions.

### Control Unit

The CU keeps the NDP operating in synchronization with its host CPU. NDP instructions are intermixed with CPU instructions in a single instruction stream. The CPU fetches all instructions from memory; by monitoring the status signals emitted by the CPU, the NDP control unit determines when an 8086-2 instruction is being fetched. The CU taps the bus in parallel with the CPU and obtains that portion of the data stream.

After decoding the instruction, the host executes all opcodes but ESCAPE (ESC), while the NDP executes only the ESCAPE class instructions. (The first five bits of all ESCAPE instructions are identical). The CPU does provide addressing for ESC instructions, however.

**Table 1. 8087 Datatypes**

| Data Formats | Range | Precision | Most Significant Byte (7 07 07 07 07 07 07 07 07 07 0) |
|---|---|---|---|
| Word Integer | $10^4$ | 16 Bits | $I_{15}$ ... $I_0$ Two's Complement |
| Short Integer | $10^9$ | 32 Bits | $I_{31}$ ... $I_0$ Two's Complement |
| Long Integer | $10^{19}$ | 64 Bits | $I_{63}$ ... $I_0$ Two's Complement |
| Packed BCD | $10^{18}$ | 18 Digits | S — $D_{17}$ $D_{16}$ ... $D_1$ $D_0$ |
| Short Real | $10^{\pm 38}$ | 24 Bits | S $E_7$ $E_0$ $F_1$ ... $F_{23}$ $F_0$ Implicit |
| Long Real | $10^{\pm 308}$ | 53 Bits | S $E_{10}$ $E_0$ $F_1$ ... $F_{52}$ $F_0$ Implicit |
| Temporary Real | $10^{\pm 4932}$ | 64 Bits | S $E_{14}$ $E_0$ $F_0$ ... $F_{63}$ |

**Note:**

**Integer: I** **Sign: S**
**Fraction: F** **BCD Digit (4 Bits): D**
**Exponent: E**

**Packed BCD:** $(-1)^S(D_{17} \ldots D_0)$

**Real:** $(-1)^S(2^{E-BIAS})(F_0F_1 \ldots)$

**Bias** = 127 for Short Real
1023 for Long Real
16i383 for Temp Real

## Table 2. 8087 Instruction Set

### Data Transfer Instructions

| Real Transfers | |
|---|---|
| FLD | Load real |
| FST | Store real |
| FSTP | Store real and pop |
| FXCH | Exchange registers |
| **Integer Transfers** | |
| FILD | Integer load |
| FIST | Integer store |
| FISTP | Integer store and pop |
| **Packed Decimal Transfers** | |
| FBLD | Packed decimal (BCD) load |
| FBSTP | Packed decimal (BCD) store and pop |

### Comparison Instructions

| | |
|---|---|
| FCOM | Compare real |
| FCOMP | Compare real and pop |
| FCOMPP | Compare real and pop twice |
| FICOM | Integer compare |
| FICOMP | Integer compare and pop |
| FTST | Test |
| FXAM | Examine |

### Transcendental Instructions

| | |
|---|---|
| FPTAN | Partial tangent |
| FPATAN | Partial arctangent |
| F2XM1 | $2^X-1$ |
| FYL2X | $Y \bullet \log_2 X$ |
| FYL2XP1 | $Y \bullet \log_2 (X+1)$ |

### Arithmetic Instructions

| Addition | |
|---|---|
| FADD | Add real |
| FADDP | Add real and pop |
| FIADD | Integer add |
| **Subtraction** | |
| FSUB | Subtract real |
| FSUBP | Subtract real and pop |
| FISUB | Integer subtract |
| FSUBR | Subtract real reversed |
| FSUBRP | Subtract real reversed and pop |
| FISUBR | Integer subtract reversed |
| **Multiplication** | |
| FMUL | Multiply real |
| FMULP | Multiply real and pop |
| FIMUL | Integer multiply |
| **Division** | |
| FDIV | Divide real |
| FDIVP | Divide real and pop |
| FIDIV | Integer divide |
| FDIVR | Divide real reversed |
| FDIVRP | Divide real reversed and pop |
| FIDIVR | Integer divide reversed |
| **Other Operations** | |
| FSQRT | Square root |
| FSCALE | Scale |
| FPREM | Partial reminder |
| FRNDINT | Round to integer |
| FXTRACT | Extract exponent and significand |
| FABS | Absolute value |
| FCHS | Change sign |

### Processor Control Instructions

| | |
|---|---|
| FINIT/FNINIT | Initialize processor |
| FDISI/FNDISI | Disable interrupts |
| FENI/FNENI | Enable interrupts |
| FLDCW | Load control word |
| FSTCW/FNSTCW | Store control word |
| FSTSW/FNSTSW | Store status word |
| FCLEX/FNCLEX | Clear exceptions |
| FSTENV/FNSTENV | Store environment |
| FLDENV | Load environment |
| FSAVE/FNSAVE | Save state |
| FRSTOR | Restore state |
| FINCSTP | Increment stack pointer |
| FDECSTP | Decrement stack pointer |
| FFREE | Free register |
| FNOP | No operation |
| FWAIT | CPU wait |

## Table 3. Execution Time for Selected 8087 Actual and Emulated Instructions

| Floating Point Instruction | Approximate Execution Time (microseconds) | | |
|---|---|---|---|
| | 8087 (5 MHz Clock) | 8086 Emulation | 8087 (8 MHz Clock) |
| Add/Subtract Magnitude | 14/18 | 1,600 | 9/11 |
| Multiply (single precision) | 19 | 1,600 | 12 |
| Multiply (extended precision) | 27 | 2,100 | 17 |
| Divide | 39 | 3,200 | 24 |
| Compare | 9 | 1,300 | 6 |
| Load (double precision) | 10 | 1,700 | 6 |
| Store (double precision) | 21 | 1,200 | 13 |
| Square Root | 36 | 19,600 | 23 |
| Tangent | 90 | 13,000 | 56 |
| Exponentiation | 100 | 17,100 | 63 |

An NDP instruction either will not reference memory, will require loading one or more operands from memory into the NDP, or will require storing one or more operands from the NDP into memory. In the first case, a non-memory reference escape is used to start NDP operation. In the last two cases, the CU makes use of a "dummy read" cycle initiated by the CPU, in which the CPU calculates the operand address and initiates a bus cycle, but does not capture the data. Instead, the CPU captures and saves the address which the CPU places on the bus. If the instruction is a load, the CU additionally captures the data word when it becomes available on the local data bus. If data required is longer than one word, the CU immediately obtains the bus from the CPU using the request/grant protocol and reads the rest of the information in consecutive bus cycles. In a store operation, the CU captures and saves the store address as in a load, and ignores the data word that follows in the "dummy read" cycle. When the NDP is ready to perform the store, the CU obtains the bus from the CPU and writes the operand starting at the specified address.

## Numeric Execution Unit

The NEU executes all instructions that involve the register stack. These include arithmetic, logical, transcendental, constant and data transfer instructions. The data path in the NEU is 80 bits wide (64 fraction bits, 15 exponent bits and a sign bit) which allows internal operand transfers to be performed at very high speeds.

When the NEU begins executing an instruction, it activates the NDP BUSY signal. This signal is used in conjunction with the CPU WAIT instruction to resynchronize both processors when the NEU has completed its current instruction.

## Register Set

The NDP register set is shown in Figure 3. Each of the eight data registers in the NDP's register stack is 80 bits wide and is divided into "fields" corresponding to the NDP's temporary real data type. The register set may be addressed as a push down stack, through a top of stack pointer or any register may be addressed explicitly relative to the top of stack.

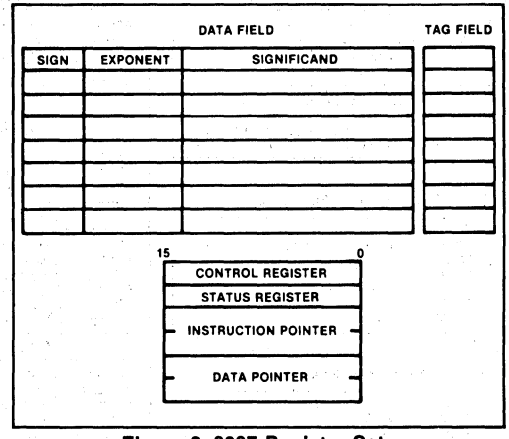



**Figure 3. 8087 Register Set**

## Status Word

The status word shown in Figure 4 reflects the overall state of the NDP; it may be stored in memory and then inspected by CPU code. The status word is a 16-bit register divided into fields as shown in Figure 4. The busy bit (bit 15) indicates whether the NEU is executing an instruction (B = 1) or is idle (B = 0). Several

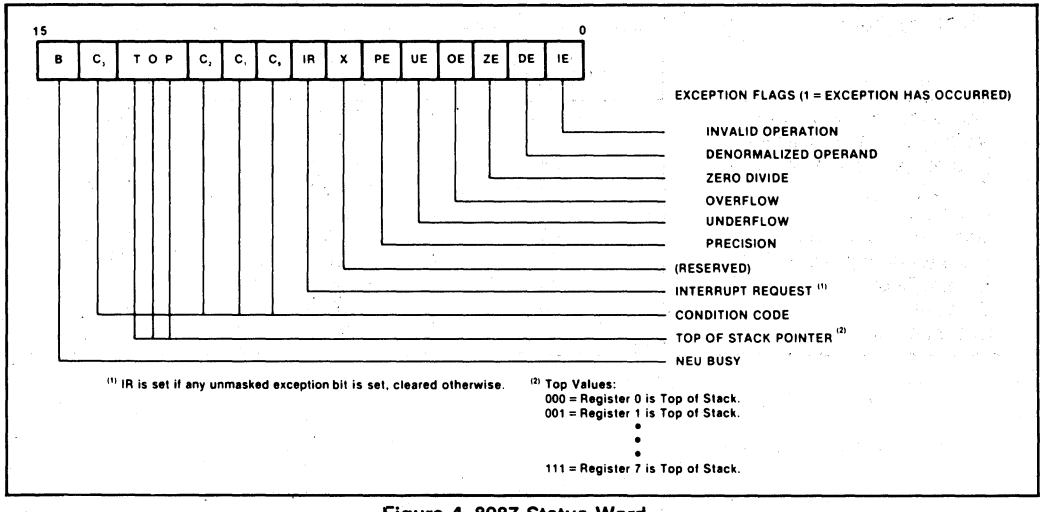



**Figure 4. 8087 Status Word**

instructions which store and manipulate the status word are executed exclusively by the CU, and these do not set the busy bit themselves.

The four numeric condition code bits ($C_0$-$C_3$) are similar to the flags in a CPU: various instructions update these bits to reflect the outcome of NDP operations.

Bits 13-11 of the status word point to the NDP register that is the current top-of-stack (TOP).

Bit 7 is the interrupt request bit. This bit is set if any unmasked exception bit is set and cleared otherwise.

Bits 5-0 are set to indicate that the NEU has detected an exception while executing an instruction.

## Tag Word

The tag word marks the content of each register as shown in Figure 5. The principal function of the tag word is to optimize the NDP's performance. The tag word can be used, however, to interpret the contents of NDP registers.

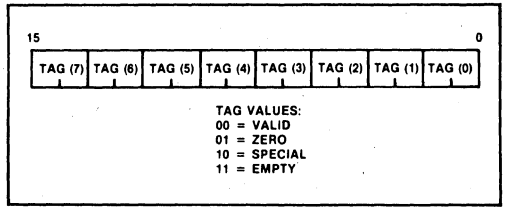



**Figure 5. 8087 Tag Word**

## Instruction and Data Pointers

The instruction and data pointers (see Figure 6) are provided for user-written error handlers. Whenever the NDP executes an NEU instruction, the CU saves the instruction address, the operand address (if present) and the instruction opcode. The NDP can then store this data in memory.

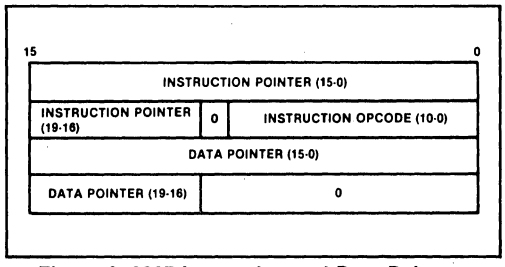



**Figure 6. 8087 Instruction and Data Pointers**

## Control Word

The NDP provides several processing options which are selected by loading a word from memory into the control word. Figure 7 shows the format and encoding of the fields in the control word.

## Exception Handling

The NDP detects six different exception conditions that can occur during instruction execution. Any or all exceptions will cause an interrupt if unmasked and interrupts are enabled.

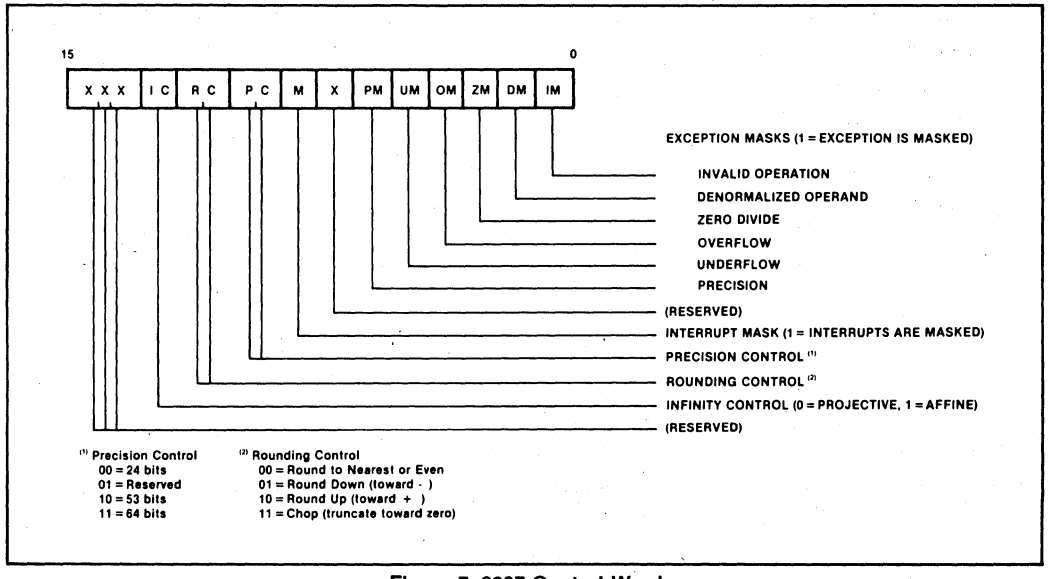



**Figure 7. 8087 Control Word**

If interrupts are disabled, the NDP will simply suspend execution until the host clears the exception. If a specific exception class is masked and that exception occurs however, the NDP will post the exception in the status register and perform an on-chip default exception handling procedure, thereby allowing processing to continue. The exceptions that the NDP detects are the following:

1. **INVALID OPERATION:** Stack overflow, stack underflow, indeterminate form (0/0, -, etc.) or the use of a Non-Number (NAN) as an operand. An exponent value is reserved and any bit pattern with this value in the exponent field is termed a Non-Number and causes this exception. If this exception is masked, the NDP default response is to generate a specific NAN called INDEFINITE, or to propagate already existing NANs as the calculation result.

2. **OVERFLOW:** The result is too large in magnitude to fit the specified format. The NDP will generate the code for infinity if this exception is masked.

3. **ZERO DIVISOR:** The divisor is zero while the dividend is a non-infinite, non-zero number. Again, the NDP will generate the code for infinity if this exception is masked.

4. **UNDERFLOW:** The result is non-zero but too small in magnitude to fit in the specified format. If this exception is masked the NDP will denormalize (shift right) the fraction until the exponent is in range. This process is called gradual underflow.

5. **DENORMALIZED OPERAND:** At least one of the operands or the result is denormalized; it has the smallest exponent but a non-zero significand. Normal processing continues if this exception is masked off.

6. **INEXACT RESULT:** If the true result is not exactly representable in the specified format, the result is rounded according to the rounding mode, and this flag is set. If this exception is masked, processing will simply continue.

## SOFTWARE SUPPORT

The iSBC 337A/337 module is supported by the following Intel software products: iRMX™ 86 Operating System, iRMX 88 Real-time Multi-tasking Executive, ASM 86/88 Assembly language, PL/M 86/88 Systems Implementation Languages, Pascal 86/88, Fortran 86/88 along with iRMX Development Utilities Package. In addition to the instructions provided in the languages to support the additional math functions, a software emulator is also available to allow the execution of iAPX instructions without the need for the iSBC 337A/337 module. This allows for the development of software in an environment without the iAPX processor and then transporting to its final run time environment with no changes in software code or mathematical results.

## SPECIFICATIONS

### Physical Characteristics

**Width** — 5.33 cm (2.100")

**Length** — 5.08 cm (2.000")

**Height** — 1.82 cm (.718")
iSBC 337A board + host board

**Weight** — 17.33 grams (.576 oz.)

### Electrical Characteristics

**DC Power Requirements**

$V_{CC} = 5V \pm 5\%$
$I_{CC} = 475$ mA max.
$I_{CC} = 350$ mA typ.

### Environmental Characteristics

**Operating Temperature** — 0°C to 55°C with 200 linear feet/minute airflow

**Relative Humidity** — Up to 90% R.H. without condensation.

### Reference Manual

**147163-001** — iSBC 337A/337 MULTIMODULE Numeric Data Processor Hardware Reference Manual (NOT SUPPLIED WITH MULTIMODULE BOARD).

Manuals may be ordered from any Intel sales representative, distributor office, or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California, 95051.

## ORDERING INFORMATION

| Part Number | Description |
|---|---|
| SBC 337A | MULTIMODULE Numeric Data Processor |
| SBC 337 | MULTIMODULE Numeric Data Processor |