



## Intel387™ SX MATH COPROCESSOR

- **New Automatic Power Management**
  - Low Power Consumption
  - Typically 100 mA in Dynamic Mode, and 4 mA in Idle Mode
- **Socket Compatible with Intel387 Family of Math CoProcessors**
  - Hardware and Software Compatible
  - Supported by Over 2100 Commercial Software Packages
  - 10% to 15% Performance Increase on Whetstone and Livermore Benchmarks
- **Compatible with the Intel386™ SX Microprocessor**
  - Extends CPU Instruction Set to Include Trigonometric, Logarithmic, and Exponential
- **High Performance 80-Bit Internal Architecture**
- **Implements ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic**
- **Available in a 68-Pin PLCC Package**

See Intel Packaging Specification, Order #231369

The Intel387™ SX Math CoProcessor is an extension to the Intel386™ SX microprocessor architecture. The combination of the Intel387™ SX with the Intel386™ SX microprocessor dramatically increases the processing speed of computer application software that utilizes high performance floating-point operations. An internal Power Management Unit enables the Intel387™ SX to perform these floating-point operations while maintaining very low power consumption for portable and desktop applications. The internal Power Management Unit effectively reduces power consumption by 95% when the device is idle.

The Intel387™ SX Math CoProcessor is available in a 68-pin PLCC package, and is manufactured on Intel's advanced 1.0 micron CHMOS IV technology.



240225-22

Intel386 and Intel387 are trademarks of Intel Corporation.

\*Other brands and names are the property of their respective owners. Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products. Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

January 1994

Order Number: 240225-009

COPYRIGHT © INTEL CORPORATION, 1995

# Intel387™ SX Math CoProcessor

| <b>CONTENTS</b>                         | PAGE | <b>CONTENTS</b>                               | PAGE |
|---|------|---|------|
| <b>1.0 PIN ASSIGNMENT</b> .....         | 5    | <b>4.0 HARDWARE SYSTEM</b>                    |      |
| 1.1 Pin Description Table .....         | 6    | <b>INTERFACE</b> .....                        | 21   |
| <b>2.0 FUNCTIONAL DESCRIPTION</b> ..... | 7    | 4.1 Signal Description .....                  | 22   |
| 2.1 Feature List .....                  | 7    | 4.1.1 Intel386 CPU Clock 2                    |      |
| 2.2 Math CoProcessor Architecture ..... | 7    | (CPUCLK2) .....                               | 22   |
| 2.3 Power Management .....              | 8    | 4.1.2 Intel387 Math CoProcessor               |      |
| 2.3.1 Dynamic Mode .....                | 8    | Clock 2 (NUMCLK2) .....                       | 22   |
| 2.3.2 Idle Mode .....                   | 8    | 4.1.3 Clocking Mode (CKM) .....               | 23   |
| 2.4 Compatibility .....                 | 8    | 4.1.4 System Reset (RESETIN) .....            | 23   |
| 2.5 Performance .....                   | 8    | 4.1.5 Processor Request                       |      |
| <b>3.0 PROGRAMMING INTERFACE</b> .....  | 9    | (PEREQ) .....                                 | 23   |
| 3.1 Instruction Set .....               | 9    | 4.1.6 Busy Status (BUSY#) .....               | 23   |
| 3.1.1 Data Transfer Instructions .....  | 9    | 4.1.7 Error Status (ERROR#) .....             | 23   |
| 3.1.2 Arithmetic Instructions .....     | 9    | 4.1.8 Data Pins (D15–D0) .....                | 23   |
| 3.1.3 Comparison Instructions .....     | 10   | 4.1.9 Write/Read Bus Cycle                    |      |
| 3.1.4 Transcendental                    |      | (W/R#) .....                                  | 23   |
| Instructions .....                      | 10   | 4.1.10 Address Strobe (ADS#) .....            | 23   |
| 3.1.5 Load Constant Instructions .....  | 10   | 4.1.11 Bus Ready Input                        |      |
| 3.1.6 Processor Instructions .....      | 11   | (READY#) .....                                | 24   |
| 3.2 Register Set .....                  | 11   | 4.1.12 Ready Output                           |      |
| 3.2.1 Status Word (SW) Register .....   | 12   | (READYO#) .....                               | 24   |
| 3.2.2 Control Word (CW)                 |      | 4.1.13 Status Enable (STEN) .....             | 24   |
| Register .....                          | 15   | 4.1.14 Math CoProcessor Select 1              |      |
| 3.2.3 Data Register .....               | 16   | (NPS1#) .....                                 | 24   |
| 3.2.4 Tag Word (TW) Register .....      | 16   | 4.1.15 Math CoProcessor Select 2              |      |
| 3.2.5 Instruction and Data              |      | (NPS2) .....                                  | 24   |
| Pointers .....                          | 16   | 4.1.16 Command (CMD0#) .....                  | 24   |
| 3.3 Data Types .....                    | 18   | 4.1.17 System Power (V <sub>CC</sub> ) .....  | 24   |
| 3.4 Interrupt Description .....         | 18   | 4.1.18 System Ground (V <sub>SS</sub> ) ..... | 24   |
| 3.5 Exception Handling .....            | 18   | 4.2 System Configuration .....                | 25   |
| 3.6 Initialization .....                | 21   | 4.3 Math CoProcessor Architecture .....       | 26   |
| 3.7 Processing Modes .....              | 21   | 4.3.1 Bus Control Logic .....                 | 26   |
| 3.8 Programming Support .....           | 21   | 4.3.2 Data Interface and Control              |      |
|   |      | Unit .....                                    | 26   |
|   |      | 4.3.3 Floating Point Unit .....               | 26   |
|   |      | 4.3.4 Power Management Unit .....             | 26   |

| <b>CONTENTS</b>  | PAGE |
|--|------|
| 4.4 Bus Cycles .....                                   | 26   |
| 4.4.1 Intel387 SX Math<br>CoProcessor Addressing ..... | 27   |
| 4.4.2 CPU/Math CoProcessor<br>Synchronization .....    | 27   |
| 4.4.3 Synchronous/Asynchronous<br>Modes .....          | 27   |
| 4.4.4 Automatic Bus Cycle<br>Termination .....         | 27   |
| <b>5.0 BUS OPERATION</b> .....                         | 27   |
| 5.1 Non-pipelined Bus Cycles .....                     | 28   |
| 5.1.1 Write Cycle .....                                | 28   |
| 5.1.2 Read Cycle .....                                 | 29   |
| 5.2 Pipelined Bus Cycles .....                         | 29   |
| 5.3 Mixed Bus Cycles .....                             | 30   |
| 5.4 BUSY# and PEREQ Timing<br>Relationship .....       | 32   |
| <b>6.0 PACKAGE SPECIFICATIONS</b> .....                | 33   |
| 6.1 Mechanical Specifications .....                    | 33   |
| 6.2 Thermal Specifications .....                       | 33   |

| <b>CONTENTS</b>   | PAGE |
|---|------|
| <b>7.0 ELECTRICAL<br/>CHARACTERISTICS</b> .....   | 33   |
| 7.1 Absolute Maximum Ratings .....  | 33   |
| 7.2 D.C. Characteristics .....  | 34   |
| 7.3 A.C. Characteristics .....  | 35   |
| <b>8.0 Intel387 SX MATH COPROCESSOR<br/>INSTRUCTION SET</b> .....                         | 41   |
| <b>APPENDIX A—Intel387 SX MATH<br/>COPROCESSOR COMPATIBILITY</b> ....                     | A-1  |
| A.1 8087/80287 Compatibility .....  | A-1  |
| A.1.1 General Differences .....   | A-1  |
| A.1.2 Exceptions .....  | A-2  |
| <b>APPENDIX B—COMPATIBILITY<br/>BETWEEN THE 80287 AND 8087<br/>MATH COPROCESSOR</b> ..... | B-1  |



## CONTENTS

### FIGURES

|   | PAGE |
|---|------|
| Figure 1-1 Intel387 SX Math CoProcessor Pinout .....  | 5    |
| Figure 2-1 Intel387 SX Math CoProcessor Block Diagram .....                                   | 7    |
| Figure 3-1 Intel 386 SX CPU and Intel387 Math CoProcessor Register Set .....                  | 11   |
| Figure 3-2 Status Word .....  | 12   |
| Figure 3-3 Control Word .....   | 15   |
| Figure 3-4 Tag Word Register .....  | 16   |
| Figure 3-5 Instruction and Data Pointer Image in Memory, 32-Bit Protected Mode Format .....   | 17   |
| Figure 3-6 Instruction and Data Pointer Image in Memory, 16-Bit Protected Mode Format .....   | 17   |
| Figure 3-7 Instruction and Data Pointer Image in Memory, 32-Bit Real Mode Format .....        | 17   |
| Figure 3-8 Instruction and Data Pointer Image in Memory, 16-Bit Real Mode Format .....        | 18   |
| Figure 4-1 Intel386 SX CPU and Intel387 SX Math CoProcessor System Configuration .....        | 25   |
| Figure 5-1 Bus State Diagram .....  | 28   |
| Figure 5-2 Non-Pipelined Read and Write Cycles .....  | 29   |
| Figure 5-3 Fastest Transition to and from Pipelined Cycles .....                              | 30   |
| Figure 5-4 Pipelined Cycles with Wait States .....  | 31   |
| Figure 5-5 BUSY # and PEREQ Timing Relationship .....   | 32   |
| Figure 7-1a Typical Output Valid Delay vs Load Capacitance at Max Operating Temperature ..... | 37   |
| Figure 7-1b Typical Output Slew Time vs Load Capacitance at Max Operating Temperature .....   | 37   |
| Figure 7-1c Maximum I <sub>CC</sub> vs Frequency .....  | 37   |

## CONTENTS

|   | PAGE |
|---|------|
| Figure 7-2 CPUCLK2/NUMCLK2 Waveform and Measurement Points for Input/Output ..... | 38   |
| Figure 7-3 Output Signals .....   | 38   |
| Figure 7-4 Input and I/O Signals .....  | 39   |
| Figure 7-5 RESET Signal .....   | 39   |
| Figure 7-6 Float from STEN .....  | 40   |
| Figure 7-7 Other Parameters .....   | 40   |

### TABLES

|   |    |
|---|----|
| Table 1-1 Pin Cross Reference—Functional Grouping .....                           | 5  |
| Table 3-1 Condition Code Interpretation .....                                     | 13 |
| Table 3-2 Condition Code Interpretation after FPREM and FPREM1 Instructions ..... | 14 |
| Table 3-3 Condition Code Resulting from Comparison .....                          | 14 |
| Table 3-4 Condition Code Defining Operand Class .....                             | 14 |
| Table 3-5 Mapping Condition Codes to Intel386 CPU Flag Bits .....                 | 14 |
| Table 3-6 Intel387 SX Math CoProcessor Data Type Representation in Memory .....   | 19 |
| Table 3-7 CPU Interrupt Vectors Reserve for Math CoProcessor .....                | 20 |
| Table 3-8 Intel387 SX Math CoProcessor Exceptions .....                           | 20 |
| Table 4-1 Pin Summary .....   | 22 |
| Table 4-2 Output Pin Status during Reset .....                                    | 23 |
| Table 4-3 Bus Cycle Definition .....  | 26 |
| Table 6-1 Thermal Resistances (°C/Watt) $\theta_{JC}$ and $\theta_{JA}$ .....     | 33 |
| Table 6-2 Maximum T <sub>A</sub> at Various Airflows .....                        | 33 |
| Table 7-1 D.C. Specifications .....   | 34 |
| Table 7-2a Timing Requirements of the Bus Interface Unit .....                    | 35 |
| Table 7-2b Timing Requirements of the Execution Unit .....                        | 36 |
| Table 7-2c Other AC Parameters .....  | 36 |
| Table 8-1 Instruction Formats .....   | 41 |



## Intel387™ SX MATH COPROCESSOR

### 1.0 PIN ASSIGNMENT

The Intel387 SX Math CoProcessor pinout as viewed from the top side of the component is shown in Figure 1-1. V<sub>CC</sub> and V<sub>SS</sub> (GND) connections must be made to multiple pins. The circuit board should

include V<sub>CC</sub> and V<sub>SS</sub> planes for power distribution and all V<sub>CC</sub> and V<sub>SS</sub> pins must be connected to the appropriate plane.

**NOTE:**

Pins identified as N.C. should remain completely unconnected.

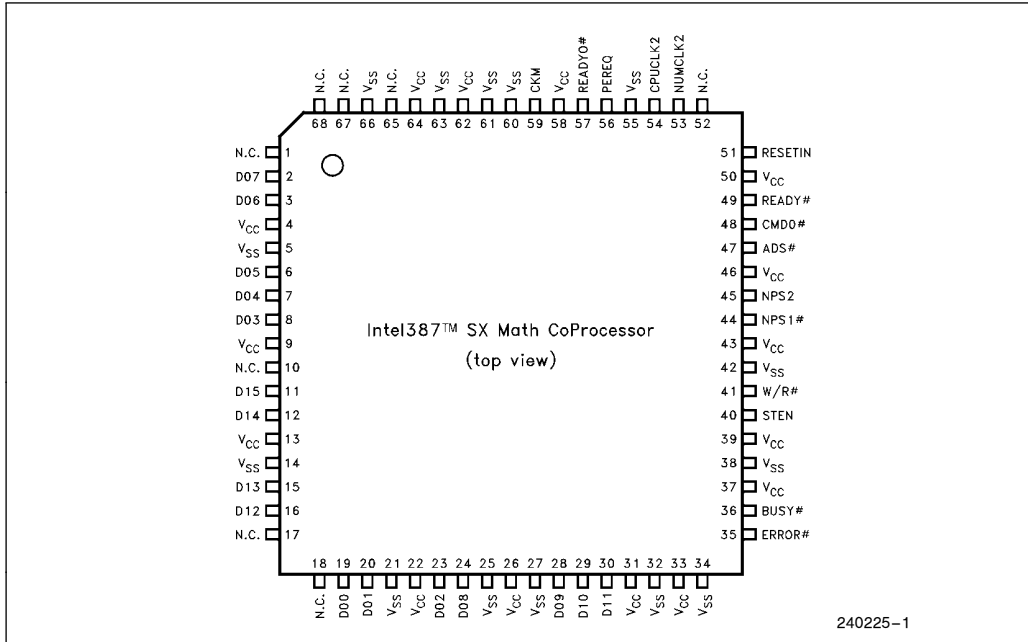


Figure 1-1. Intel387™ SX Math CoProcessor Pinout

Table 1-1. Pin Cross Reference—Functional Grouping

|          |    |     |    |                 |    |                 |    |      |    |
|----------|----|-----|----|-----------------|----|-----------------|----|------|----|
| BUSY #   | 36 | D00 | 19 | V <sub>CC</sub> | 4  | V <sub>SS</sub> | 5  | N.C. | 1  |
| PEREQ    | 56 | D01 | 20 |                 | 9  |                 | 14 |      | 10 |
| ERROR #  | 35 | D02 | 23 |                 | 13 |                 | 21 |      | 17 |
| ADS #    | 47 | D03 | 8  |                 | 22 |                 | 25 |      | 18 |
| CMD0 #   | 48 | D04 | 7  |                 | 26 |                 | 27 |      | 52 |
| NPS1 #   | 44 | D05 | 6  |                 | 31 |                 | 32 |      | 65 |
| NPS2     | 45 | D06 | 3  |                 | 33 |                 | 34 |      | 67 |
| STEN     | 40 | D07 | 2  |                 | 37 |                 | 38 |      | 68 |
| W/R #    | 41 | D08 | 24 |                 | 39 |                 | 42 |      |    |
| READY #  | 49 | D09 | 28 |                 | 43 |                 | 55 |      |    |
| READYO # | 57 | D10 | 29 |                 | 46 |                 | 60 |      |    |
|          |    | D11 | 30 |                 | 50 |                 | 61 |      |    |
|          |    | D12 | 16 |                 | 58 |                 | 63 |      |    |
|          |    | D13 | 15 |                 | 62 |                 | 66 |      |    |
| CKM      | 59 | D14 | 12 |                 | 64 |                 |    |      |    |
| CPUCLK2  | 54 | D15 | 11 |                 |    |                 |    |      |    |
| NUMCLK2  | 53 |     |    |                 |    |                 |    |      |    |
| RESETIN  | 51 |     |    |                 |    |                 |    |      |    |

## 1.1 Pin Description Table

The following table lists a brief description of each pin on the Intel387 SX Math CoProcessor. For a more complete description refer to Section 4.1 Signal Description. The following definitions are used in these descriptions:

- # The signal is active LOW.
- I Input Signal
- O Output Signal
- I/O Input and Output Signal

| Symbol          | Type | Name and Function   |
|-----------------|------|---|
| ADS#            | I    | <b>ADDRESS STROBE</b> indicates that the address and bus cycle definition is valid.   |
| BUSY#           | O    | <b>BUSY</b> indicates that the Math CoProcessor is currently executing an instruction.  |
| CKM             | I    | <b>CLOCKING MODE</b> is used to select synchronous or asynchronous clock modes.   |
| CMD0            | I    | <b>COMMAND</b> determines whether an opcode or operand are being sent to the Math CoProcessor. During a read cycle it indicates which register group is being read.     |
| CPUCLK2         | I    | <b>CPU CLOCK</b> input provides the timing for the bus interface unit and the execution unit in synchronous mode.   |
| D15–D0          | I/O  | <b>DATA BUS</b> is used to transfer instructions and data between the Math CoProcessor and CPU.   |
| ERROR#          | O    | <b>ERROR</b> signals that an unmasked exception has occurred.   |
| NC              | —    | <b>NO CONNECT</b> should always remain unconnected. Connection of a N.C. pin may cause the Math CoProcessor to malfunction or be incompatible with future steppings.    |
| NPS1#           | I    | <b>NPX SELECT 1</b> is used to select the Math CoProcessor.   |
| NPS2            | I    | <b>NPX SELECT 2</b> is used to select the Math CoProcessor.   |
| NUMCLK2         | I    | <b>NUMERICS CLOCK</b> is used in asynchronous mode to drive the Floating Point Execution Unit.  |
| PEREQ           | O    | <b>PROCESSOR EXTENSION REQUEST</b> signals the CPU that the Math CoProcessor is ready for data transfer to/from its FIFO.   |
| READY#          | I    | <b>READY</b> indicates that the bus cycle is being terminated.  |
| READYO#         | O    | <b>READY OUT</b> signals the CPU that the Math CoProcessor is terminating the bus cycle.  |
| RESETIN         | I    | <b>SYSTEM RESET</b> terminates any operation in progress and forces the Math CoProcessor to enter a dormant state.  |
| STEN            | I    | <b>STATUS ENABLE</b> serves as a master chip select for the Math CoProcessor. When inactive, this pin forces all outputs and bi-directional pins into a floating state. |
| W/R#            | I    | <b>WRITE/READ</b> indicates whether the CPU bus cycle in progress is a read or a write cycle.   |
| V <sub>CC</sub> | I    | <b>SYSTEM POWER</b> provides the +5V nominal D.C. supply input.   |
| V <sub>SS</sub> | I    | <b>SYSTEM GROUND</b> provides the 0V connection from which all inputs and outputs are measured.   |

## 2.0 FUNCTIONAL DESCRIPTION

The Intel387 SX Math CoProcessor is designed to support the Intel386 SX Microprocessor and effectively extend the CPU architecture by providing fast execution of arithmetic instructions and transcendental functions. This component contains internal power management circuitry for reduced active power dissipation and an automatic idle mode.

## 2.1 Feature List

- New power saving design provides low power dissipation in active and idle modes.
- Higher Performance, 10%–25% higher benchmark performance than the original Intel387 SX Math CoProcessor.
- High Performance 84-bit Internal Architecture
- Eight 80-bit Numeric Registers, usable as individually addressable general registers or as a register stack.
- Full-range transcendental operations for SINE, COSINE, TANGENT, ARCTANGENT, and LOG-ARITHM.
- Programmable rounding modes and notification of rounding effects.
- Exception reporting either by software polling or hardware interrupts.
- Fully compatible with the SX Microprocessors.

- Expands Intel386 SX CPU data types to include 32-bit, 64-bit, and 80-bit Floating Point; 32-bit and 64-bit Integers; and 18 Digit BCD Operands.
- Directly extends the Intel386 SX CPU Instruction Set to trigonometric, logarithmic, exponential, and arithmetic functions for all data types.
- Operates independently of Real, Protected, and Virtual-86 Modes of the Intel386 SX Microprocessors.
- Fully compatible with the Intel387 SL Mobile and DX Math CoProcessors. Implements all Intel387 Math CoProcessor architectural enhancements over 8087 and 80287.
- Implements ANSI/IEEE Standard 754-1985 for binary floating point arithmetic.
- Upward Object Code compatible from 8087 and 80287.

## 2.2 Math CoProcessor Architecture

As shown in Figure 2-1, the Intel387 SX Math CoProcessor is internally divided into four sections; the Bus Control Logic, the Data Interface and Control Unit, the Floating Point Unit, and the Power Management Unit. The Bus Control Logic is responsible for the CPU bus tracking and interface. The Data Interface and Control Unit latches data and decodes instructions. The Floating Point Unit executes the mathematical instructions. The Power Management Unit is new to the Intel387 family and is the nucleus

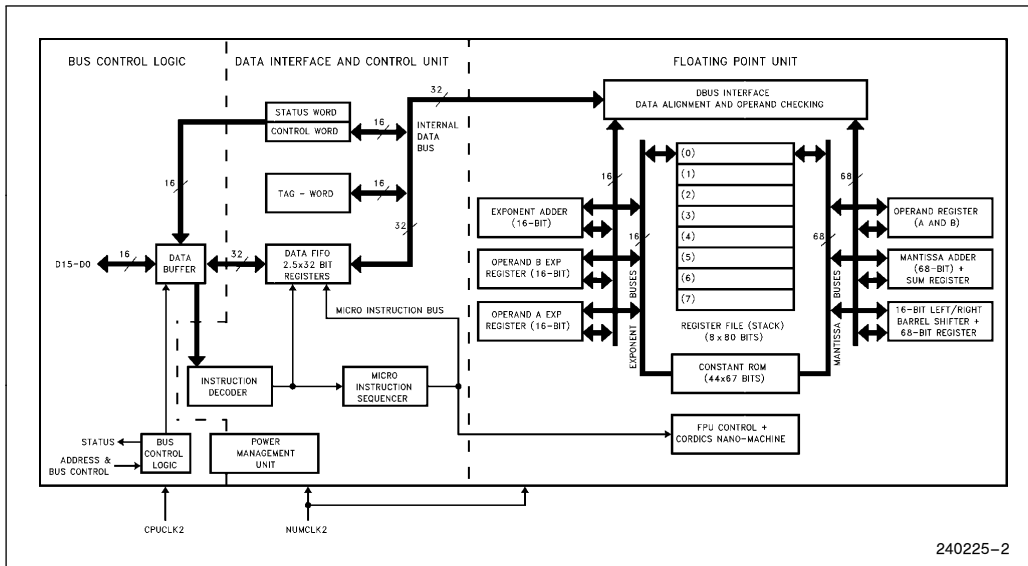


Figure 2-1. Intel387™ SX Math CoProcessor Block Diagram



of the static architecture. It is responsible for shutting down idle sections of the device to save power.

#### Microprocessor/Math CoProcessor Interface

The Intel386 CPU interprets the pattern 11011B in most significant five bits of an instruction as an opcode intended for a math coprocessor. Instructions thus marked are called ESCAPE or ESC instructions. Upon decoding the instruction as an ESC instruction, the Intel386 CPU transfers the opcode to the math coprocessor through an I/O write cycle at a dedicated address (8000F8H) outside the normal programmed I/O address range. The math coprocessor has dedicated output signals for controlling the data transfer and notifying the CPU if the Math CoProcessor is busy or that a floating point error has occurred.

### 2.3 Power Management

The Intel387 SX Math CoProcessor offers two modes of power management; dynamic and idle.

#### 2.3.1 DYNAMIC MODE

**Dynamic Mode** is when the device is executing an instruction. Using Intel's CHMOS IV technology, the Intel387 SX Math CoProcessor draws considerably less power than its predecessor. The active power supply current is reduced to approximately 100 mA at 20 MHz and provides low case temperatures.

#### 2.3.2 IDLE MODE

When an instruction is not being executed, the Intel387 SX Math CoProcessor will automatically change to **Idle Mode**. Three clocks after completion of the previous instruction, the internal power manager shuts down the floating point execution unit and all non-essential circuitry. Only portions of the Bus Interface Unit remain active to monitor the CPU bus activity and to accept the next instruction when it is transferred. When the CPU transfers the next instruction to the Math CoProcessor, the Intel387 SX

Math CoProcessor accepts the instruction and ramps the internal core within one clock so there is no impact to performance or throughput. In idle mode, the Intel387 SX Math CoProcessor draws typically 4 mA of current and reduces case temperature to near ambient.

#### NOTE:

In asynchronous clock mode (CKM = 0), the internal idle mode is disabled.

### 2.4 Compatibility

The Intel387 SX Math CoProcessor is compatible with the Intel387 SL Mobile Math CoProcessor. Due to the increased performance and internal pipelining effects, diagnostic programs should never use instruction execution time for test purposes.

### 2.5 Performance

The increased performance of floating point calculations can be attributed to the 84-bit architecture and floating point processor. For the CPU to execute floating point calculations requires very long software emulation methods with reduced resolution and accuracy. The performance of the Intel387 SX Math CoProcessor has been further enhanced through improvements in the internal microcode and through internal architectural changes. These refinements will increase Whetstone benchmarks by approximately 10% to 25% over the original Intel387 SX Math CoProcessor.

Real performance, however, should be measured with application software. Depending upon software coding, system overhead, and percentage of floating point instructions, performance can vary significantly.





### 3.0 PROGRAMMING INTERFACE

The Intel387 SX Math CoProcessor effectively extends to an Intel386 Microprocessor system additional instructions, registers, data types, and interrupts specifically designed to facilitate high-speed floating point processing. All communication between the CPU and the Math CoProcessor is transparent to applications software. The CPU automatically controls the Math CoProcessor whenever a numerics instruction is executed. All physical memory and virtual memory of the CPU are available for storage of the instructions and operands of programs that use the Math CoProcessor. All memory addressing modes, including use of displacement, base register, index register, and scaling are available for addressing numerical operands.

The Intel387 SX Math CoProcessor is software compatible with the Intel387 DX Math CoProcessors and supports all applications written for the Intel386 CPU and Intel387 Math CoProcessors.

### 3.1 Instruction Set

The Intel386 CPU interprets the pattern 11011B in most significant five bits of an instruction as an opcode intended for a math coprocessor. Instructions thus marked are called ESCAPE or ESC instruction.

The typical Math CoProcessor instruction accepts one or two operands and produces one or sometimes two results. In two-operand instructions, one operand is the contents of the Math CoProcessor register, while the other may be a memory location. The operands of some instructions are predefined; for example, FSQRT always takes the square root of the number in the top stack element.

The Intel387 SX Math CoProcessor instruction set can be divided into six groups. The following sections give a brief description of each instruction. Section 8.0 defines the instruction format and byte fields. Further details can be obtained from the Intel387 User's Manual, Programmer's Reference, Order #231917.

#### 3.1.1 DATA TRANSFER INSTRUCTIONS

The class includes the operations that load, store, and convert operands of any support data types.

##### Real Transfers

- FLD Load Real (single, double, extended)
- FST Store Real (single, double)
- FSTP Store Real and pop (single, double, extended)
- FXCH Exchange registers

##### Integer Transfers

- FILD Load (convert from) Integer (word, short, long)
- FIST Store (convert to) Integer (word, short)
- FISTP Store (convert to) Integer and pop (word, short, long)

##### Packed Decimal Transfers

- FBLD Load (convert from) packed decimal
- FBSTP Store packed decimal and pop

#### 3.1.2 ARITHMETIC INSTRUCTIONS

This class of instructions provide variations on the basic add, subtract, multiply, and divide operations and a number of other basic arithmetic operations. Operands may reside in registers or one operand may reside in memory.

##### Addition

- FADD Add Real
- FADDP Add Real and pop
- FIADD Add Integer

##### Subtraction

- FSUB Subtract Real
- FSUBP Subtract Real and pop
- FISUB Subtract Integer
- FSUBR Subtract Real reversed
- FSUBRP Subtract Real reversed and pop
- FISUBR Subtract Integer reversed

##### Multiplication

- FMUL Multiply Real
- FMULP Multiply Real and pop
- FIMUL Multiply Integer

##### Division

- FDIV Divide Real
- FDIVP Divide Real and pop
- FIDIV Divide Integer
- FDIVR Divide Real reversed
- FDIVRP Divide Real reversed and pop
- FIDIVR Divide Integer reversed

## Other Operations

|         |                                  |
|---------|----------------------------------|
| FSQRT   | Square Root                      |
| FSCALE  | Scale                            |
| FPREM   | Partial Remainder                |
| FPREM1  | IEEE standard partial remainder  |
| FRNDINT | Round to Integer                 |
| FXTRACT | Extract Exponent and Significand |
| FABS    | Absolute Value                   |
| FCHS    | Change sign                      |

**3.1.3 COMPARISON INSTRUCTION**

Instructions of this class allow comparison of numbers of all supported real and integer data types. Each of these instructions analyzes the top stack element often in relationship to another operand and reports the result as a condition code in the status word.

|         |                                      |
|---------|--------------------------------------|
| FCOM    | Compare Real                         |
| FCOMP   | Compare Real and pop                 |
| FCOMPP  | Compare Real and pop twice           |
| FUCOM   | Unordered compare Real               |
| FUCOMP  | Unordered compare Real and pop       |
| FUCOMPP | Unordered compare Real and pop twice |
| FICOM   | Compare Integer                      |
| FICOMP  | Compare Integer and pop              |
| FTST    | Test                                 |
| FXAM    | Examine                              |

**3.1.4 TRANSCENDENTAL INSTRUCTIONS**

This group of the Intel387 operations includes trigonometric, inverse trigonometric, logarithmic and exponential functions. The transcendental operate on the top one or two stack elements, and they return their results to the stack. The trigonometric operations assume their arguments are expressed in radians. The logarithmic and exponential operations work in base 2.

|         |                        |
|---------|------------------------|
| FSIN    | Sine                   |
| FCOS    | Cosine                 |
| FSINCOS | Sine and cosine        |
| FPTAN   | Tangent                |
| FPATAN  | Arctangent of ST(1)/ST |
| F2XM1   | $2^x - 1$              |
| FYL2X   | $Y * \log_2 X$         |
| FYL2XP1 | $Y * \log_2(X + 1)$    |

**3.1.5 LOAD CONSTANT INSTRUCTIONS**

Each of these instructions loads (pushes) a commonly used constant onto the stack. The constants have extended real values nearest to the infinitely precise numbers. The only error that can be generated is an Invalid Exception if a stack overflow occurs.

|        |                    |
|--------|--------------------|
| FLDZ   | Load +0.0          |
| FLD1   | Load +1.0          |
| FLDPI  | Load $\pi$         |
| FLDL2T | Load $\log_2 10$   |
| FLDL2E | Load $\log_2 e$    |
| FLDLG2 | Load $\log_{10} 2$ |
| FLDLN2 | Load $\log_e 2$    |



**3.1.6 PROCESSOR INSTRUCTIONS  
(ADMINISTRATIVE)**

- FINIT Initialize Math CoProcessor
- FLDCW Load Control Word
- FSTCW Store Control Word
- FLDCW Load Status Word
- FSTSW Store Status Word
- FSTSW AX Store Status Word to AX register
- FCLEX Clear Exceptions
- FSTENV Store Environment
- FLDENV Load Environment
- FSAVE Save State

- FRSTOR Restore State
- FINCSTP Increment Stack pointer
- FDECSTP Decrement Stack pointer
- FFREE Free Register
- FNOP No Operation
- FWAIT Report Math CoProcessor Error

**3.2 Register Set**

Figure 3-1 shows the Intel387 SX Math CoProcessor register set. When a Math CoProcessor is present in a system, programmers may use these registers in addition to the registers normally available on the CPU.

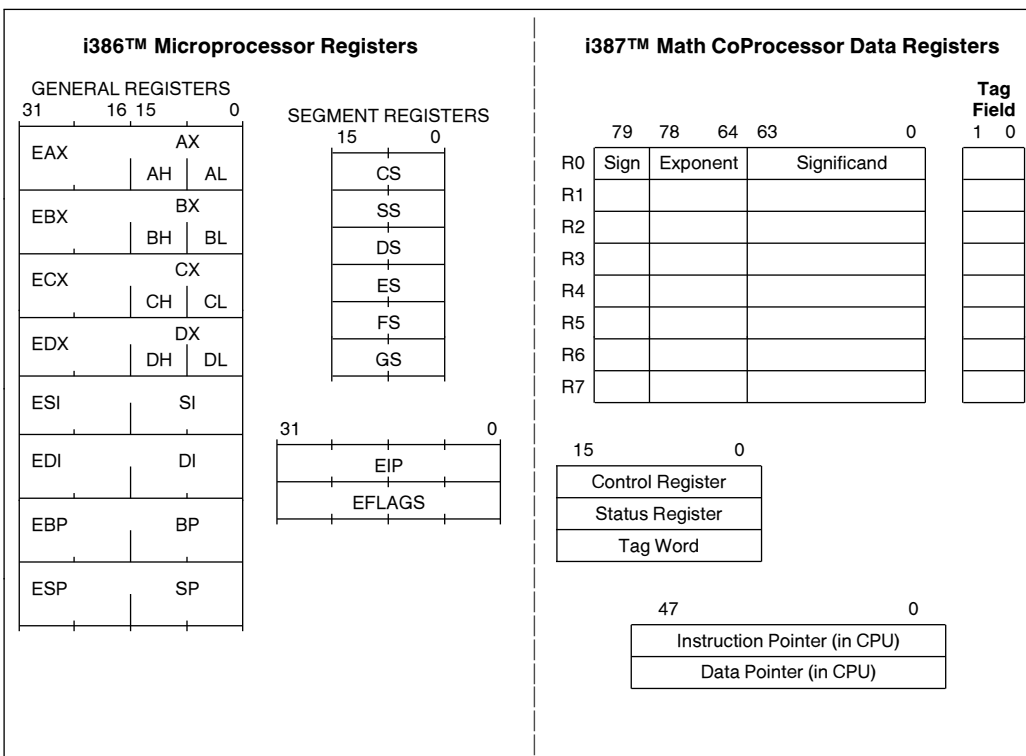


Figure 3-1. Intel386™ CPU and Intel387™ Math CoProcessor Register Set



**3.2.1 STATUS WORD (SW) REGISTER**

The 16-bit status word (in the status register) shown in Figure 3-2 reflects the overall state of the Math CoProcessor. It can be read and inspected by programs using the FSTSW memory or FSTSW AX instructions.

Bit 15, the Busy bit (B) is included for 8087 compatibility only. It always has the same value as the Error Summary bit (ES, bit 7 of status word); it does not indicate the status of the BUSY# output of the Math CoProcessor.

Bits 13–11 (TOP) serves as the pointer to the Math CoProcessor data register that is the current Top-Of-Stack. The significance of the stack top is described in Section 3.2.5 Data Registers.

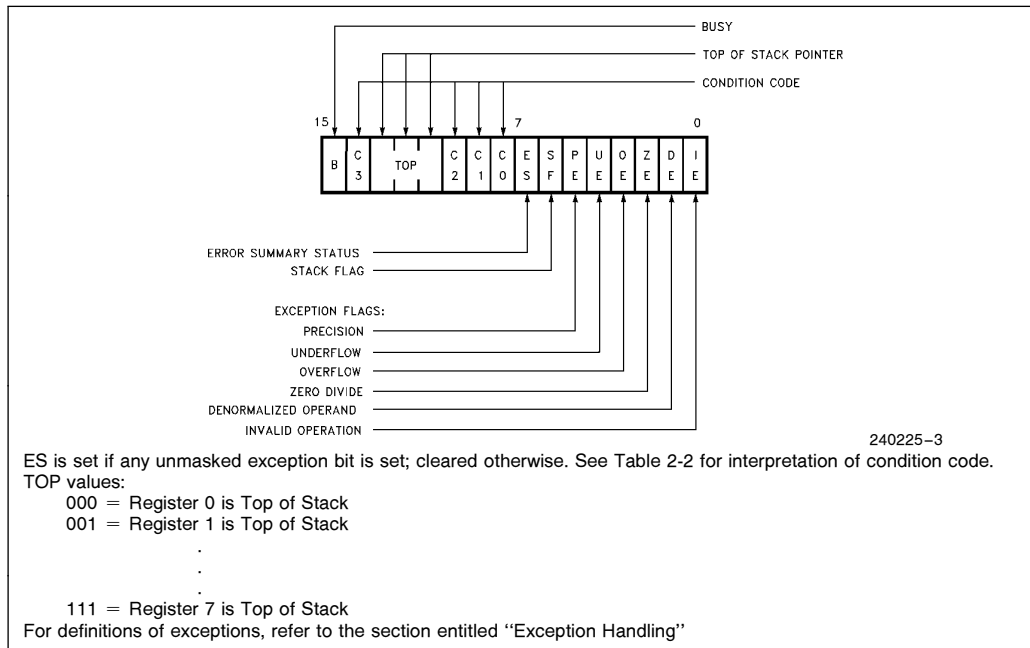
The four numeric condition code bits (C<sub>3</sub>–C<sub>0</sub>, Bit 14, 10–8) are similar to the flags in a CPU; instructions that perform arithmetic operations update these bits to reflect the outcome. The effects of the instructions on the condition code are summarized in Tables 3-1 through 3-4. These condition code bits are used principally for conditional branching. The FSTSW AX instructions stores the Math CoProcessor status word directly to the CPU AX register, allowing the condition codes to be inspected efficiently by Intel386 CPU code. The Intel386 CPU SAHF instruction can copy C<sub>3</sub>–C<sub>0</sub> directly to the flag bits to simplify conditional branching. Table 3-5 shows the mapping of these bits to the Intel386 CPU flag bits.

Bit 7 is the error summary (ES) status bit. This bit is set if any unmasked exception bit is set; it is clear otherwise. If this bit is set, the ERROR# signal is asserted.

Bit 6 is the stack flag (SF). This bit is used to distinguish invalid operations due to stack overflow or underflow from other kinds of invalid operations. When SF is set, bit 9 (C<sub>1</sub>) distinguishes between stack overflow (C<sub>1</sub> = 1) or underflow (C<sub>1</sub> = 0).

Bit 5–0 are the six exception flags of the status word and are set to indicate that during an instruction execution the Math CoProcessor has detected one of six possible exception conditions since these status bits were last cleared or reset. Section 3.5 entitled Exception Handling explains how they are set and used.

The exception flags are “sticky” bits and can only be cleared by the instructions FINIT, FCLEX, FLDENV, FSAVE, and FRSTOR. Note that when a new value is loaded into the status word by the FLDENV or FRSTOR instruction, the value of ES (bit 7) and B (bit 15) are not derived from the values loaded from memory but rather are dependent upon the values of the exception flags (bits 5–0) in the status word and their corresponding masks in the control word. If ES is set in such a case, the ERROR# output of the Math CoProcessor is activated immediately.



**Figure 3-2. Status Word**





**Table 3-2. Condition Code Interpretation after FPREM and FPREM1 Instructions**

| Condition Code |    |    |    | Interpretation after FPREM and FPREM1   |   |
|----------------|----|----|----|---|---|
| C2             | C3 | C1 | C0 |   |   |
| 1              | X  | X  | X  | Incomplete Reduction:<br>further iteration required<br>for complete reduction |   |
| 0              | Q1 | Q0 | Q2 | Q MOD8  | Complete Reduction:<br>C0, C3, C1 contain three least<br>significant bits of quotient |
|                | 0  | 0  | 0  | 0   |   |
|                | 0  | 1  | 0  | 1   |   |
|                | 1  | 0  | 0  | 2   |   |
|                | 1  | 1  | 0  | 3   |   |
|                | 0  | 0  | 1  | 4   |   |
|                | 0  | 1  | 1  | 5   |   |
|                | 1  | 0  | 1  | 6   |   |
| 1              | 1  | 1  | 7  |   |   |

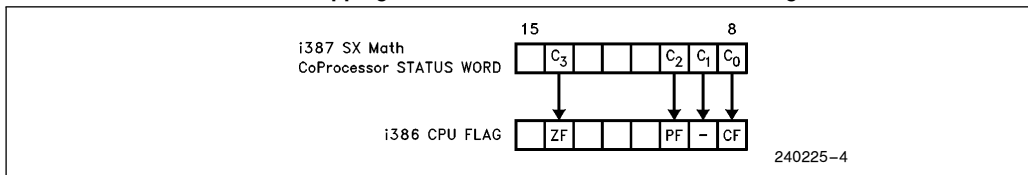
**Table 3-3. Condition Code Resulting from Comparison**

| Order         | C3 | C2 | C0 |
|---------------|----|----|----|
| TOP > Operand | 0  | 0  | 0  |
| TOP < Operand | 0  | 0  | 1  |
| TOP = Operand | 1  | 0  | 0  |
| Unordered     | 1  | 1  | 1  |

**Table 3-4. Condition Code Defining Operand Class**

| C3 | C2 | C1 | C0 | Value at TOP  |
|----|----|----|----|---------------|
| 0  | 0  | 0  | 0  | + Unsupported |
| 0  | 0  | 0  | 1  | + NaN         |
| 0  | 0  | 1  | 0  | - Unsupported |
| 0  | 0  | 1  | 1  | - NaN         |
| 0  | 1  | 0  | 0  | + Normal      |
| 0  | 1  | 0  | 1  | + Infinity    |
| 0  | 1  | 1  | 0  | - Normal      |
| 0  | 1  | 1  | 1  | - Infinity    |
| 1  | 0  | 0  | 0  | + 0           |
| 1  | 0  | 0  | 1  | + Empty       |
| 1  | 0  | 1  | 0  | - 0           |
| 1  | 0  | 1  | 1  | - Empty       |
| 1  | 1  | 0  | 0  | + Denormal    |
| 1  | 1  | 1  | 0  | - Denormal    |

**Table 3-5 Mapping Condition Codes to Intel386™ CPU Flag Bits**



### 3.2.2 CONTROL WORD (CW) REGISTER

The Math CoProcessor provides the programmer with several processing options that are selected by loading a control word from memory into the control register. Figure 3-3 show the format and encoding of fields in the control word.

The low-order byte of the control word register is used to configure the exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the Math CoProcessor recognizes. See Section 3.5, Exception Handling, for further explanation on the exception control and definition.

The high-order byte of the control word is used to configure the Math CoProcessor operating mode, including precision, rounding and infinity control.

- The rounding control (RC) field (bits 11–10) provide for directed rounding and true chop, as well as the unbiased round to nearest even mode specified in the IEEE standard. Rounding control affects only those instructions that perform rounding at the end of the operation (and thus can generate a precision exception); namely, FST, FSTP, FIST, all arithmetic instructions (except FPREM, FPREM1, FEXTRACT, FABS, and FCHS) and all transcendental instructions.

- The precision control (PC) field (bits 9–8) can be used to set the Math CoProcessor internal operating precision of the significand at less than the default of 64 bits (extended precision). This can be useful in providing compatibility with early generation arithmetic processors of smaller precision. PC affects only the instructions FADD, FSUB(R), FMUL, FDIV(R), and FSQRT. For all other instructions, either the precision is determined by the opcode or extended precision is used.
- The “infinity control bit” (bit 12) is not meaningful to the Intel387 SX Math CoProcessor and programs must ignore its value. To maintain compatibility with the 8087 and 80287 (non-387 core), this bit can be programmed, however, regardless of its value the Intel387 SX Math CoProcessor always treats infinity in the affine sense ( $-\infty < +\infty$ ). This bit is initialized to zero both after a hardware reset and after FINIT instruction.

All other bits are reserved and should not be programmed, to assure compatibility with future processors.

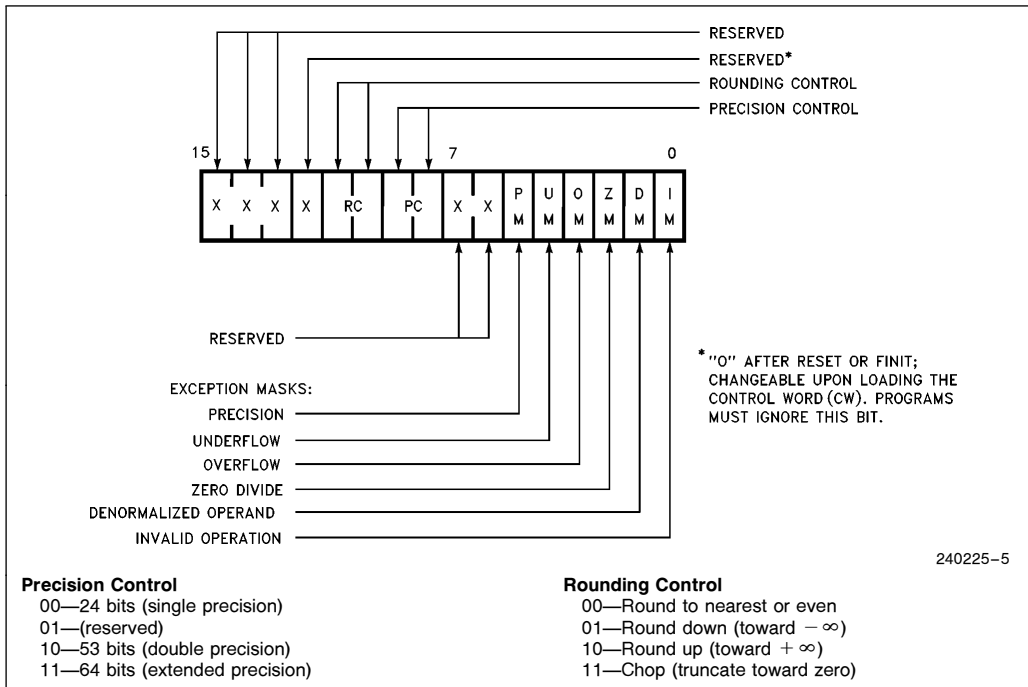


Figure 3-3. Control Word



**3.2.3 DATA REGISTER**

Intel387 SX Math CoProcessor data register set consists of eight registers (R0–R7) which are treated as both a stack and a general register file. Each of these data registers in the Math CoProcessor is 80 bits wide and is divided into fields corresponding to the Math CoProcessor’s extended-precision real data type, which is used for internal calculations.

The Math CoProcessor register set can be accessed either as a stack, with instructions operating on the top one or two stack elements, or as individually addressable registers. The TOP field in the status word identifies the current top-of-stack register. A “push” operation decrements TOP by one and loads a value into the new top register. A “store and pop” operation stores the value from the current top register into memory and then increments TOP by one. The Math CoProcessor register stack grows “down” toward lower-addressed registers.

Most of the Intel387 SX Math CoProcessor operations use the register stack as the operand(s) and/or as a place to store the result. Instructions may address the data register either implicitly or explicitly. Many instructions operate on the register at the top of the stack. These instructions implicitly address the register at which TOP points. Other instructions allow the programmer to explicitly specify which register to use. Explicit register addressing is also relative to TOP (where ST denotes the current stack top and ST(i) refers to the i’th register from the ST in the stack so the real register address is computed as ST+i).

**3.2.4 TAG WORD (TW) REGISTER**

The tag word marks the content of each numeric data register, as Figure 3-4 shows. Each two-bit tag represents one of the eight data register. The principal

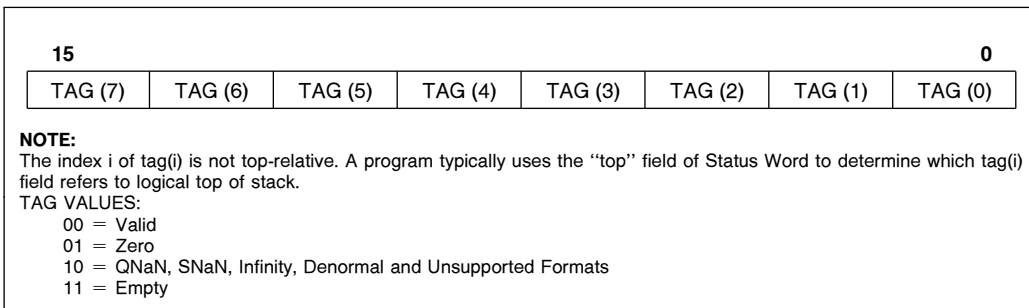
function of the tag word is to optimize the Math CoProcessor’s performance and stack handling by making it possible to distinguish between empty and non-empty register locations. It also enables exception handlers to identify special values (e.g. NaNs or denormals) in the contents of a stack location without the need to perform complex decoding of the actual data.

**3.2.5 INSTRUCTION AND DATA POINTERS**

Because the Math CoProcessor operates in parallel with the CPU, any exceptions detected by the Math CoProcessor may be reported after the CPU has executed the ESC instruction which caused it. To allow identification of the numeric instruction which caused the exception, the Intel386 Microprocessor contains registers that aid in diagnosis. These registers supply the address of the failing instruction and the address of its numeric memory operand (if appropriate).

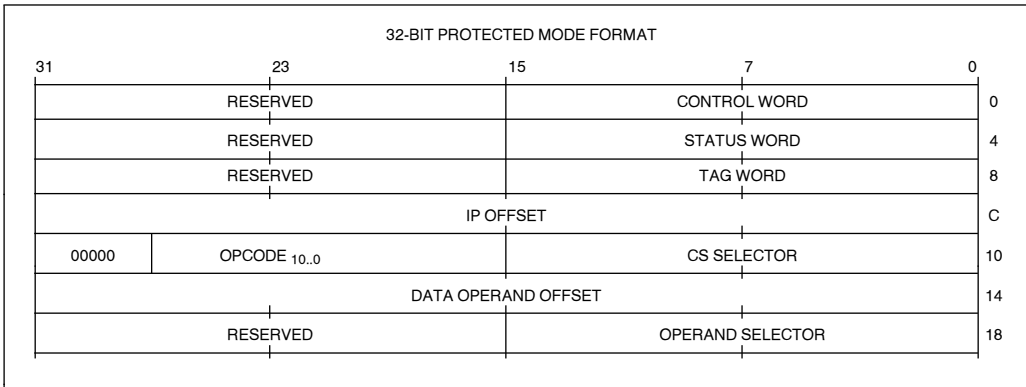
The instruction and data pointers are provided for user-written exception handlers. These registers are located in the CPU, but appear to be located in the Math CoProcessor because they are accessed by the ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR; which transfer the values between the registers and memory. Whenever the CPU executes a new ESC instruction (except administrative instructions), it saves the address of the instruction (including any prefixes that may be present), the address of the operand (if present) and the opcode.

The instruction and data pointers appear in one of four formats depending on the operating mode of the CPU (protected mode or real-address mode) and depending on the operand size attribute in effect (32-bit operand or 16-bit operand). (See Figures 3-5, 3-6, 3-7, and 3-8.) Note that the value of the data pointer is *undefined* if the prior ESC instruction did not have a memory operand.

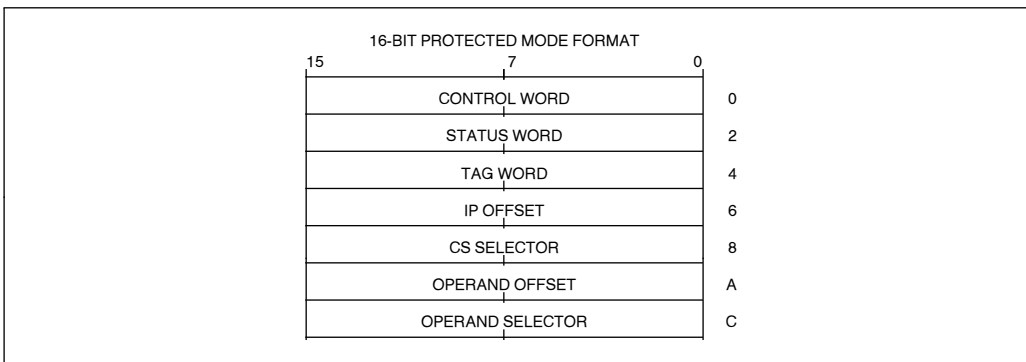


**Figure 3-4. Tag Word Register**

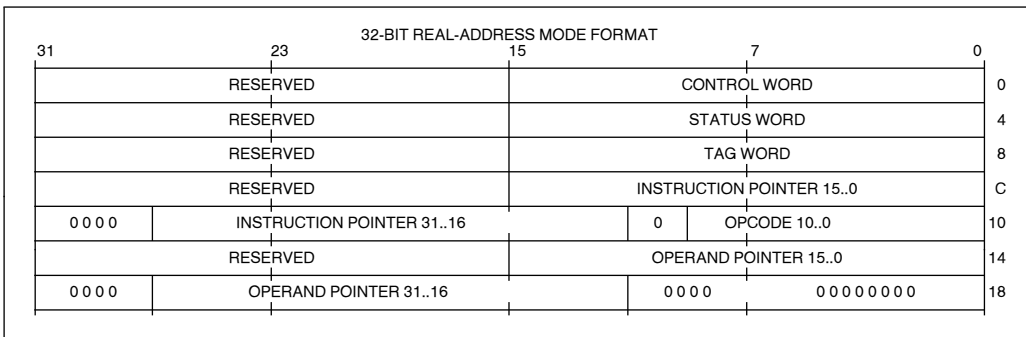




**Figure 3-5. Instruction and Data Pointer Image in Memory, 32-Bit Protected-Mode Format**



**Figure 3-6. Instruction and Data Pointer Image in Memory, 16-Bit Protected-Mode Format**



**Figure 3-7. Instruction and Data Pointer Image in Memory, 32-Bit Real-Mode Format**



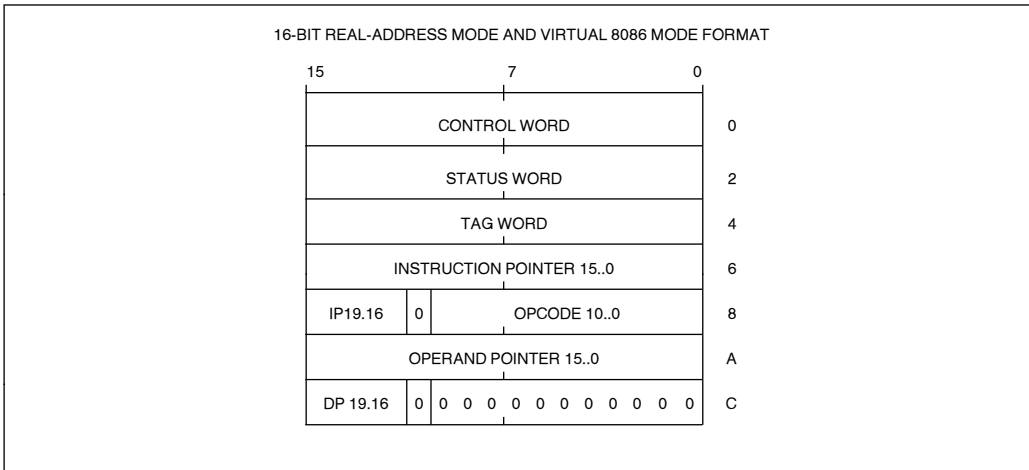


Figure 3-8. Instruction and Data Pointer Image in Memory, 16-Bit Real-Mode Format

### 3.3 Data Types

Table 3-6 lists the seven data types that the Math CoProcessor supports and presents the format for each type. Operands are stored in memory with the least significant digit at the lowest memory address. Programs retrieve these values by generating the lowest address. For maximum system performance, all operands should start at physical-memory addresses that correspond to the word size of the CPU; operands may begin at any other addresses, but will require extra memory cycles to access the entire operand.

The data type formats can be divided into three classes: binary integer, decimal integer, and binary real. These formats, however, exist in memory only. Internally, the Math CoProcessor holds all numbers in the extended-precision real format. Instructions that load operands from memory 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 extended-precision real format. Instructions that store operands in memory perform the inverse type conversion.

In addition to the typical real and integer data values, the Intel387 SX Math CoProcessor data formats encompass encodings for a variety of special values. These special values have significance and can express relevant information about the computations or operations that produced them. The various types of special values are denormal real numbers, zeros, positive and negative infinity, NaNs (Not-a-Number), Indefinite, and unsupported formats. For further information on data types and formats, see the Intel387 Programmer's Reference Manual.

### 3.4 Interrupt Description

CPU interrupts are used to report errors or exceptional conditions while executing numeric programs in either real or protected mode. Table 3-7 shows these interrupts and their functions.

### 3.5 Exception Handling

The Math CoProcessor detects six different exception conditions that occur during instruction execution. Table 3-8 lists the exception conditions in order of precedence, showing for each the cause and the default action taken by the Math CoProcessor if the exception is masked by its corresponding mask bit in the control word.

Any exception that is not masked by the control word sets the corresponding exception flag of the status word, sets the ES bit of the status word, and asserts the ERROR# signal. When the CPU attempts to execute another ESC instruction or WAIT, exception 16 occurs. The exception condition must be resolved via an interrupt service routine. The return address pushed onto the CPU stack upon entry to the service routine does not necessarily point to the failing instruction nor to the following instruction. The CPU saves the address of the floating-point instruction that caused the exception and the address of any memory operand required by that instruction.



Table 3-6. Intel387™ SX Math CoProcessor Data Type Representation in Memory

| Data Formats       | Range               | Precision | Most Significant Byte = HIGHEST ADDRESSED BYTE |                 |                 |                 |                 |                 |                 |                 |                 |                 |                |                |                |                |                |                |                |                |                |                |  |
|--------------------|---------------------|-----------|--|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|--|
|                    |                     |           | 7  | 0               | 7               | 0               | 7               | 0               | 7               | 0               | 7               | 0               | 7              | 0              | 7              | 0              |                |                |                |                |                |                |  |
| Word Integer       | $\pm 10^4$          | 16 Bits   | [15 to 0] (TWO'S COMPLEMENT)                   |                 |                 |                 |                 |                 |                 |                 |                 |                 |                |                |                |                |                |                |                |                |                |                |  |
| Short Integer      | $\pm 10^9$          | 32 Bits   | [31 to 0] (TWO'S COMPLEMENT)                   |                 |                 |                 |                 |                 |                 |                 |                 |                 |                |                |                |                |                |                |                |                |                |                |  |
| Long Integer       | $\pm 10^{18}$       | 64 Bits   | [63 to 0] (TWO'S COMPLEMENT)                   |                 |                 |                 |                 |                 |                 |                 |                 |                 |                |                |                |                |                |                |                |                |                |                |  |
| Packed BCD         | $\pm 10^{18}$       | 18 Digits | S  | X               | d <sub>17</sub> | d <sub>16</sub> | d <sub>15</sub> | d <sub>14</sub> | d <sub>13</sub> | d <sub>12</sub> | d <sub>11</sub> | d <sub>10</sub> | d <sub>9</sub> | d <sub>8</sub> | d <sub>7</sub> | d <sub>6</sub> | d <sub>5</sub> | d <sub>4</sub> | d <sub>3</sub> | d <sub>2</sub> | d <sub>1</sub> | d <sub>0</sub> |  |
| Single Precision   | $\pm 10^{\pm 38}$   | 24 Bits   | S  | BIASED EXPONENT |                 |                 | SIGNIFICAND     |                 |                 |                 |                 |                 |                |                |                |                |                |                |                |                |                |                |  |
| Double Precision   | $\pm 10^{\pm 308}$  | 53 Bits   | S  | BIASED EXPONENT |                 |                 | SIGNIFICAND     |                 |                 |                 |                 |                 |                |                |                |                |                |                |                |                |                |                |  |
| Extended Precision | $\pm 10^{\pm 4932}$ | 64 Bits   | S  | BIASED EXPONENT |                 |                 | I               | SIGNIFICAND     |                 |                 |                 |                 |                |                |                |                |                |                |                |                |                |                |  |

240225-23

**NOTES:**

1. S = Sign bit (0 = positive, 1 = negative)
2. d<sub>n</sub> = Decimal digit (two per byte)
3. X = Bits have no significance; Math CoProcessor ignores when loading, zeros when storing
4. ▲ = Position of implicit binary point
5. I = Integer bit of significand; stored in temporary real, implicit in single and double precision
6. Exponent Bias (normalized values):  
 Single: 127 (7FH)  
 Double: 1023 (3FFH)  
 Extended REal: 16383 (3FFFH)
7. Packed BCD:  $(-1)^S (D_{17}..D_0)$
8. Real:  $(-1)^S (2E\text{-BIAS}) (F_0 F_1..)$

Table 3-7. CPU Interrupt Vectors Reserved for Math CoProcessor

| Interrupt Number | Cause of Interrupt   |
|------------------|--|
| 7                | An ESC instruction was encountered when EM or TS of CPU control register zero (CR0) was set. EM = 1 indicates that software emulation of the instruction is required. When TS is set, either an ESC or WAIT instruction causes interrupt 7. This indicates that the current Math CoProcessor context may not belong to the current task.   |
| 9                | In a protected-mode system, an operand of a coprocessor instruction wrapped around an addressing limit (0FFFFH for expand-up segments, zero for expand-down segments) and spanned inaccessible addresses <sup>(1)</sup> . The failing numerics instruction is not restartable. The address of the failing numerics instruction and data operand may be lost; an FSTENV does not return reliable addresses. The segment overrun exception should be handled by executing an FNINIT instruction (i.e., an FINIT without a preceding WAIT). The exception can be avoided by never allowing numerics operands to cross the end of a segment. |
| 13               | In a protected-mode system, the first word of a numeric operand is not entirely within the limit of its segment. The return address pushed onto the stack of the exception handler points at the ESC instruction that caused the exception, including any prefixes. The Math CoProcessor has not executed this instruction; the instruction pointer and data pointer register refer to a previous, correctly executed instruction.   |
| 16               | The previous numerics instruction caused an unmasked exception. The address of the faulty instruction and the address of its operand are stored in the instruction pointer and data pointer registers. Only ESC and WAIT instructions can cause this interrupt. The CPU return address pushed onto the stack of the exception handler points to a WAIT or ESC instruction (including prefixes). This instruction can be restarted after clearing the exception condition in the Math CoProcessor. FNINIT, FNCLEX, FNSTSW, FNSTENV, and FNSAVE cannot cause this interrupt.   |

**NOTE:**

1. An operand may wrap around an addressing limit when the segment limit is near an addressing limit and the operand is near the largest valid address in the segment. Because of the wrap-around, the beginning and ending addresses of such an operand will be at opposite ends of the segment. There are two ways that such an operand may also span inaccessible addresses: 1) if the segment limit is not equal to the addressing limit (e.g. addressing limit is FFFFH and segment limit is FFFDH) the operand will span addresses that are not within the segment (e.g. an 8-byte operand that starts at valid offset FFFCH will span addresses FFFC–FFFFH and 0000-0003H; however addresses FFFEh and FFFFh are not valid, because they exceed the limit); 2) if the operand begins and ends in present and accessible segments but intermediate bytes of the operand fall in a not-present page or in a segment or page to which the procedure does not have access rights.

Table 3-8. Intel387™ SX Math CoProcessor Exceptions

| Exception                  | Cause  | Default Action (if exception is masked)                      |
|----------------------------|--|--|
| Invalid Operation          | Operation on a signalling NaN, unsupported format, indeterminate for $(0-\infty, 0/0, (+\infty) + (-\infty), \text{etc.})$ , or stack overflow/underflow (SF is also set). | Result is a quiet NaN, integer indefinite, or BCD indefinite |
| Denormalized Operand       | At least one of the operands is denormalized, i.e., it has the smallest exponent but a nonzero significand.  | Normal processing continues                                  |
| Zero Divisor               | The divisor is zero while the dividend is a noninfinite, nonzero number.   | Result is $\infty$   |
| Overflow                   | The result is too large in magnitude to fit in the specified format.   | Result is largest finite value or $\infty$                   |
| Underflow                  | The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes the loss of accuracy.    | Result is denormalized or zero                               |
| Inexact Result (Precision) | The true result is not exactly representable in the specified format (e.g. $1/3$ ); the result is rounded according to the rounding mode.                                  | Normal processing continues                                  |



### 3.6 Initialization

After FNINIT or RESET, the control word contains the value 037FH (all exceptions masked, precision control 64 bits, rounding to nearest) the same values as in an Intel287 after RESET. For compatibility with the 8087 and Intel287, the bit that used to indicate infinity control (bit 12) is set to zero; however, regardless of its setting, infinity is treated in the affine sense. After FNINIT or RESET, the status word is initialized as follows:

- All exceptions are set to zero.
- Stack TOP is zero, so that after the first push the stack top will be register seven (111B).
- The condition code C<sub>3</sub>–C<sub>0</sub> is undefined.
- The B-bit is zero.

The tag word contains FFFFH (all stack locations are empty).

The Intel386 Microprocessor and Intel387 Math CoProcessor initialization software must execute a FNINIT instruction (i.e., FINIT without a preceding WAIT) after RESET. The FNINIT is not strictly required for the Intel386 software, but Intel recommends its use to help ensure upware compatibility with other processors. After a hardware RESET, the ERROR# output is asserted to indicate that an Intel387 Math CoProcessor is present. To accomplish this, the IE (Invalid Exception) and ES (Error Summary) bits of the status word are set, and the IM bit (Invalid Exception Mask) in the control word is cleared. After FNINIT, the status word and the control word have the same values as in an Intel287 Math CoProcessor after RESET.

### 3.7 Processing Modes

The Intel387 SX Math CoProcessor works the same whether the CPU is executing in real-addressing mode, protected mode, or virtual-8086 mode. All references to memory for numerics data or status information are performed by the CPU, and therefore obey the memory-management and protection rules of the CPU mode currently in effect. The Intel387 SX Math CoProcessor merely operates on instruc-

tions and values passed to it by the CPU and therefore is not sensitive to the processing mode of the CPU.

The real-address mode and virtual-8086 mode, the Intel387 SX Math CoProcessor is completely upward compatible with software for the 8086/8087 and 80286/80287 real-address mode systems.

In protected mode, the Intel387 SX Math CoProcessor is completely upward compatible with software for the 80286/80287 protected mode system.

The only differences of operation that may appear when 8086/8087 programs are ported to the protected mode (not using virtual-8086 mode) is in the format of operands for the administrative instructions FLDENV, FSTENV, FRSTOR, and FSAVE.

### 3.8 Programming Support

Using the Intel387 SX Math CoProcessor requires no special programming tools, because all new instructions and data types are directly supported by the assembler and compilers for high-level languages. All Intel386 Microprocessor development tools that support Intel387 Math CoProcessor programs can also be used to develop software for the Intel386 SX Microprocessors and Intel387 SX Math CoProcessors. All 8086/8088 development tools that support the 8087 can also be used to develop software for the CPU and Math CoProcessor in real-address mode or virtual-8086 mode. All 80286 development tools that support the Intel287 Math CoProcessor can also be used to develop software for the Intel386 CPU and Intel387 Math CoProcessor.

## 4.0 HARDWARE SYSTEM INTERFACE

In the following description of hardware interface, the # symbol at the end of a signal name indicates that the active or asserted state occurs when the signal is at a low voltage. When no # is present after the signal name, the signal is asserted when at the high voltage level.

#### 4.1 Signal Description

In the following signal descriptions, the Intel387 SX Math CoProcessor pins are grouped by function as shown by Table 4-1. Table 4-1 lists every pin by its identifier, gives a brief description and lists some of its characteristics (Refer to Figure 1-1 and Table 1-1 for pin configuration).

All output signals can be tri-stated by driving STEN inactive. The output buffers of the bi-directional data pins D15–D0 are also tri-state; they only leave the floating state during read cycles when the Math CoProcessor is selected.

##### 4.1.1 Intel386 CPU CLOCK 2 (CPUCLK2)

This input uses the CLK2 signal of the CPU to time the bus control logic. Several other Math CoProcessor signals are referenced to the rising edge of this signal. When CKM = 1 (synchronous mode) this pin

also clocks the data interface and control unit and the floating point unit of the Math CoProcessor. This pin requires CMOS-level input. The signal on this pin is divided by two to produce the internal clock signal CLK.

##### 4.1.2 Intel387 MATH COPROCESSOR CLOCK 2 (NUMCLK2)

When CKM = 0 (asynchronous mode), this pin provides the clock for the data interface and control unit and the floating point unit of the Math CoProcessor. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10 and the maximum frequency must not exceed the device specifications. When CKM = 1 (synchronous mode), signals on this pin are ignored: CPUCLK2 is used instead for the data interface and control unit and the floating point unit. This pin requires CMOS level input and should be tied low if not used.

Table 4-1. Pin Summary

| Pin Name                          | Function                    | Active State | Input/Output | Referenced To ... |
|-----------------------------------|-----------------------------|--------------|--------------|-------------------|
| <b>Execution Control</b>          |                             |              |              |                   |
| CPUCLK2                           | Microprocessor Clock2       |              | I            |                   |
| NUMCLK2                           | Math CoProcessor Clock2     |              | I            |                   |
| CKM                               | Math CoProcessor Clock Mode |              | I            |                   |
| RESETIN                           | System Reset                | High         | I            | CPUCLK2           |
| <b>Math CoProcessor Handshake</b> |                             |              |              |                   |
| PEREQ                             | Processor Request           | High         | O            | CPUCLK2           |
| BUSY #                            | Busy Status                 | Low          | O            | CPUCLK2           |
| ERROR #                           | Error Status                | Low          | O            | NUMCLK2           |
| <b>Bus Interface</b>              |                             |              |              |                   |
| D15–D0                            | Data Pins                   |              | I/O          | CPUCLK2           |
| W/R #                             | Write/Read Bus Cycle        | High/Low     | I            | CPUCLK2           |
| ADS #                             | Address Strobe              | Low          | I            | CPUCLK2           |
| READY #                           | Bus Ready Input             | Low          | I            | CPUCLK2           |
| READYO #                          | Ready Output                | Low          | O            | CPUCLK2           |
| <b>Chip/Port Select</b>           |                             |              |              |                   |
| STEN                              | Status Enable               | High         | I            | CPUCLK2           |
| NPS1 #                            | Numerics Select #1          | Low          | I            | CPUCLK2           |
| NPS2                              | Numerics Select #2          | High         | I            | CPUCLK2           |
| CMD0 #                            | Command                     | Low          | I            | CPUCLK2           |
| <b>Power and Ground</b>           |                             |              |              |                   |
| V <sub>CC</sub>                   | System Power                |              |              |                   |
| V <sub>SS</sub>                   | System Ground               |              |              |                   |



4.1.3 CLOCKING MODE (CKM)

This pin is strapping option. When it is strapped to VCC (HIGH), the Math CoProcessor operates in synchronous mode; when strapped to VSS (LOW), the Math CoProcessor operates in asynchronous mode. These modes relate to clocking of the internal data interface and control unit and the floating point unit only; the bus control logic always operates synchronously with respect to the CPU.

Synchronous mode requires the use of only one clock, the CPU's CLK2. Use of synchronous mode eliminates one clock generator from the board design and is recommended for all designs. Synchronous mode also allows the internal Power Management Unit to enable the idle and standby power saving modes.

Asynchronous mode can provide higher performance of the floating point unit by running a faster clock on NUMCLK2. (The CPU's CLK2 must still be connected to CPUCLK2 input.) This allows the floating point unit to run up to 40% faster than in synchronous mode. Internal power management is disabled in asynchronous mode.

4.1.4 SYSTEM RESET (RESETIN)

A LOW to HIGH transition on this pin causes the Math CoProcessor to terminate its present activity and to enter a dormant state. RESETIN must remain active (HIGH) for at least 40 CPUCLK2 (NUMCLK2 if CKM = 0) periods.

The HIGH to LOW transitions of RESETIN must be synchronous with CPUCLK2, so that the phase of the internal clock of the bus control logic (which is the CPUCLK2 divided by two) is the same as the phase of the internal clock of the CPU. After RESETIN goes LOW, at least 50 CPUCLK2 (NUMCLK2 if CKM = 0) periods must pass before the first Math CoProcessor instruction is written into the Math CoProcessor. This pin should be connected to the CPU RESET pin. Table 4-2 shows the status of the output pins during the reset sequence. After a reset, all output pins return to their inactive state except for ERROR# which remains active (for CPU recognition) until cleared.

Table 4-2. Output Pin Status during Reset

| Pin Value     | Pin Name       |
|---------------|----------------|
| HIGH          | READYO#, BUSY# |
| LOW           | PEREQ, ERROR#  |
| Tri-State OFF | D15-D0         |

4.1.5 PROCESSOR REQUEST (PEREQ)

When active, this pin signals to the CPU that the Math CoProcessor is ready for data transfer to/from its data FIFO. When all data is written to or read from the data FIFO, PEREQ is deactivated. This signal always goes inactive before BUSY# goes inactive. This signal is reference to CPUCLK2. It should be connected to the CPU PEREQ input pin.

4.1.6 BUSY STATUS (BUSY#)

When active, this pin signals to the CPU that the Math CoProcessor is currently executing an instruction. This signal is referenced to CPUCLK2. It should be connected to the CPU BUSY# input pin.

4.1.7 ERROR STATUS (ERROR#)

This pin reflects the ES bit of the status register. When active, it indicates that an unmasked exception has occurred. This signal can be changed to the inactive state only by the following instructions (without a preceding WAIT); FNINIT, FNCLEX, FNSTENV, FNSAVE, FLDCW, FLDENV, and FRSTOR. ERROR# is driven active during RESET to indicate to the CPU that the Math CoProcessor is present. This pin is referenced to NUMCLK2 (or CPUCLK2 if CKM = 1). It should be connected to the ERROR# pin of the CPU.

4.1.8 DATA PINS (D15-D0)

These bi-directional pins are used to transfer data and opcodes between the CPU and Math CoProcessor. They are normally connected directly to the corresponding CPU data pins. HIGH state indicates a value of one. D0 is the least significant data bit. Timings are referenced to rising edge of CPUCLK2.

4.1.9 WRITE/READ BUS CYCLE (W/R#)

This signal indicates to the Math CoProcessor whether the CPU bus cycle in progress is a read or a write cycle. This pin should be connected directly to the CPU's W/R# pin. HIGH indicates a write cycle to the Math CoProcessor; LOW a read cycle from the Math CoProcessor. This input is ignored if any of the signals STEN, NPS1#, or NPS2 are inactive. Setup and hold times are referenced to CPUCLK2.

4.1.10 ADDRESS STROBE (ADS#)

This input, in conjunction with the READY# input, indicates when the Math CoProcessor bus control logic may sample W/R# and the chip select signals. Setup and hold times are referenced to CPUCLK2. This pin should be connected to the ADS# pin of the CPU.

#### 4.1.11 BUS READY INPUT (READY#)

This input indicates to the Math CoProcessor when a CPU bus cycle is to be terminated. It is used by the bus control logic to trace bus activities. Bus cycles can be extended indefinitely until terminated by READY#. This input should be connected to the same signal that drives the CPU's READY# input. Setup and hold times are referenced to CPUCLK2.

#### 4.1.12 READY OUTPUT (READYO#)

This pin is activated at such a time that write cycles are terminated after two clocks (except FLDENV and FRSTOR) and read cycles after three clocks. In configurations where no extra wait states are required, this pin must directly or indirectly drive the READY# input of the CPU. Refer to the section entitled "BUS OPERATION" for details. This pin is activated only during bus cycles that select the Math CoProcessor. This signal is referenced to CPUCLK2.

(FLDENV and FRSTOR require data transfers larger than the FIFO. Therefore, PEREQ is activated for the duration of transferring 2 words of 32 bits and then deactivated until the FIFO is ready to accept two additional words. The length of the write cycles of the last operand word in each transfer as well as the first operand word transfer of the entire instruction is 3 clocks instead of 2 clocks. This is done to give the Intel386 CPU enough time to sample PEREQ and to notice that the Intel387 is **not** ready for additional transfers.)

#### 4.1.13 STATUS ENABLE (STEN)

This pin serves as a chip select for the Math CoProcessor. When inactive, this pin forces BUSY#, PEREQ, ERROR# and READYO# outputs into a floating state. D15–D0 are normally floating and will leave the floating state only if STEN is active and additional conditions are met (read cycle). STEN also causes the chip to recognize its other chip select inputs. STEN makes it easier to do on-board testing (using the overdrive method) of other chips in systems containing the Math CoProcessor. STEN should be pulled up with a resistor so that it can be pulled down when testing. In boards that do not use on-board testing STEN should be connected to V<sub>CC</sub>. Setup and hold times are relative to CPUCLK2. Note that STEN must maintain the same setup and hold times as NPS1#, NPS2, and CMD0# (i.e., if STEN changes state during a Math CoProcessor bus cycle, it must change state during the same CLK period as the NPS1#, NPS2, and CMD0# signals).

#### 4.1.14 MATH COPROCESSOR SELECT 1 (NPS1#)

When active (along with STEN and NPS2) in the first period of a CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the Math CoProcessor. This pin should be connected directly to the M/IO# pin of the CPU, so that the Math CoProcessor is selected only when the CPU performs I/O cycles. Setup and hold times are referenced to the rising edge of CPUCLK2.

#### 4.1.15 MATH COPROCESSOR SELECT 2 (NPS2)

When active (along with STEN and NPS1#) in the first period of a CPU bus cycle, this signal indicates that the purpose of the bus cycle is to communicate with the Math CoProcessor. This pin should be connected directly to the A23 pin of the CPU, so that the Math CoProcessor is selected only when the CPU issues one of the I/O addresses reserved for the Math CoProcessor (8000F8h, 8000FCh, or 8000FEh which is treated as 8000FCh by the Math CoProcessor). Setup and hold times are referenced to the rising edge of CPUCLK2.

#### 4.1.16 COMMAND (CMD0#)

During a write cycle, this signal indicates whether an opcode (CMD0# active low) or data (CMD0# inactive high) is being sent to the Math CoProcessor. During a read cycle, it indicates whether the control or status register (CMD0# active) or a data register (CMD0#) is being read. CMD0# should be connected directly to the A2 output of the CPU. Setup and hold times are referenced to the rising edge of CPUCLK2 at the end of PH2.

#### 4.1.17 SYSTEM POWER (V<sub>CC</sub>)

System power provides the +5V DC supply input. All V<sub>CC</sub> pins should be tied together on the circuit board and local decoupling capacitors should be used between V<sub>CC</sub> and V<sub>SS</sub>.

#### 4.1.18 SYSTEM GROUND (V<sub>SS</sub>)

System ground provides the 0V connection from which all inputs and outputs are measured. All V<sub>SS</sub> pins should be tied together on the circuit board and local decoupling capacitors should be used between V<sub>CC</sub> and V<sub>SS</sub>.



### 4.2 System Configuration

The Intel387 SX Math CoProcessor is designed to interface with the Intel386 SX Microprocessor as shown by Figure 4-1. A dedicated communication protocol makes possible high-speed transfer of op-codes and operands between the CPU and Math CoProcessor. The Intel387 SX Math CoProcessor is designed so that no additional components are required for interface with the CPU. Most control pins of the Math CoProcessor are connected directly to pins of the CPU.

The interface between the Math CoProcessor and the CPU has these characteristics:

- The Math CoProcessor shares the local bus of the Intel386 SX Microprocessor.

- The CPU and Math CoProcessor share the same reset signals. They may also share the same clock input; however, for greatest performance, an external oscillator may be needed.
- The corresponding Busy#, ERROR#, and PEREQ pins are connected together.
- The Math CoProcessor NPS1# and NPS2 inputs are connected to the latched CPU M/IO# and A23 outputs respectively. For Math CoProcessor cycles, M/IO# is always LOW and A23 always HIGH.
- The Math CoProcessor input CMD0 is connected to the latched A<sub>2</sub> output. The Intel386 SX Microprocessor generates address 8000F8H when writing a command and address 8000FCH or 8000FEH (treated as 8000FCH by the Intel387 SX Math CoProcessor) when writing or reading data. It does not generate any other addresses during Math CoProcessor bus cycles.

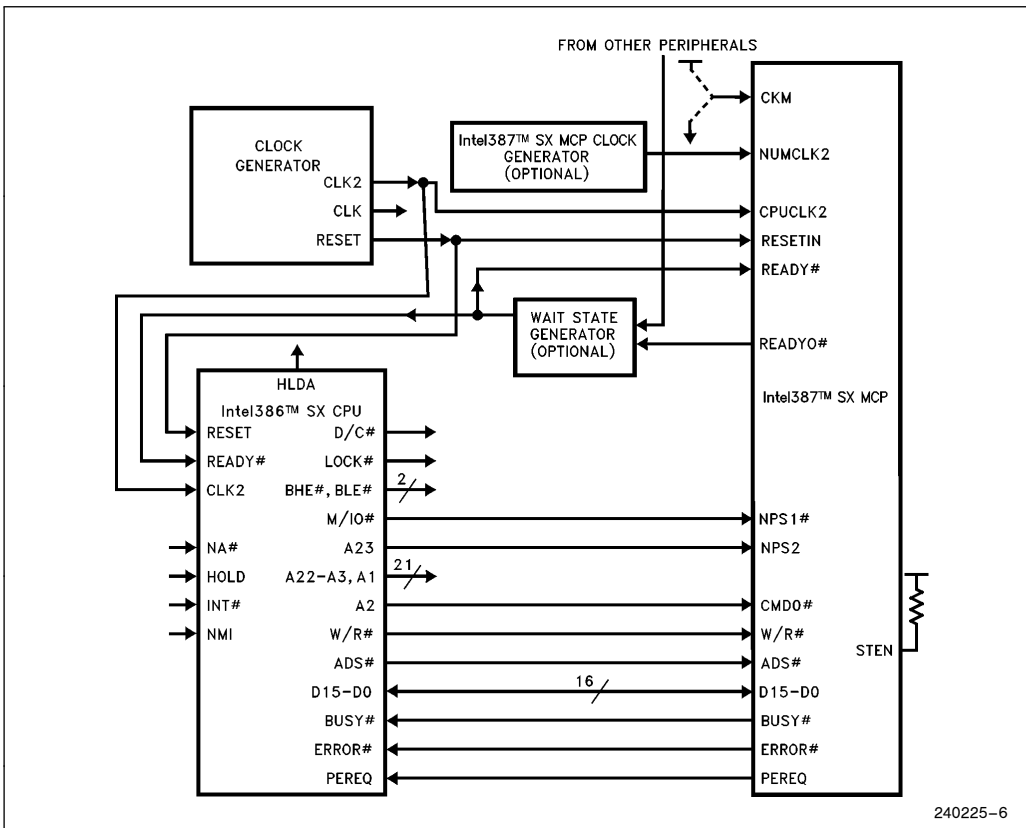


Figure 4-1. Intel386™ SX CPU and Intel387™ SX Math CoProcessor System Configuration



### 4.3 Math CoProcessor Architecture

As shown in Figure 2-1 Block Diagram, the Intel387 SX Math CoProcessor is internally divided into four sections; the Bus Control Logic (BCL), the Data Interface and Control Logic, the Floating Point Unit (FPU), and the Power Management Unit (PMU). The Bus Control Logic is responsible for the CPU bus tracking and interface. The BCL is the only unit in the Math CoProcessor that must run synchronously with the CPU; the rest of the Math CoProcessor can run asynchronously with respect to the CPU. The Data Interface and Control Unit is responsible for the data flow to and from the FPU and the control registers, for receiving the instructions, decoding them, sequencing the microinstructions, and for handling some of the administrative instructions. The Floating Point Unit (with the support of the control unit which contains the sequencer and other support units) executes the mathematical instructions. The Power Manager is new to the Intel387 family. It is responsible for shutting down idle sections of the device to save power.

#### 4.3.1 BUS CONTROL LOGIC

The BCL communicates solely with the CPU using I/O bus cycles. The BCL appears to the CPU as a special peripheral device. It is special in two respects: the CPU initiates I/O automatically when it encounters ESC instructions, and the CPU uses reserved I/O addresses to communicate with the BCL. The BCL does not communicate directly with memory. The CPU performs all memory access, transferring input operands from the memory to the Math CoProcessor and transferring outputs from the Math CoProcessor to memory.

#### 4.3.2 DATA INTERFACE AND CONTROL UNIT

The data interface and control unit latches the data and, subject to BCL control, directs the data to the

FIFO or the instruction decoder. The instruction decoder decodes the ESC instructions sent to it by the CPU and generates controls that direct the data flow in the FIFO. It also triggers the microinstruction sequencer that controls execution of each instruction. If the ESC instruction is FINIT, FCLEX, FSTSW, FSTSW AX, FSTCW, FSETPM, or FRSTPM, the control unit executes it independently of the FPU and the sequencer. The data interface and control unit is the unit that generates the BUSY#, PEREQ, and ERROR# signals that synchronize the Math CoProcessor activities with the CPU.

#### 4.3.3 FLOATING POINT UNIT

The FPU executes all instructions that involve the register stack, including arithmetic, logical, transcendental, constant, and data transfer instructions. The data path in the FPU is 84 bits wide (68 significant bits, 15 exponent bits, and a sign bit) which allows internal operand transfers to be performed at very high speeds.

#### 4.3.4 POWER MANAGEMENT UNIT

The Power Management Unit (PMU) controls all internal power savings circuits. When the Math CoProcessor is not executing an instruction, the PMU disables the internal clock to the FPU, Control Unit, and Data Interface within three clocks. The Bus Control Logic remains enabled to accept the next instruction. Upon decode of a valid Math CoProcessor bus cycle, the PMU enables the internal clock to all circuits. No loss in performance occurs.

### 4.4 Bus Cycles

All bus cycles are initiated by the CPU. The pins STEN, NPS1#, NPS2, CMD0, and W/R# identify bus cycles for the Math CoProcessor. Table 4-3 defines the types of Math CoProcessor bus cycles.

Table 4-3. Bus Cycle Definition

| STEN | NPS1 # | NPS2 | CMD0 # | W/R # | Bus Cycle Type  |
|------|--------|------|--------|-------|---|
| 0    | X      | X    | X      | X     | Math CoProcessor not selected and all outputs in floating state |
| 1    | 1      | X    | X      | X     | Math CoProcessor not selected                                   |
| 1    | X      | 0    | X      | X     | Math CoProcessor not selected                                   |
| 1    | 0      | 1    | 0      | 0     | CW or SW read from Math CoProcessor                             |
| 1    | 0      | 1    | 0      | 1     | Opcode write to Math CoProcessor                                |
| 1    | 0      | 1    | 1      | 0     | Data read from Math CoProcessor                                 |
| 1    | 0      | 1    | 1      | 1     | Data write to Math CoProcessor                                  |



#### 4.4.1 INTEL387 SX MATH COPROCESSOR ADDRESSING

The NPS1#, NPS2, and CMD0 signals allow the Math CoProcessor to identify which bus cycles are intended for the Math CoProcessor. The Math CoProcessor responds to I/O cycles when the I/O address is 8000F8h, 8000FCh, and 8000FEh (treated as 8000FCh). The Math CoProcessor responds to I/O cycles when bit 23 of the I/O address is set. In other words, the Math CoProcessor acts as an I/O device in a reserved I/O address space.

Because A23 is used to select the Intel387 SX Math CoProcessor for data transfers, it is not possible for a program running on the CPU to address the Math CoProcessor with an I/O instruction. Only ESC instructions cause the CPU to communicate with the Math CoProcessor.

#### 4.4.2 CPU/MATH COPROCESSOR SYNCHRONIZATION

The pins BUSY#, PEREQ, and ERROR# are used for various aspects of synchronization between the CPU and the Math CoProcessor.

BUSY# is used to synchronize instruction transfer from the CPU to the Math CoProcessor. When the Math CoProcessor recognizes an ESC instruction it asserts BUSY#. For most ESC instructions, the CPU waits for the Math CoProcessor to deassert BUSY# before sending the new opcode.

The Math CoProcessor uses the PEREQ pin of the CPU to signal that the Math CoProcessor is ready for data transfer to or from its data FIFO. The Math CoProcessor does not directly access memory; rather, the CPU provides memory access services for the Math CoProcessor. (For this reason, memory access on behalf of the Math CoProcessor always obeys the protection rules applicable to the current CPU mode.) Once the CPU initiates a Math CoProcessor instruction that has operands, the CPU waits for PEREQ signals that indicate when the Math CoProcessor is ready for operand transfer. Once all operands have been transferred (or if the instruction has no operands) the CPU continues program execution while the Math CoProcessor executes the ESC instruction.

In 8087/8087 systems, WAIT instructions may be required to achieve synchronization of both commands and operands. In the Intel386 Microprocessor and Intel387 Math CoProcessor systems, however, WAIT instructions are required only for operand synchronization; namely, after Math CoProcessor stores to memory (except FSTSW and FSTCW) or load from memory. (In 80286/80287 systems, WAIT is required before FLDENV and FRSTOR.) Used this way, WAIT ensures that the

value has already been written or read by the Math CoProcessor before the CPU reads or changes the value.

Once it has started to execute a numerics instruction and has transferred operands from the CPU, the Math CoProcessor can process the instruction in parallel with and independent of the host CPU. When the Math CoProcessor detects an exception, it asserts the ERROR# signal, which causes a CPU interrupt.

#### 4.4.3 SYNCHRONOUS/ASYNCHRONOUS MODES

The internal logic of the Math CoProcessor can operate either directly from the CPU clock (synchronous mode) or from a separate clock (asynchronous mode). The two configurations are distinguished by the CKM pin. In either case, the bus control logic (BCL) of the Math CoProcessor is synchronized with the CPU clock. Use of asynchronous mode allows the BCL and the FPU section of the Math CoProcessor to run at different speeds. In this case, the ratio of the frequency of NUMCLK2 to the frequency of CPUCLK2 must lie within the range 10:16 to 14:10. Use of synchronous mode eliminates one clock generator from the board design. The internal Power Management Unit of the Intel387 SX Math CoProcessor is disabled in asynchronous mode.

#### 4.4.4 AUTOMATIC BUS CYCLE TERMINATION

In configurations where no extra wait states are required, READYO# can drive the CPU's READY# input and the Math CoProcessors READY# input. If wait states are required, this pin should be connected to the logic that ORs all READY outputs from peripheral devices on the CPU bus. READYO# is asserted by the Math CoProcessor only during I/O cycles that select the Math CoProcessor. Refer to Section 5.0 Bus Operation for details.

### 5.0 BUS OPERATION

With respect to bus interface, the Intel387 SX Math CoProcessor is fully synchronous with the CPU. Both operate at the same rate because each generates its internal CLK signal by dividing CPUCLK2 by two. Furthermore, both internal CLK signals are in phase, because they are synchronized by the same RESETIN signal.

A bus cycle for the Math CoProcessor starts when the CPU activates ADS# and drives new values on the address and cycle definition lines (W/R#, M/IO#, etc.). The Math CoProcessor examines the address and cycle definition lines in the same CLK period during which ADS# is activated. This CLK period is considered the first CLK of the bus cycle.

During this first CLK period, the Math CoProcessor also examines the W/R# input signal to determine whether the cycle is a read or a write cycle and examines the CMD0# input to determine whether an opcode, operand, or control/status register transfer is to occur.

The Intel387 SX Math CoProcessor supports both pipelined (i.e., overlapped) and non-pipelined bus cycles. A non-pipelined cycle is one for which the CPU asserts ADS# when no other bus cycle is in progress. A pipelined bus cycle is one for which the CPU asserts ADS# and provides valid next address and control signals before the prior Math CoProcessor cycle terminates. The CPU may do this as early as the second CLK period after asserting ADS# for the prior cycle. Pipelining increases the availability of the bus by at least one CLK period. The Intel387 SX Math CoProcessor supports pipelined bus cycles in order to optimize address pipelining by the CPU for memory cycles.

Bus operation is described in terms of an abstract state machine. Figure 5-1 illustrates the states and state transitions for Math CoProcessor bus cycles:

- T<sub>I</sub> is the idle state. This is the state of the bus logic after RESET, the state to which bus logic returns after every non-pipelined bus cycle, and the state to which bus logic returns after a series of pipelined cycles.
- T<sub>RS</sub> is the READY# sensitive state. Different types of bus cycles may require a minimum of one or two successive T<sub>RS</sub> states. The bus logic remains in T<sub>RS</sub> state until READY# is sensed, at which point the bus cycle terminates. Any number of wait states may be implemented by delaying READY#, thereby causing additional successive T<sub>RS</sub> states.
- T<sub>P</sub> is the first state for every pipelined bus cycle. This state is not used by non-pipelined cycles.

Note that the bus logic tracks bus state regardless of the values on the chip/port select pins. The

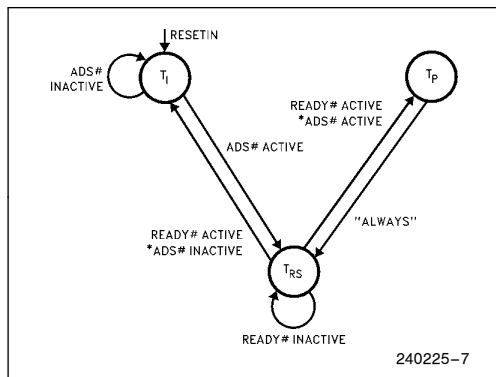


Figure 5-1. Bus State Diagram

READYO# output of the Math CoProcessor indicates when a Math CoProcessor bus cycle may be terminated if no extra wait states are required. For all write cycles (except those for the instructions FLDENV and FRSTOR), READYO# is always asserted during the first T<sub>RS</sub> state, regardless of the number of wait states. For all read cycles (and write cycles for FLDENV and FRSTOR), READY# is always asserted in the second T<sub>RS</sub> state, regardless of the number of wait states. These rules apply to both pipelined and non-pipelined cycles. Systems designers may use READYO# in one of the following ways:

1. Connect it (directly or through logic that ORs READY# signals from other devices) to the READY# inputs of the CPU and Math CoProcessor.
2. Use it as one input to a wait-state generator.

The following sections illustrate different types of Intel387 SX Math CoProcessor bus cycles. Because different instructions have different amounts of overhead before, between, and after operand transfer cycles, it is not possible to represent in a few diagrams all of the combinations of successive operand transfer cycles. The following bus cycle diagrams show memory cycles between Math CoProcessor operand transfer cycles. Note however that, during FRSTOR, some consecutive accesses to the Math CoProcessor do not have intervening memory accesses. For the timing relationship between operand transfer cycles and opcode write or other overhead activities, see Figure 7-7 “Other Parameters”.

### 5.1 Non-Pipelined Bus Cycles

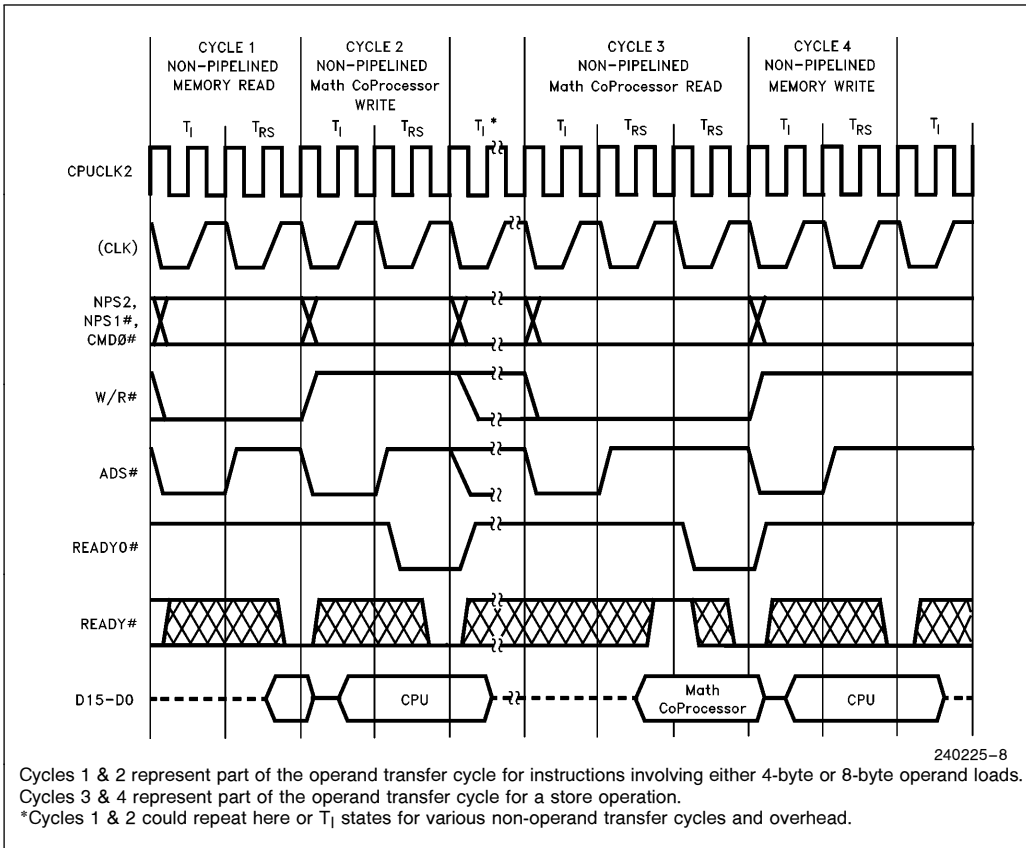
Figure 5-2 illustrates bus activity for consecutive non-pipelined bus cycles.

At the second clock of the bus cycle, the Math CoProcessor enters the T<sub>RS</sub> state. During this state, it samples the READY# input and stays in this state as long as READY# is inactive.

#### 5.1.1 WRITE CYCLE

In write cycles, the Math CoProcessor drives the READYO# signal for one CLK period during the second CLK period of the cycle (i.e., the first T<sub>RS</sub> state); therefore, the fastest write cycle takes two CLK periods (see cycle 2 of Figure 5-2). For the instructions FLDENV and FRSTOR, however, the Math CoProcessor forces wait state by delaying the activation of READYO# to the second T<sub>RS</sub> state (not shown in Figure 5-2).

The Math CoProcessor samples the D15–D0 inputs into data latches at the falling edge of CLK as long as it stays in T<sub>RS</sub> state.



**Figure 5-2. Non-Pipelined Read and Write Cycles**

When **READY#** is asserted, the Math CoProcessor returns to the idle state. Simultaneously with the Math CoProcessor entering the idle state, the CPU may assert **ADS#** again, signaling the beginning of yet another cycle.

**5.1.2 READ CYCLE**

At the rising edge of **CLK** in the second **CLK** period of the cycle (i.e., the first **T<sub>RS</sub>** state), the Math CoProcessor starts to drive the **D15-D0** outputs and continues to drive them as long as it stays in **T<sub>RS</sub>** state.

At least one wait state must be inserted to ensure that the CPU latches the correct data. Because the Math CoProcessor starts driving the data bus only at the rising edge of **CLK** in the second clock period of the bus cycle, not enough time is left for the data signals to propagate and be latched by the CPU before the next falling edge of **CLK**. Therefore, the Math CoProcessor does not drive the **READY0#**

signal until the third **CLK** period of the cycle. Thus, if the **READY0#** output drives the CPU's **READY#** input, one wait state is automatically inserted.

Because one wait state is required for Math CoProcessor reads, the minimum length of a Math CoProcessor read cycle is three **CLK** periods, as cycle 3 of Figure 5-2 shows.

When **READY#** is asserted, the Math CoProcessor returns to the idle state. Simultaneously with the Math CoProcessor's entering the idle state, the CPU may assert **ADS#** again, signaling the beginning of yet another cycle. The transition from **T<sub>RS</sub>** state to idle state causes the Math CoProcessor to put the **D15-D0** outputs into the floating state, allowing another device to drive the data bus.

**5.2 Pipelined Bus Cycles**

Because all the activities of the Math CoProcessor bus interface occur either during the **T<sub>RS</sub>** state or

during the transitions to or from that state, the only difference between a pipelined and a non-pipelined cycle is the manner of changing from one state to another. The exact activities during each state are detailed in the previous section “Non-pipelined Bus Cycles”.

When the CPU asserts ADS# before the end of a bus cycle, both ADS# and READY# are active during a  $T_{RS}$  state. This condition causes the Math CoProcessor to change to a different state named  $T_P$ . One clock period after a  $T_P$  state, the Math CoProcessor always returns to the  $T_{RS}$  state. In consecutive pipelined cycles, the Math CoProcessor bus logic uses only the  $T_{RS}$  and  $T_P$  states.

Figure 5-3 shows the fastest transitions into and out of the pipelined bus cycles. Cycle 1 in the figure represents a non-pipelined cycle. (Non-pipelined write are always followed by another non-pipelined cycle,

because READY# is asserted before the earliest possible assertion of ADS# for the next cycle.)

Figure 5-4 shows pipelined write and read cycles with one additional  $T_{RS}$  state beyond the minimum required. To delay the assertion of READY# requires external logic.

### 5.3 Mixed Bus Cycles

When the Math CoProcessor bus logic is in the  $T_{RS}$  state, it distinguishes between non-pipelined and pipelined cycles according to the behavior of ADS# and READY#. In a non-pipelined cycle, only READY# is activated, and the transition is from the  $T_{RS}$  state to the idle state. In a pipelined cycle, both READY# and ADS# are active, and the transition is first from  $T_{RS}$  state to  $T_P$  state, then, after one clock period, back to  $T_{RS}$  state.

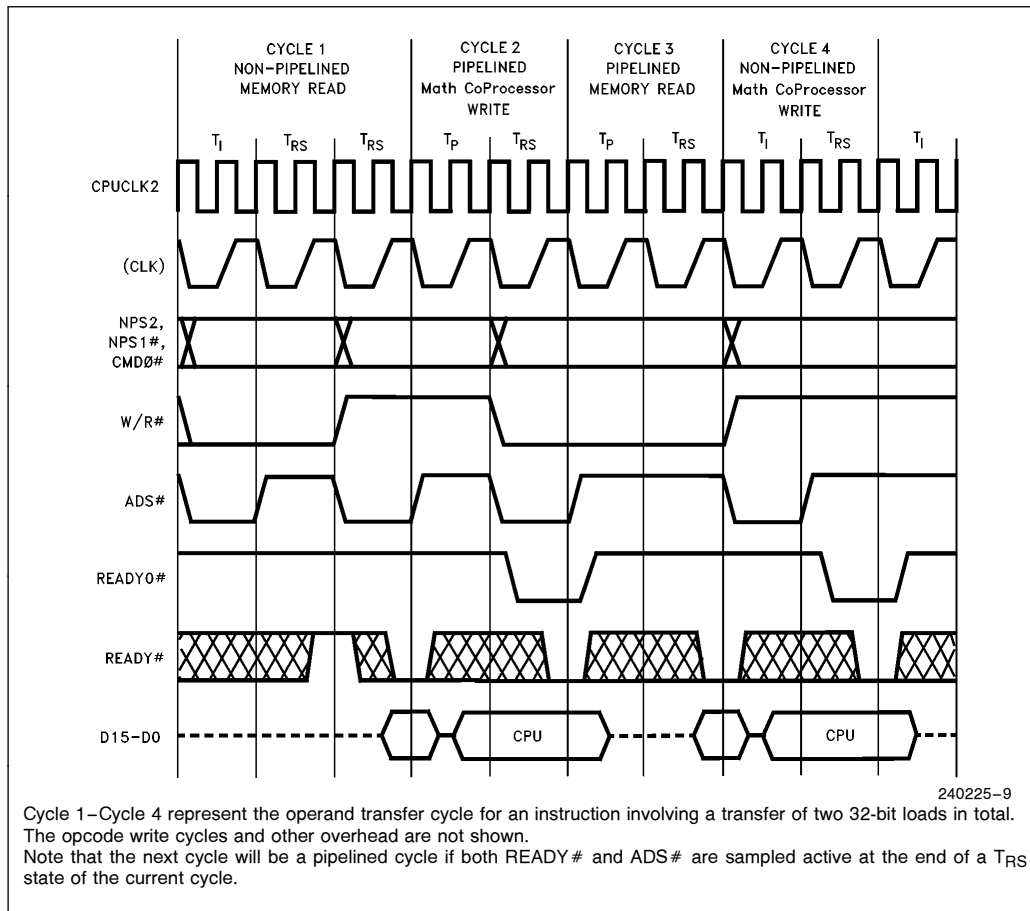


Figure 5-3. Fastest Transitions to and from Pipelined Cycles

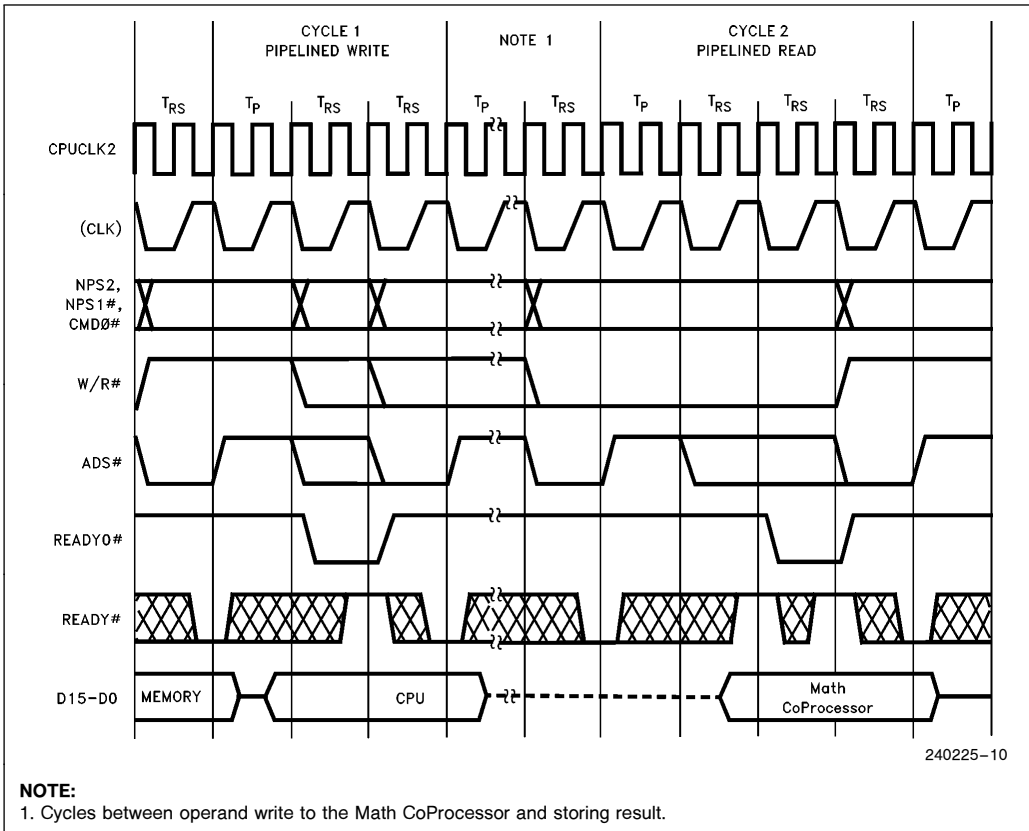


Figure 5-4. Pipelined Cycles with Wait States

### 5.4 BUSY# and PEREQ Timing Relationship

Figure 5-5 shows the activation of BUSY# at the beginning of instruction execution and its deactivation upon completion of the instruction.

PEREQ is activated within this interval. If ERROR# is ever asserted, it would be asserted at least six CPUCLK2 periods after the deactivation of PEREQ and would be deasserted at least six CPUCLK2 periods before the deactivation of BUSY#.

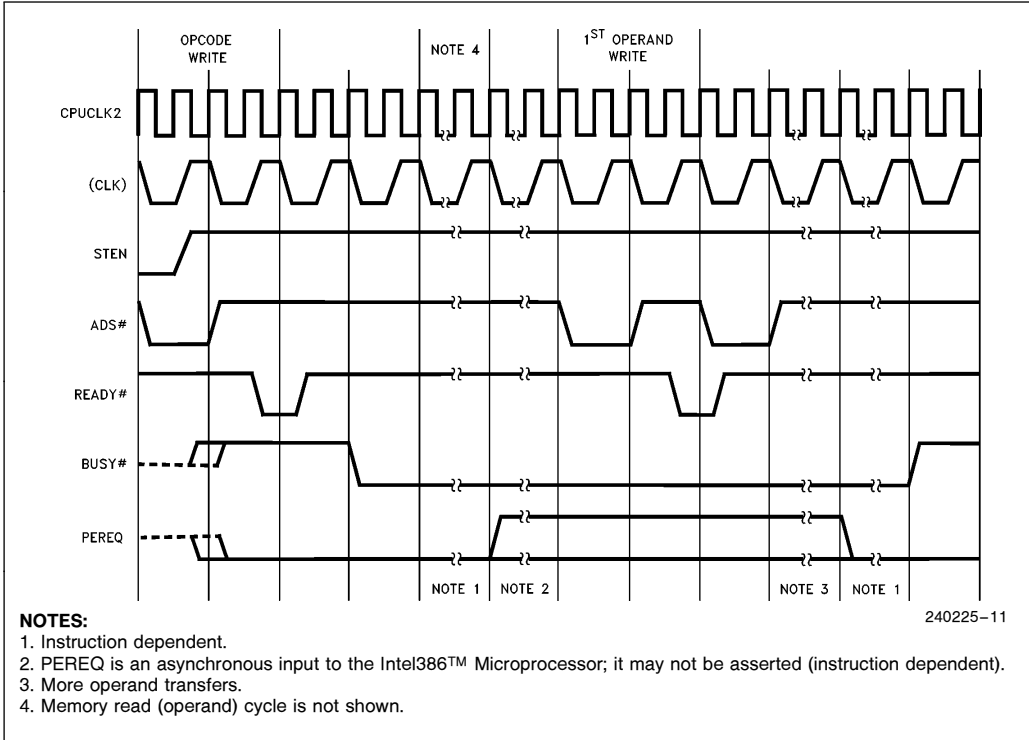


Figure 5-5. STEN, BUSY#, and PEREQ Timing Relationships





**6.0 PACKAGE SPECIFICATIONS**

**6.1 Mechanical Specifications**

The Intel387 SX Math CoProcessor is packaged in a 68-pin PLCC package. Detailed mechanical specifications can be found in the Intel Packaging Specification, Order Number 231369.

**6.2 Thermal Specifications**

The Intel387 SX Math CoProcessor is specified for operation when the case temperature is within the range of 0°C to 100°C. The case temperature (T<sub>C</sub>) may be measured in any environment to determine whether the Intel387 SX Math CoProcessor is within the specified operating range. The case temperature should be measured at the center of the top surface.

The ambient temperature (T<sub>A</sub>) is guaranteed as long as T<sub>C</sub> is not violated. The ambient temperature can be calculated from the θ<sub>JC</sub> (thermal resistance constant from the transistor junction to the case) and θ<sub>JA</sub> (thermal resistance from junction to ambient) from the following calculations:

$$\begin{aligned} \text{Junction Temperature } T_J &= T_C + P \cdot \theta_{JC} \\ \text{Ambient Temperature } T_A &= T_J - P \cdot \theta_{JA} \\ \text{Case Temperature } T_C &= T_A + P \cdot (\theta_{JA} - \theta_{JC}) \end{aligned}$$

Values for θ<sub>JA</sub> and θ<sub>JC</sub> are given in Table 6-1 for the 68 pin PLCC package. θ<sub>JC</sub> is given at various airflows. Table 6-2 shows the maximum T<sub>A</sub> allowable without exceeding T<sub>C</sub> at various airflows. Note that T<sub>A</sub> can be improved further by attaching a heat sink to the package. P is calculated by using the maximum hot I<sub>CC</sub> and maximum V<sub>CC</sub>.

**Table 6-1. Thermal Resistances (°C/Watt) θ<sub>JC</sub> and θ<sub>JA</sub>**

| Package     | θ <sub>JC</sub> | θ <sub>JA</sub> versus Airflow - ft/min (m/sec) |               |               |               |               |                |
|-------------|-----------------|---|---------------|---------------|---------------|---------------|----------------|
|             |                 | 0<br>(0)  | 200<br>(1.01) | 400<br>(2.03) | 600<br>(3.04) | 800<br>(4.06) | 1000<br>(5.07) |
| 68-Pin PLCC | 8               | 30  | 25            | 20            | 15.5          | 13            | 12             |

**Table 6-2. Maximum T<sub>A</sub> at Various Airflows**

| Package     | T <sub>A</sub> (°C) versus Airflow - ft/min (m/sec) |               |               |               |               |                |
|-------------|---|---------------|---------------|---------------|---------------|----------------|
|             | 0<br>(0)  | 200<br>(1.01) | 400<br>(2.03) | 600<br>(3.04) | 800<br>(4.06) | 1000<br>(5.07) |
| 68-Pin PLCC | 84.9  | 88.3          | 91.8          | 94.8          | 96.6          | 97.2           |

Maximum T<sub>A</sub> is calculated at maximum V<sub>CC</sub> and maximum I<sub>CC</sub>.

**7.0 ELECTRICAL CHARACTERISTICS**

The following specifications represent the targets of the design effort. They are subject to change without notice. Contact your Intel representative to get the most up-to-date values.

**7.1 Absolute Maximum Ratings\***

- Case Temperature T<sub>C</sub> Under Bias . . . 0°C to +100°C
- Storage Temperature . . . . . -65°C to +150°C
- Voltage on Any Pin  
with Respect to Ground . . . . . -0.5 to V<sub>CC</sub> + 0.5
- Power Dissipation . . . . . 0.8W

NOTICE: This is a production data sheet. The specifications are subject to change without notice.

*\*WARNING: Stressing the device beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only. Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.*

## 7.2 D.C. Characteristics

**Table 7-1. D.C. Specifications**  $T_C = 0^\circ\text{C}$  to  $+100^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 10\%$

| Symbol    | Parameter  | Min            | Max                                 | Units                            | Test Conditions   |
|-----------|--|----------------|-------------------------------------|----------------------------------|---|
| $V_{IL}$  | Input LO Voltage   | -0.3           | +0.8                                | V                                | (Note 1)  |
| $V_{IH}$  | Input HI Voltage   | 2.0            | $V_{CC} + 0.3$                      | V                                | (Note 1)  |
| $V_{CL}$  | CPUCLK2 and NUMCLK2<br>Input LO Voltage  | -0.3           | +0.8                                | V                                |   |
| $V_{CH}$  | CPUCLK2 and NUMCLK2<br>Input HI Voltage  | $V_{CC} - 0.8$ | $V_{CC} + 0.8$                      | V                                |   |
| $V_{OL}$  | Output LO Voltage  |                | 0.45                                | V                                | (Note 2)  |
| $V_{OH}$  | Output HI Voltage  | 2.4            |                                     | V                                | (Note 3)  |
| $V_{OH}$  | Output HI Voltage  | $V_{CC} - 0.8$ |                                     | V                                | (Note 4)  |
| $I_{CC}$  | Power Supply Current<br>Dynamic Mode<br>Freq. = 33 MHz <sup>(5)</sup><br>Freq. = 25 MHz <sup>(5)</sup><br>Freq. = 20 MHz <sup>(5)</sup><br>Freq. = 16 MHz <sup>(5)</sup><br>Freq. = 1 MHz <sup>(5)</sup><br>Idle Mode <sup>(6)</sup> |                | 150<br>150<br>125<br>100<br>20<br>7 | mA<br>mA<br>mA<br>mA<br>mA<br>mA | $I_{CC}$ typ. = 135 mA<br>$I_{CC}$ typ. = 130 mA<br>$I_{CC}$ typ. = 110 mA<br>$I_{CC}$ typ. = 90 mA<br>$I_{CC}$ typ. = 5 mA<br>$I_{CC}$ typ. = 4 mA |
| $I_{LI}$  | Input Leakage Current  |                | $\pm 15$                            | $\mu\text{A}$                    | $0\text{V} \leq V_{IN} \leq V_{CC}$   |
| $I_{LO}$  | I/O Leakage Current  |                | $\pm 15$                            | $\mu\text{A}$                    | $0.45\text{V} \leq V_O \leq V_{CC}$   |
| $C_{IN}$  | Input Capacitance  | 7              | 10                                  | pF                               | $f_c = 1\text{ MHz}$  |
| $C_O$     | I/O Capacitance  | 7              | 12                                  | pF                               | $f_c = 1\text{ MHz}$  |
| $C_{CLK}$ | Clock Capacitance  | 7              | 20                                  | pF                               | $f_c = 1\text{ MHz}$  |

### NOTES:

- This parameter is for all inputs, excluding the clock inputs.
- This parameter is measured at  $I_{OL}$  as follows:  
Data = 4.0 mA  
READYO#, ERROR#, BUSY#, PEREQ = 25 mA
- This parameter is measured at  $I_{OH}$  as follows:  
Data = 1.0 mA  
READYO#, ERROR#, BUSY#, PEREQ = 0.6 mA
- This parameter is measured at  $I_{OH}$  as follows:  
Data = 0.2 mA  
READYO#, ERROR#, BUSY#, PEREQ = 0.12 mA
- Synchronous Clock Mode (CKM = 1).  $I_{CC}$  is measured at steady state, maximum capacitive loading on the outputs, and worst-case D.C. level at the inputs.
- Intel387 SX Math CoProcessor Internal Idle Mode. Synchronous clock mode, clock and control inputs are active but the Math CoProcessor is not executing an instruction. Outputs driving CMOS inputs.

**7.3 A.C. Characteristics**
**Table 7-2a. Timing Requirements of the Bus Interface Unit**
**T<sub>C</sub> = 0°C to +100°C, V<sub>CC</sub> = 5V ± 10% (All measurements made at 1.5V unless otherwise specified)**

| Pin          | Symbol | Parameter  | 16 MHz–<br>25 MHz |             | 33 MHz      |             | Test<br>Conditions                 | Refer to<br>Figure |
|--------------|--------|------------|-------------------|-------------|-------------|-------------|------------------------------------|--------------------|
|              |        |            | Min<br>(ns)       | Max<br>(ns) | Min<br>(ns) | Max<br>(ns) |                                    |                    |
| CPUCLK2      | t1     | Period     | 20                | DC          | 15          | DC          | 2.0V                               | 7.2                |
| CPUCLK2      | t2a    | High Time  | 6                 |             | 6.25        |             | 2.0V                               |                    |
| CPUCLK2      | t2b    | High Time  | 3                 |             | 4.5         |             | V <sub>CC</sub> –0.8V              |                    |
| CPUCLK2      | t3a    | Low Time   | 6                 |             | 6.25        |             | 2.0V                               |                    |
| CPUCLK2      | t3b    | Low Time   | 4                 |             | 4.5         |             | 0.8V                               |                    |
| CPUCLK2      | t4     | Fall Time  |                   | 7           |             | 4           | From V <sub>CC</sub> –0.8V to 0.8V |                    |
| CPUCLK2      | t5     | Rise Time  |                   | 7           |             | 4           | From 0.8V to V <sub>CC</sub> –0.8V |                    |
| READYO #     | t7a    | Out Delay  | 4                 | 25          | 4           | 17          | C <sub>L</sub> = 50 pF             | 7.3                |
| PEREQ        | t7b    | Out Delay  | 4                 | 23          | 4           | 21          | C <sub>L</sub> = 50 pF             |                    |
| BUSY #       | t7c    | Out Delay  | 4                 | 23          | 4           | 21          | C <sub>L</sub> = 50 pF             |                    |
| ERROR #      | t7d    | Out Delay  | 4                 | 23          | 4           | 23          | C <sub>L</sub> = 50 pF             |                    |
| D15–D0       | t8     | Out Delay  | 1                 | 45          | 0           | 37          | C <sub>L</sub> = 50 pF             | 7.4                |
| D15–D0       | t10    | Setup Time | 11                |             | 8           |             |                                    |                    |
| D15–D0       | t11    | Hold Time  | 11                |             | 8           |             |                                    |                    |
| D15–D0       | t12*   | Float Time | 6                 | 24          | 6           | 19          |                                    |                    |
| READYO #     | t13a*  | Float Time | 1                 | 40          | 1           | 30          |                                    | 7.6                |
| PEREQ        | t13b*  | Float Time | 1                 | 40          | 1           | 30          |                                    |                    |
| BUSY #       | t13c*  | Float Time | 1                 | 40          | 1           | 30          |                                    |                    |
| ERROR #      | t13d*  | Float Time | 1                 | 40          | 1           | 30          |                                    |                    |
| ADS #        | t14a   | Setup Time | 15                |             | 13          |             |                                    | 7.4                |
| ADS #        | t15a   | Hold Time  | 4                 |             | 4           |             |                                    |                    |
| W/R #        | t14b   | Setup Time | 15                |             | 13          |             |                                    |                    |
| W/R #        | t15b   | Hold Time  | 4                 |             | 4           |             |                                    |                    |
| READY #      | t16a   | Setup Time | 9                 |             | 7           |             |                                    | 7.4                |
| READY #      | t17a   | Hold Time  | 4                 |             | 4           |             |                                    |                    |
| CMD0 #       | t16b   | Setup Time | 16                |             | 13          |             |                                    |                    |
| CMD0 #       | t17b   | Hold Time  | 2                 |             | 2           |             |                                    |                    |
| NPS1 #, NPS2 | t16c   | Setup Time | 16                |             | 13          |             |                                    |                    |
| NPS1 #, NPS2 | t17c   | Hold Time  | 2                 |             | 2           |             |                                    |                    |
| STEN         | t16d   | Setup Time | 15                |             | 13          |             |                                    |                    |
| STEN         | t17d   | Hold Time  | 2                 |             | 2           |             |                                    |                    |
| RESETIN      | t18    | Setup Time | 8                 |             | 5           |             |                                    |                    |
| RESETIN      | t19    | Hold Time  | 3                 |             | 2           |             | 7.5                                |                    |

**NOTE:**

 \*Float condition occurs when maximum output current becomes less than I<sub>LO</sub> in magnitude. Float delay is not tested.

Table 7-2b. Timing Requirements of the Execution Unit (Asynchronous Mode CKM = 0)

| Pin                 | Symbol | Parameter | 16 MHz–<br>25 MHz |             | 33 MHz      |             | Test<br>Conditions           | Refer to<br>Figure |
|---------------------|--------|-----------|-------------------|-------------|-------------|-------------|------------------------------|--------------------|
|                     |        |           | Min<br>(ns)       | Max<br>(ns) | Min<br>(ns) | Max<br>(ns) |                              |                    |
| NUMCLK2             | t1     | Period    | 20                | 500         | 15          | 500         | 2.0V                         | 7.2                |
| NUMCLK2             | t2a    | High Time | 6                 |             | 6.25        |             | 2.0V                         |                    |
| NUMCLK2             | t2b    | High Time | 3                 |             | 4.5         |             | $V_{CC} - 0.8V$              |                    |
| NUMCLK2             | t3a    | Low Time  | 6                 |             | 6.25        |             | 2.0V                         |                    |
| NUMCLK2             | t3b    | Low Time  | 4                 |             | 4.5         |             | 0.8V                         |                    |
| NUMCLK2             | t4     | Fall Time |                   | 7           |             | 6           | From $V_{CC} - 0.8V$ to 0.8V |                    |
| NUMCLK2             | t5     | Rise Time |                   | 7           |             | 6           | From 0.8V to $V_{CC} - 0.8V$ |                    |
| NUMCLK2/<br>CPUCLK2 |        | Ratio     | 10/16             | 14/10       | 10/16       | 14/10       |                              |                    |

**NOTE:**

If not used (CKM = 1) tie NUMCLK2 low.

Table 7-2c. Other A.C. Parameters

| Pin             | Symbol | Parameter  | Min | Max | Units   |
|-----------------|--------|--|-----|-----|---------|
| RESETIN         | t30    | Duration   | 40  |     | NUMCLK2 |
| RESETIN         | t31    | RESETIN Inactive to 1st<br>Opcode Write                      | 50  |     | NUMCLK2 |
| BUSY #          | t32    | Duration   | 6   |     | CPUCLK2 |
| BUSY #, ERROR # | t33    | ERROR # (In)Active to<br>BUSY # Inactive                     | 6   |     | CPUCLK2 |
| PEREQ, ERROR #  | t34    | PEREQ Inactive to<br>ERROR # Active                          | 6   |     | CPUCLK2 |
| READY #, BUSY # | t35    | READY # Active to BUSY #<br>Active                           | 0   | 4   | CPUCLK2 |
| READY #         | t36    | Minimum Time from<br>Opcode Write to<br>Opcode/Operand Write | 4   |     | CPUCLK2 |
| READY #         | t37    | Minimum Time from<br>Operand Write to Operand<br>Write       | 4   |     | CPUCLK2 |

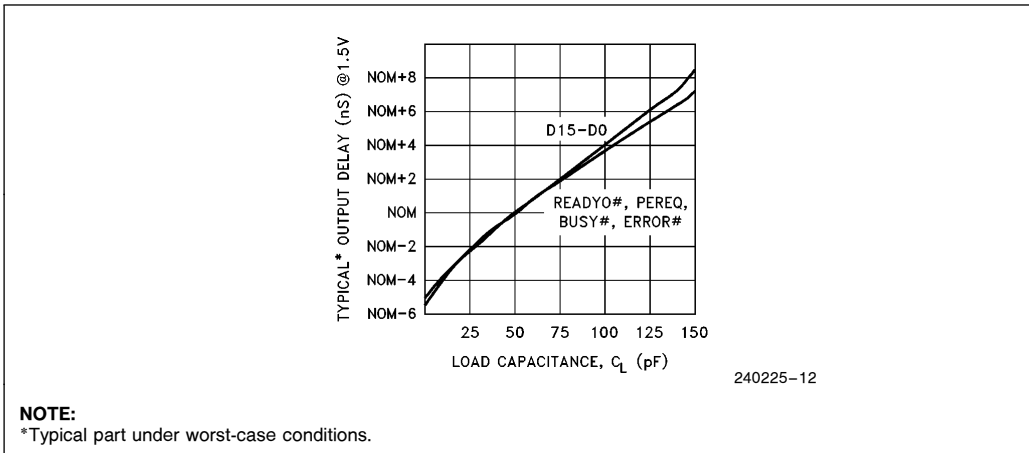


Figure 7-1a. Typical Output Valid Delay vs Load Capacitance at Max Operating Temperature

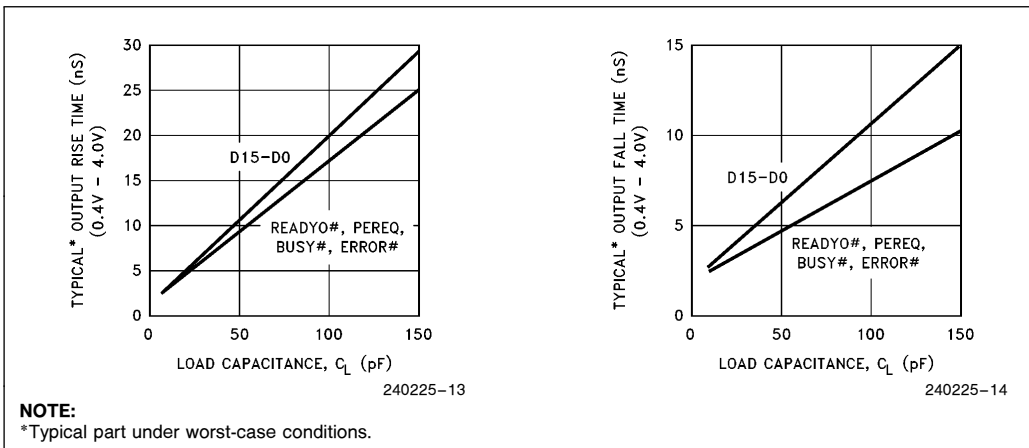


Figure 7-1b. Typical Output Slew Time vs Load Capacitance at Max Operating Temperature

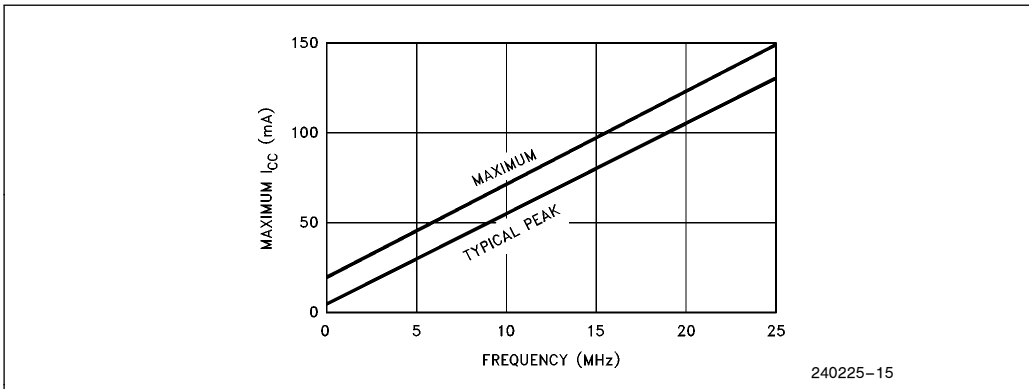


Figure 7-1c. Maximum ICC vs Frequency

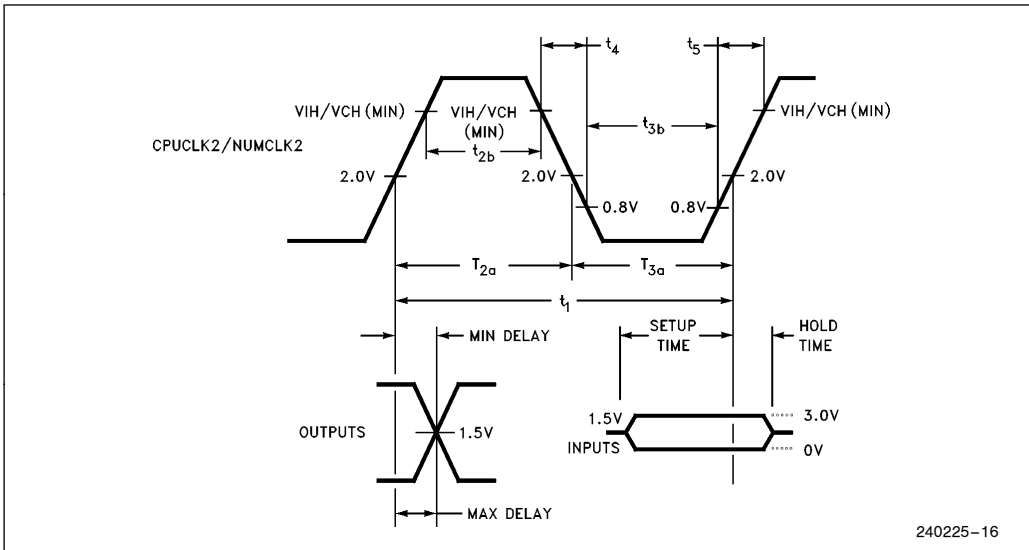


Figure 7-2. CPUCLK2/NUMCLK2 Waveform and Measurement Points for Input/Output

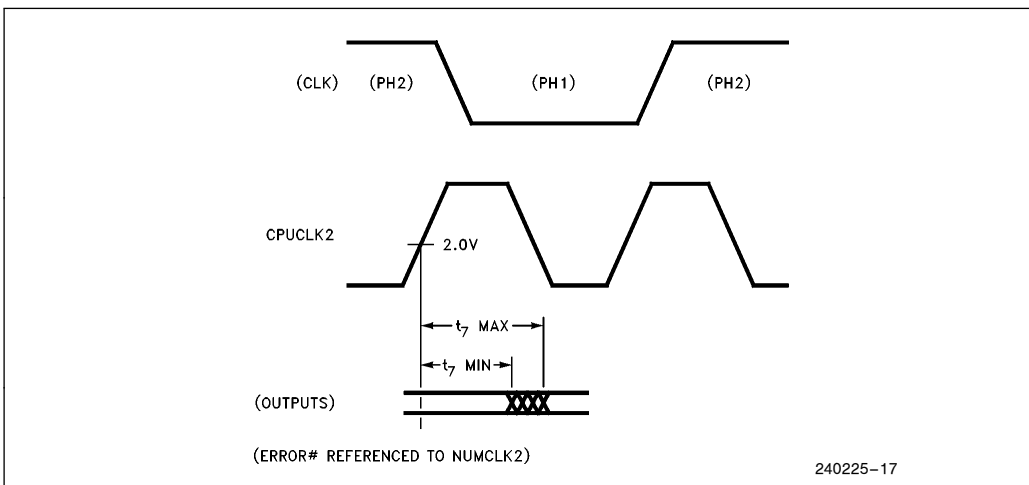


Figure 7-3. Output Signals

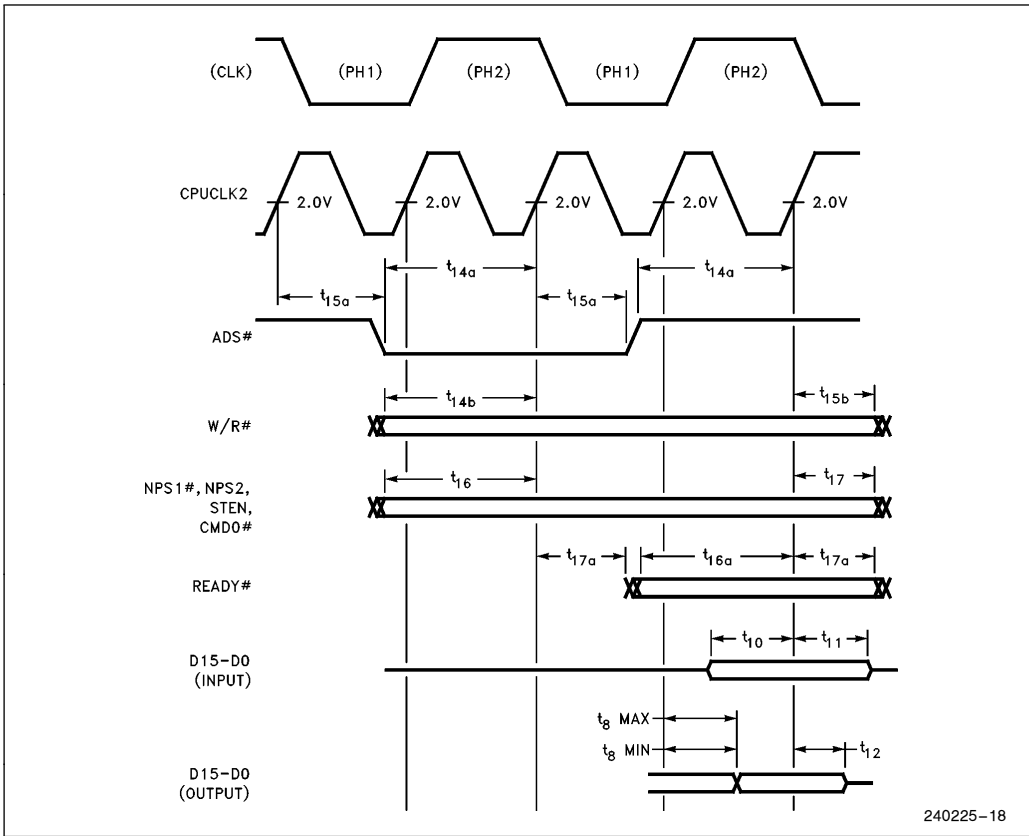
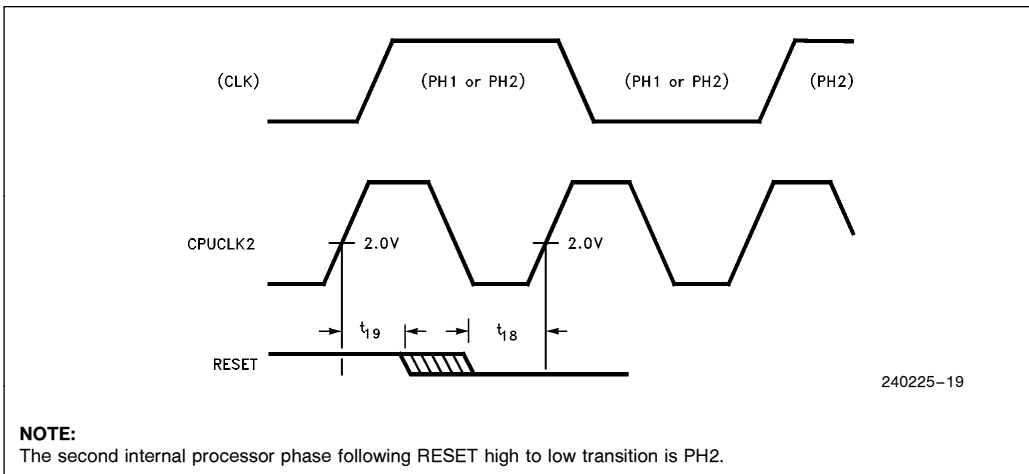


Figure 7-4. Input and I/O Signals



**NOTE:**  
The second internal processor phase following RESET high to low transition is PH2.

Figure 7-5. RESET Signal

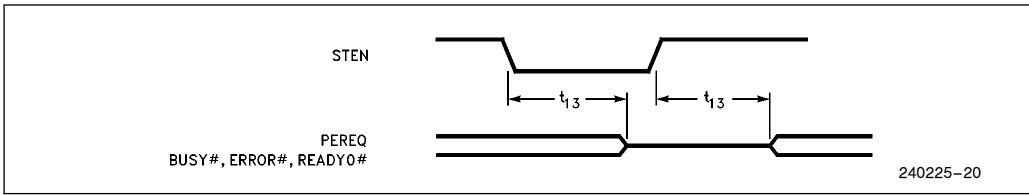


Figure 7-6. Float from STEN

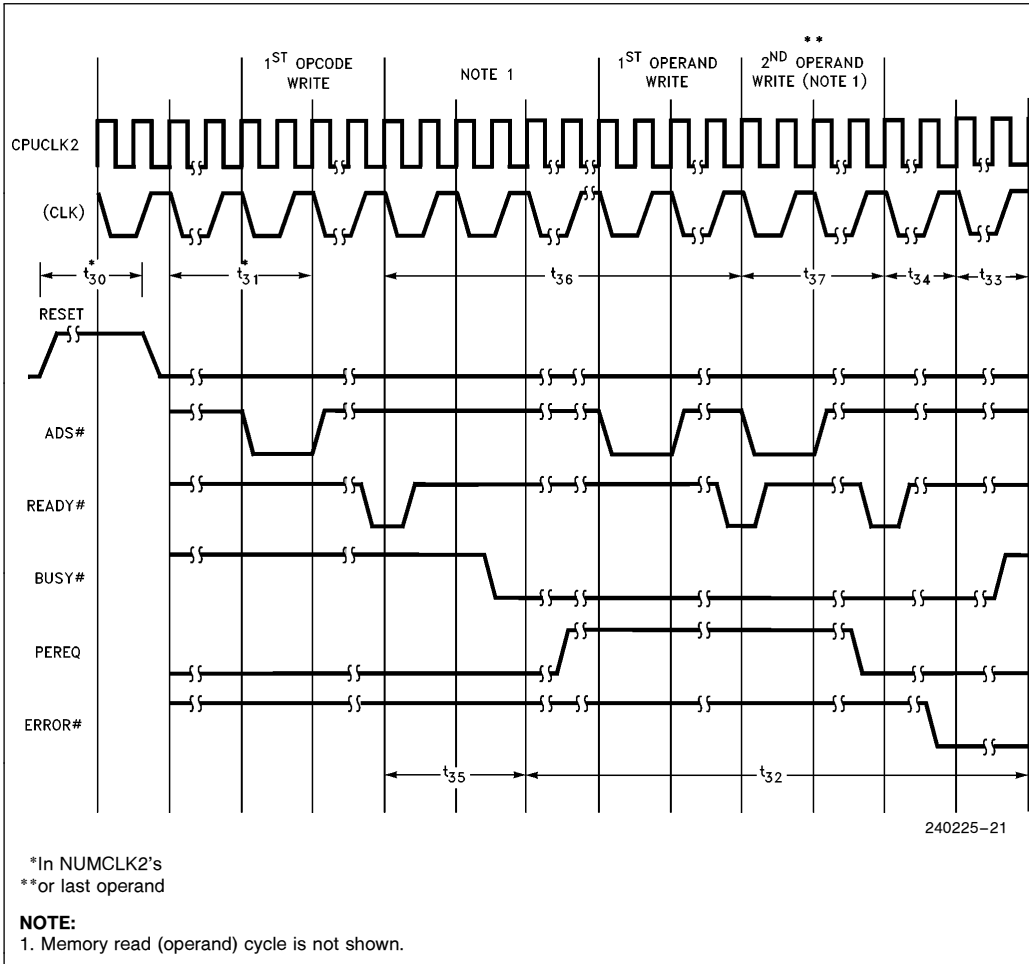


Figure 7-7. Other Parameters





### 8.0 INTEL387 SX MATH COPROCESSOR INSTRUCTION SET

Instructions for the Intel387 SX Math CoProcessor assume one of the five forms shown in Table 8-1. In all cases, instructions are at least two bytes long and begin with the bit pattern 11011B, which identifies the ESCAPE class of instruction. Instructions that refer to memory operands specify addresses using the CPU's addressing modes.

MOD (Mode field) and R/M (Register/Memory specifier) have the same interpretation as the corresponding fields of CPU instructions (refer to Pro-

grammer's Reference Manual for the CPU). SIB (Scale Index Base) byte and DISP (displacement) are optionally present in instructions that have MOD and R/M fields. Their presence depends on the values of MOD and R/M, as for instructions of the CPU.

The instruction summaries that follow in Table 8-2 assume that the instruction has been prefetched, decoded, and is ready for execution; that bus cycles do not require wait states; that there are no local bus HOLD requests delaying processor access to the bus; and that no exceptions are detected during instruction execution. If the instruction has MOD and R/M fields that call for both base and index registers, add one clock.

Table 8-1. Instruction Formats

|   |       | Instruction |    |     |             |      |      |       | Optional Fields |         |
|---|-------|-------------|----|-----|-------------|------|------|-------|-----------------|---------|
|   |       | First Byte  |    |     | Second Byte |      |      |       |                 |         |
| 1 | 11011 | OPA         |    | 1   | MOD         | 1    | OPB  | R/M   | SIB             | DISP    |
| 2 | 11011 | MF          |    | OPA | MOD         | OPB* |      | R/M   | SIB             | DISP    |
| 3 | 11011 | d           | P  | OPA | 1           | 1    | OPB* | ST(i) |                 |         |
| 4 | 11011 | 0           | 0  | 1   | 1           | 1    | 1    | OP    |                 |         |
| 5 | 11011 | 0           | 1  | 1   | 1           | 1    | 1    | OP    |                 |         |
|   |       | 15-11       | 10 | 9   | 8           | 7    | 6    | 5     | 4               | 3 2 1 0 |

- OP = Instruction opcode, possibly split into two fields OPA and OPB
- MF = Memory Format
  - 00 - 32-bit real
  - 01 - 32-bit integer
  - 10 - 64-bit real
  - 11 - 16-bit integer
- d = Destination
  - 0 - Destination is ST(0)
  - 1 - Destination is ST(i)
- R XOR d = 0 - Destination (op) Source
- R XOR d = 1 - Source (op) Destination
- \*In FSUB and FDIV, the low-order bit of OPB is the R (reversed) bit
- P = POP
  - 0 - Do not pop stack
  - 1 - Pop stack after operation
- ESC = 11011
- ST(i) = Register stack element i
  - 000 = Stack top
  - 001 = Second stack element
  - 
  - 
  - 
  - 111 = Eighth stack element



| Instruction                                      | Encoding |             |                    | Clock Count Range |                |             |                |
|--|----------|-------------|--------------------|-------------------|----------------|-------------|----------------|
|  | Byte 0   | Byte 1      | Optional Bytes 2-6 | 32-Bit Real       | 32-Bit Integer | 64-Bit Real | 16-Bit Integer |
| <b>DATA TRANSFER</b>                             |          |             |                    |                   |                |             |                |
| <b>FLD = Load<sup>a</sup></b>                    |          |             |                    |                   |                |             |                |
| Integer/real memory to ST(0)                     | ESC MF 1 | MOD 000 R/M | SIB/DISP           | 11-20             | 28-44          | 20-27       | 42-53          |
| Long integer memory to ST(0)                     | ESC 111  | MOD 101 R/M | SIB/DISP           |                   |                | 30-58       |                |
| Extended real memory to ST(0)                    | ESC 011  | MOD 101 R/M | SIB/DISP           |                   |                | 16-47       |                |
| BCD memory to ST(0)                              | ESC 111  | MOD 100 R/M | SIB/DISP           |                   |                | 49-101      |                |
| ST(i) to ST(0)                                   | ESC 001  | 11000 ST(i) |                    |                   |                | 7-12        |                |
| <b>FST = Store</b>                               |          |             |                    |                   |                |             |                |
| ST(0) to integer/real memory                     | ESC MF 1 | MOD 010 R/M | SIB/DISP           | 27-45             | 59-78          | 59          | 58-76          |
| ST(0) to ST(i)                                   | ESC 101  | 11010 ST(i) |                    |                   |                | 7-11        |                |
| <b>FSTP = Store and Pop</b>                      |          |             |                    |                   |                |             |                |
| ST(0) to integer/real memory                     | ESC MF 1 | MOD 011 R/M | SIB/DISP           | 27-45             | 59-78          | 59          | 58-76          |
| ST(0) to long integer memory                     | ESC 111  | MOD 111 R/M | SIB/DISP           |                   |                | 64-86       |                |
| ST(0) to extended real memory                    | ESC 011  | MOD 111 R/M | SIB/DISP           |                   |                | 50-56       |                |
| ST(0) to BCD memory                              | ESC 111  | MOD 110 R/M | SIB/DISP           |                   |                | 116-194     |                |
| ST(0) to ST(i)                                   | ESC 101  | 11011 ST(i) |                    |                   |                | 7-11        |                |
| <b>FXCH = Exchange</b>                           |          |             |                    |                   |                |             |                |
| ST(i) and ST(0)                                  | ESC 001  | 11001 ST(i) |                    |                   |                | 10-17       |                |
| <b>COMPARISON</b>                                |          |             |                    |                   |                |             |                |
| <b>FCOM = Compare</b>                            |          |             |                    |                   |                |             |                |
| Integer/real memory to ST(0)                     | ESC MF 0 | MOD 010 R/M | SIB/DISP           | 15-27             | 36-54          | 18-31       | 39-62          |
| ST(i) to ST(0)                                   | ESC 000  | 11010 ST(i) |                    |                   |                | 13-21       |                |
| <b>FCOMP = Compare and pop</b>                   |          |             |                    |                   |                |             |                |
| Integer/real memory to ST(0)                     | ESC MF 0 | MOD 011 R/M | SIB/DISP           | 15-27             | 36-54          | 18-31       | 39-62          |
| ST(i) to ST(0)                                   | ESC 000  | 11011 ST(i) |                    |                   |                | 13-21       |                |
| <b>FCOMPP = Compare and pop twice</b>            |          |             |                    |                   |                |             |                |
| ST(1) to ST(0)                                   | ESC 110  | 1101 1001   |                    |                   |                | 13-21       |                |
| <b>FTST = Test ST(0)</b>                         |          |             |                    |                   |                |             |                |
| ST(0)  | ESC 001  | 1110 0100   |                    |                   |                | 17-25       |                |
| <b>FUCOM = Unordered compare</b>                 |          |             |                    |                   |                |             |                |
| ST(0)  | ESC 101  | 11100 ST(i) |                    |                   |                | 13-21       |                |
| <b>FUCOMP = Unordered compare and pop</b>        |          |             |                    |                   |                |             |                |
| ST(0)  | ESC 101  | 11101 ST(i) |                    |                   |                | 13-21       |                |
| <b>FUCOMPP = Unordered compare and pop twice</b> |          |             |                    |                   |                |             |                |
| ST(0)  | ESC 010  | 1110 1001   |                    |                   |                | 13-21       |                |
| <b>FXAM = Examine ST(0)</b>                      |          |             |                    |                   |                |             |                |
| ST(0)  | ESC 001  | 1110 0101   |                    |                   |                | 24-37       |                |

Shaded areas indicate instructions not available in 8087/80287.

**NOTE:**

a. When loading single or double precision zero from memory, add 5 clocks.



| Instruction  | Encoding  |               |                    | Clock Count Range |                      |             |                      |
|--|-----------|---------------|--------------------|-------------------|----------------------|-------------|----------------------|
|  | Byte 0    | Byte 1        | Optional Bytes 2-6 | 32-Bit Real       | 32-Bit Integer       | 64-Bit Real | 16-Bit Integer       |
| <b>ARITHMETIC</b>  |           |               |                    |                   |                      |             |                      |
| <b>FADD</b> = Add  |           |               |                    |                   |                      |             |                      |
| Integer/real memory to ST(0)   | ESC MF 0  | MOD 000 R/R/M | SIB/DISP           | 14-31             | 36-58                | 19-38       | 38-64                |
| ST(i) and ST(0)  | ESC d P 0 | 11000 ST(i)   | SIB/DISP           |                   | 12-26 <sup>b</sup>   |             |                      |
| <b>FSUB</b> = Subtract   |           |               |                    |                   |                      |             |                      |
| Integer/real memory with ST(0)                                       | ESC MF 0  | MOD 10 R R/M  | SIB/DISP           | 14-31             | 36-58                | 19-38       | 38-64 <sup>c</sup>   |
| ST(i) to ST(0)   | ESC d P 0 | 1110 R R/M    |                    |                   | 12-26 <sup>d</sup>   |             |                      |
| <b>FMUL</b> = Multiply   |           |               |                    |                   |                      |             |                      |
| Integer/real memory with ST(0)                                       | ESC MF 0  | MOD 001 R/R/M | SIB/DISP           | 21-33             | 45-73                | 27-57       | 46-74                |
| ST(i) and ST(0)  | ESC d P 0 | 1100 1 R/M    |                    |                   | 17-50 <sup>e</sup>   |             |                      |
| <b>FDIV</b> = Divide   |           |               |                    |                   |                      |             |                      |
| Integer/real memory with ST(0)                                       | ESC MF 0  | MOD 11 R R/M  | SIB/DISP           | 79-87             | 103-116 <sup>f</sup> | 85-95       | 105-124 <sup>g</sup> |
| ST(i) and ST(0)  | ESC d P 0 | 1111 R R/M    |                    |                   | 77-80 <sup>h</sup>   |             |                      |
| <b>FSQRT</b> <sup>i</sup> = Square root                              | ESC 001   | 1111 1010     |                    |                   | 97-111               |             |                      |
| <b>FSCALE</b> = Scale ST(0) by ST(1)                                 | ESC 001   | 1111 1101     |                    |                   | 44-82                |             |                      |
| <b>FPREM</b> = Partial remainder                                     | ESC 001   | 1111 1000     |                    |                   | 56-140               |             |                      |
| <b>FPREM1</b> = Partial remainder (IEEE)                             | ESC 001   | 1111 0101     |                    |                   | 81-168               |             |                      |
| <b>FRNDINT</b> = Round ST(0) to integer                              | ESC 001   | 1111 1100     |                    |                   | 41-62                |             |                      |
| <b>FTRACT</b> = Extract components of ST(0)                          | ESC 001   | 1111 0100     |                    |                   | 42-63                |             |                      |
| <b>FABS</b> = Absolute value of ST(0)                                | ESC 001   | 1110 0001     |                    |                   | 14-21                |             |                      |
| <b>FCHS</b> = Change sign of ST(0)                                   | ESC 001   | 1110 0000     |                    |                   | 17-24                |             |                      |
| <b>TRANSCENDENTAL</b>  |           |               |                    |                   |                      |             |                      |
| <b>FCOS</b> <sup>k</sup> = Cosine of ST(0)                           | ESC 001   | 1111 1111     |                    |                   | 122-680              |             |                      |
| <b>FPTAN</b> <sup>k</sup> = Partial tangent of ST(0)                 | ESC 001   | 1111 0010     |                    |                   | 162-430 <sup>j</sup> |             |                      |
| <b>FPATAN</b> = Partial arctangent of ST(0)                          | ESC 001   | 1111 0011     |                    |                   | 250-420              |             |                      |
| <b>FSIN</b> <sup>k</sup> = Sine of ST(0)                             | ESC 001   | 1111 1110     |                    |                   | 121-680              |             |                      |
| <b>FSINCOS</b> <sup>k</sup> = Sine and cosine of ST(0)               | ESC 001   | 1111 1011     |                    |                   | 150-650              |             |                      |
| <b>F2XM1</b> <sup>l</sup> = 2 <sup>ST(0)</sup> - 1                   | ESC 001   | 1111 0000     |                    |                   | 167-410              |             |                      |
| <b>FYL2XM</b> <sup>m</sup> = ST(1) * log <sub>2</sub> ST(0)          | ESC 001   | 1111 0001     |                    |                   | 99-436               |             |                      |
| <b>FYL2XP1</b> <sup>n</sup> = ST(1) * log <sub>2</sub> [ST(0) + 1.0] | ESC 001   | 1111 1001     |                    |                   | 210-447              |             |                      |

Shaded areas indicate instructions not available in 8087/80287.

**NOTES:**

- b. Add 3 clocks to the range when d = 1.
- c. Add 1 clock to each range when R = 1.
- d. Add 3 clocks to the range when d = 0.
- e. typical = 52 (When d = 0, 46-54, typical = 49).
- f. Add 1 clock to the range when R = 1.
- g. 135-141 when R = 1.
- h. Add 3 clocks to the range when d = 1.
- i. -0 ≤ ST(0) ≤ +∞.
- j. These timings hold for operands in the range |x| < π. For operands not in this range, up to 76 additional clocks may be needed to reduce the operand.
- k. 0 ≤ ST(0) < 2<sup>63</sup>.
- l. -1.0 ≤ ST(0) ≤ 1.0.
- m. 0 ≤ ST(0) < ∞, -∞ < ST(1) < +∞.
- n. 0 ≤ |ST(0)| < [2-SQRT(2)]/2, -∞ < ST(1) < +∞.

| Instruction                                  | Encoding |              |                    | Clock Count Range |                |             |                |
|--|----------|--------------|--------------------|-------------------|----------------|-------------|----------------|
|  | Byte 0   | Byte 1       | Optional Bytes 2-6 | 32-Bit Real       | 32-Bit Integer | 64-Bit Real | 16-Bit Integer |
| <b>CONSTANTS</b>                             |          |              |                    |                   |                |             |                |
| <b>FLDZ</b> = Load +0.0 to ST(0)             | ESC 001  | 1110 1110    |                    |                   |                |             | 10-17          |
| <b>FLD1</b> = Load +1.0 to ST(0)             | ESC 001  | 1110 1000    |                    |                   |                |             | 15-22          |
| <b>FLDPI</b> = Load $\pi$ to ST(0)           | ESC 001  | 1110 1011    |                    |                   |                |             | 26-36          |
| <b>FLDL2T</b> = Load $\log_2(10)$ to ST(0)   | ESC 001  | 1110 1001    |                    |                   |                |             | 26-36          |
| <b>FLDL2E</b> = Load $\log_2(e)$ to ST(0)    | ESC 001  | 1110 1010    |                    |                   |                |             | 26-36          |
| <b>FLDLG2</b> = Load $\log_{10}(2)$ to ST(0) | ESC 001  | 1110 1100    |                    |                   |                |             | 25-35          |
| <b>FLDLN2</b> = Load $\log_e(2)$ to ST(0)    | ESC 001  | 1110 1101    |                    |                   |                |             | 26-38          |
| <b>PROCESSOR CONTROL</b>                     |          |              |                    |                   |                |             |                |
| <b>FINIT</b> = Initialize Math CoProcessor   | ESC 011  | 1110 0011    |                    |                   |                |             | 33             |
| <b>FLDCW</b> = Load control word from memory | ESC 001  | MOD 101 R/M  | SIB/DISP           |                   |                |             | 19             |
| <b>FSTCW</b> = Store control word to memory  | ESC 001  | MOD 111 R/M  | SIB/DISP           |                   |                |             | 15             |
| <b>FSTSW</b> = Store status word to memory   | ESC 101  | MOD 111 R/M  | SIB/DISP           |                   |                |             | 15             |
| <b>FSTSW AX</b> = Store status word to AX    | ESC 111  | 1110 0000    |                    |                   |                |             | 13             |
| <b>FCLEX</b> = Clear exceptions              | ESC 011  | 1110 0010    |                    |                   |                |             | 11             |
| <b>FSTENV</b> = Store environment            | ESC 001  | MOD 110 R/M  | SIB/DISP           |                   |                |             | 117-118        |
| <b>FLDENV</b> = Load environment             | ESC 001  | MOD 100 R/M  | SIB/DISP           |                   |                |             | 85             |
| <b>FSAVE</b> = Save state                    | ESC 101  | MOD 110 R/M  | SIB/DISP           |                   |                |             | 402-403        |
| <b>FRSTOR</b> = Restore state                | ESC 101  | MOD 100 R/M  | SIB/DISP           |                   |                |             | 415            |
| <b>FINCSTP</b> = Increment stack pointer     | ESC 001  | 1111 0111    |                    |                   |                |             | 21             |
| <b>FDECSTP</b> = Decrement stack pointer     | ESC 001  | 1111 0110    |                    |                   |                |             | 22             |
| <b>FFREE</b> = Free ST(i)                    | ESC 101  | 1100 0 ST(i) |                    |                   |                |             | 18             |
| <b>FNOP</b> = No operations                  | ESC 001  | 1101 0000    |                    |                   |                |             | 12             |



## APPENDIX A INTEL387 SX MATH COPROCESSOR COMPATIBILITY

### A.1 8087/80287 Compatibility

This section summarizes the differences between the Intel387 SX Math CoProcessor and the 80287 Math CoProcessor. Any migration from the 8087 directly to the Intel387 SX Math CoProcessor must also take into account the differences between the 8087 and the 80287 Math CoProcessor as listed in Appendix B.

Many changes have been designed into the Intel387 SX Math CoProcessor to directly support the IEEE standard in hardware. These changes result in increased performance by eliminating the need for software that supports the standard.

#### A.1.1 GENERAL DIFFERENCES

The Intel387 SX Math CoProcessor supports only affine closure for infinity arithmetic, not projective closure.

Operands for FSCALE and FPATAN are no longer restricted in range (except for  $\pm \infty$ ); F2XM1 and FPTAN accept a wider range of operands.

Rounding control is in effect for FLD constant.

Software cannot change entries of the tag word to values (other than empty) that differ from actual register contents.

After reset, FINIT, and incomplete FPREM, the Intel387 SX Math CoProcessor resets to zero the condition code bits  $C_3$ – $C_0$  of the status word.

In conformance with the IEEE standard, the Intel387 SX Math CoProcessor does not support the special data formats pseudo-zero, pseudo-NaN, pseudo-infinity, and unnormal.

The denormal exception has a different purpose on the Intel387 SX Math CoProcessor. A system that uses the denormal exception handler solely to normalize the denormal operands, would better mask the denormal exception on the Intel387 SX Math CoProcessor. The Intel387 SX Math CoProcessor automatically normalizes denormal operands when the denormal exception is masked.

### A.1.2 EXCEPTIONS

A number of differences exist due to changes in the IEEE standard and to functional improvements to the architecture of the Intel387 SX Math CoProcessor:

1. When the overflow or underflow exception is masked, the Intel387 SX Math CoProcessor differs from the 80287 in rounding when overflow or underflow occurs. The Intel387 SX Math CoProcessor produces results that are consistent with the rounding mode.
2. When the underflow exception is masked, the Intel387 SX Math CoProcessor sets its underflow flag only if there is also a loss of accuracy during denormalization.
3. Fewer invalid-operations exceptions due to denormal operand, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.
4. The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
5. The denormal exception can occur during the transcendental instruction and the FEXTRACT instruction.
6. The denormal exception no longer takes precedence over all other exceptions.
7. When the denormal exception is masked, the Intel387 SX Math CoProcessor automatically normalizes denormal operands. The 8087/80287 performs unnormal arithmetic, which might produce an unnormal result.
8. When the operand is zero, the FEXTRACT instruction reports a zero-divide exception and leaves  $-\infty$  in ST(1).
9. The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
10. FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
11. FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormal operand exception. When loading a signaling NaN, FLD *single/double precision* signals an invalid-operation exception.
12. The Intel387 SX Math CoProcessor only generates quiet NaNs (as on the 80287); however, the Intel387 SX Math CoProcessor distinguishes between quiet NaNs and signaling NaNs. Signaling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP which also raise IE for quiet NaNs).
13. When stack overflow occurs during FPTAN and overflow is masked, both ST(0) and ST(1) contain quiet NaNs. The 80287/8087 leaves the original operand in ST(1) intact.
14. When the scaling factor is  $\pm\infty$ , the FSCALE instruction behaves as follows:
  - FSCALE (0,  $\infty$ ) generates the invalid operation exception.
  - FSCALE (finite,  $-\infty$ ) generates zero with the same sign as the scaled operand.
  - FSCALE (finite,  $+\infty$ ) generates  $\infty$  with the same sign as the scaled operand.
 The 8087/80287 returns zero in the first case and raises the invalid-operation exception in the other cases.
15. The Intel387 SX Math CoProcessor returns signed infinity/zero as the unmasked response to massive overflow/underflow. The 8087 and 80287 support a limited range for the scaling factor; within this range either massive overflow/underflow do not occur or undefined results are produced.

## APPENDIX B

# COMPATIBILITY BETWEEN THE 80287 AND 8087 MATH COPROCESSOR

The 80286/80287 operating in Real Address mode will execute 8086/8087 programs without major modification. However, because of differences in the handling of numeric exceptions by the 80287 Math CoProcessor and the 8087 Math CoProcessor, exception handling routines *may* need to be changed. This appendix summarizes the differences between the 80287 Math CoProcessor and the 8087 Math CoProcessor, and provides details showing how 8087/8087 programs can be ported to the 80286/80287.

1. The Math CoProcessor signals exceptions through a dedicated ERROR# line to the 80286. The Math CoProcessor error signal does not pass through an interrupt controller (the 8087 INT signal does). Therefore, any interrupt controller oriented instructions in numeric exception handlers for the 8086/8087 should be deleted.
2. The 8087 instructions FENI and FDISI perform no useful function in the 80287. If the 80287 encounters one of these opcodes in its instruction stream, the instruction will effectively be ignored; none of the 80287 internal states will be updated. While 8086/8087 programs containing the instruction may be executed on the 80286/80287, it is unlikely that the exception handling routines containing these instructions will be completely portable to the 80287.
3. Interrupt vector 16 must point to the numeric exception handling routine.
4. The ESC instruction address saved in the 80287 includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.
5. In Protected Address mode, the format of the 80287's saved instruction and address pointers is different than for the 8087. The instruction opcode is not saved in Protected mode; exception handlers will have to retrieve the opcode from memory if needed.
6. Interrupt 7 will occur in the 80286 when executing ESC instructions with either TS (task switched) or EM (emulation) of the 80286 MSW set (TS = 1 or EM = 1). If TS is set, then a WAIT instruction will also cause interrupt 7. An exception handler should be included in 80286/80287 code to handle these situations.
7. Interrupt 9 will occur if the second or subsequent words of a floating point operand fall outside a segment's size. Interrupt 13 will occur if the starting address of a numeric operand falls outside a segment's size. An exception handler should be included in 80286/80287 code to report these programming errors.
8. Except for the processor control instructions, all of the 80287 numeric instructions are automatically synchronized by the 80286 CPU; the 80286 CPU automatically tests the BUSY# line from the 80287 to ensure that the 80287 has completed its previous instruction before executing the next ESC instruction. No explicit WAIT instructions are required to assure this synchronization. For the 8087 used with 8086 and 8088 processors, explicit WAITs are required before each numeric instruction to ensure synchronization. Although 8086/8087 programs having explicit WAIT instructions will execute perfectly on the 80286/80287 without reassembly, these WAIT instructions are unnecessary.
9. Since the 80287 does not require WAIT instructions before each numeric instruction, the ASM286 assembler does not automatically generate these WAIT instructions. The ASM86 assembler, however, automatically precedes every ESC instruction with a WAIT instruction. Although numeric routines generated using the ASM86 assembler will generally execute correctly on the 80286/80287, reassembly using ASM286 may result in a more compact code image.

The processor control instructions for the 80287 may be coded using either a WAIT or No-WAIT form of mnemonic. The WAIT forms of these instructions cause ASM286 to precede the ESC instructions with a CPU WAIT instruction, in the identical manner as does ASM86.