The 80x87 Math Coprocessor

Why we need a math coprocessor?

- Using a general-purpose microprocessor such as the 8088/86 to perform mathematical functions such as log, sine, and others is very time consuming, not only for the CPU but also for programmers writing such programs.

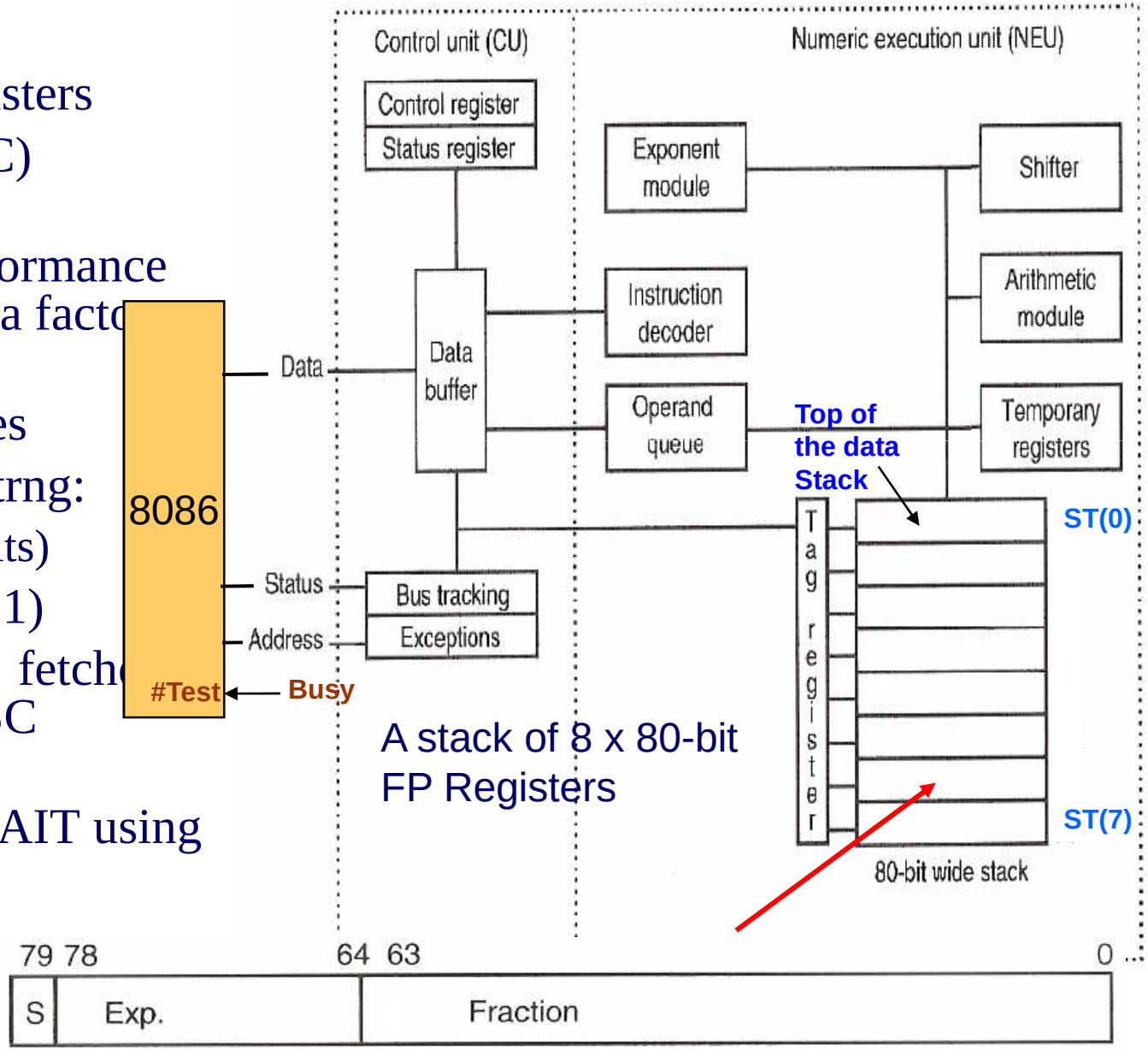
- In the absence of a math coprocessor, programmers must write subroutines using 8088/86 instructions for mathematical functions.

The Math Coprocessor: (Numeric Data Processor (NDP))

- The 8086 performs integer math operations
- Floating point operations are needed, e.g. for Sqrt (X), sin (x), etc.
- These are complex math operations that require large registers, complex circuits, and large areas on the chip
- A general data processor avoids this much burden and delegates such operations to a processor designed specifically for this purpose -
- e.g. math coprocessor (8087) for the 8086
- The 8086 and the 8087 coprocessors operate in parallel and share the busses and memory resources
- The 8086 marks floating point operations as ESC instructions, will ignore them and 8087 will pick them up and execute them

The 8087 Coprocessor: Organization

- CU and NEU units
- Eight 80-bit FP Registers
- Supports 68 FP (ESC) instructions
- Speeds up 8086 performance on FP operations by a facto of 50-100 time
- 8087 Tracks activities of the 8086 by monitrng:
  - Bus status (S0-S2 bits)
  - Queue status (QS0,1)
  - Instruction being fetch (to check if its an ESC instruction)
- Synchronize with WAIT using the BUSY-#TEST signals



8086 Maximum mode outputs for NDP Connection

- **Bus Status Outputs S0-S2:**
  Status bits that encode the type of the current bus cycle

Table 9-7

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	*Function*
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Opcode fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

- **Bus Request/Grant Outputs RQ0/GT0:**
  Allow 8087 to request use of the bus, e.g. for DMA memory access

- **Queue Status Outputs QS1,QS0:**

  - For use by coprocessors that receive their instructions via ESC prefix.

  - Allow the coprocessor to track the progress of an instruction through the 8086 queue and help it determine when to access the bus for the escape op-code and operand.

  - Indicate the status of the internal instruction queue as given in the table:

QS1	QS0	
0	0	Queue is idle
0	1	First byte of opcode from queue
1	0	Queue is empty
1	1	Subsequent byte of opcode from queue

8087 pin diagrams

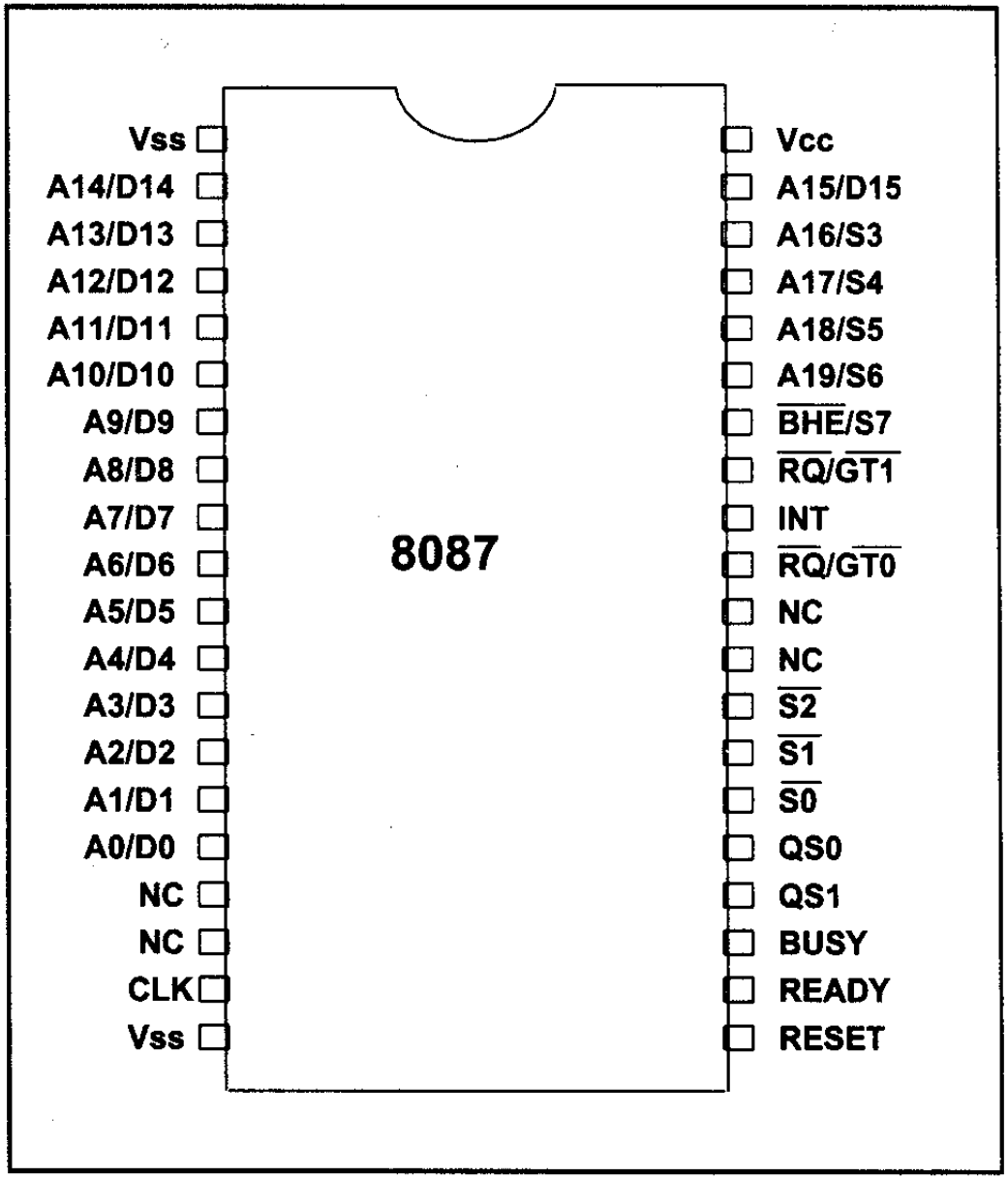


Figure 20-3. 8087 Pin Diagrams

The following is description of the signal connection.

1. The 8086 and 8087 receive the same signals, CLK, READY, and RESET, from the 8284. This ensures that they are synchronized.
2. S0, S1, and S2 are going from the 8086 or 8087 to the 8288, which allows either of these two processors to provide the status signal to the 8288.
3. The Queue Status, QS1 and QS0, from the 8089 go to the 8087, allowing it to know the status of the queue of the 8088 at any given time.
4. The TEST signal to the 8086 comes from BUSY of the 8087.
   By deactivating (going low) the BUSY signal, the 8087 informs the 8086 that it finished execution of the instruction which it has been WAITing for.
5. RQ/GT1 (request/grant) of the 8088 is connected to RQ/GTO of the 8087, allowing them to arbitrate mastery over the buses.
   There are two sets of RQ/GT: RQ/GT1 and RQ/GTO . RQ/GT1 of the 8087 is not used and is connected to Vcc permanently.
   This extra RQ/GT is provided in case there is a third microprocessor connected to the local bus.

The following is description of the signal connection.

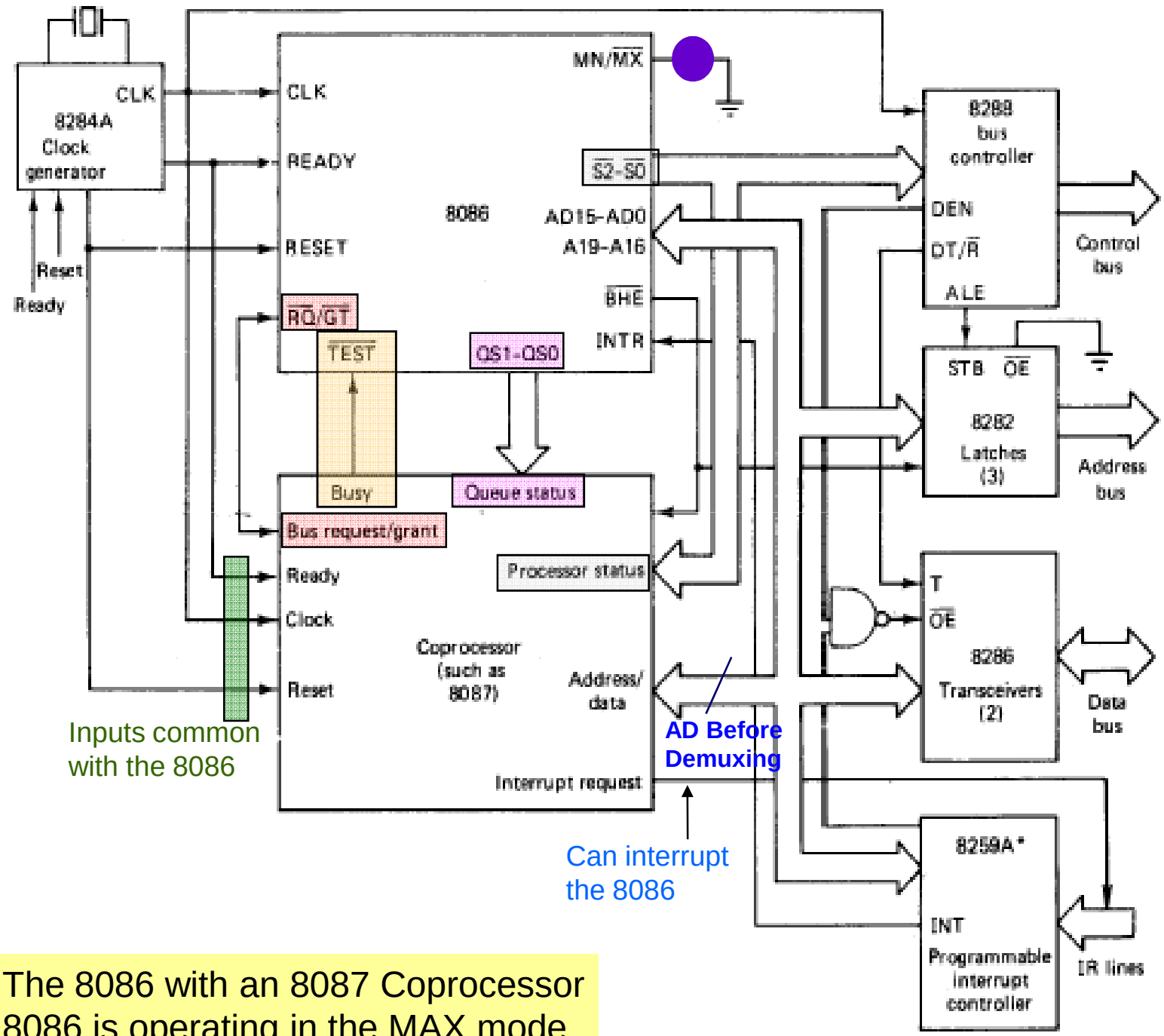
6. Both the 8086 and 8087 share buses ADO -AD7 and A8 -A19, allowing either one to access memory.
   Since the 8087 is designed for both the 8088 and 8086, signal BHE is provided for the 8086 processor.
   If the microprocessor used was an 8086, BHE from the 8086 is connected to BHE of the 8087.
7. INT of the 8087 is an output signal indicating error conditions, also called exceptions, such as divide by zero. Error conditions are given in the status word. Assuming the bit for that error is not masked and an interrupt is enabled, whenever any of these errors occurs, the 8087 automatically activates the INT pin by putting high on it.
8. The 8088, often called the host processor, must be connected in maximum mode to be able to accommodate a coprocessor such as the 8087.

How the 8088 and 8087 work together in the IBM PC/XT

- Each gets a copy of the instructions as they are fetched from memory.
- Since all the instructions of the 8087 have 9BH in the most significant byte of the opcode, the 8088/86 ignores these instructions.
- In reality, 9BH is the opcode for the 8088/86 ESCAPE instruction.
- Likewise, the 8087 ignores any opcode that lacks 9BH.
- It must be made clear that although both receive a copy of each fetched opcode, only the 8088/86 can fetch opcodes since it is the only device that has the instruction pointer.
- Now one might ask how the 8088/86 makes sure it is not flooding the 8087 by fetching instructions for the coprocessor faster than the 8087 can process them.
- The first rule of working together is that the 8088/86 cannot fetch another 8087 instruction until the 8087 has finished execution of the present instruction.

How the 8088 and 8087 work together in the IBM PC/XT

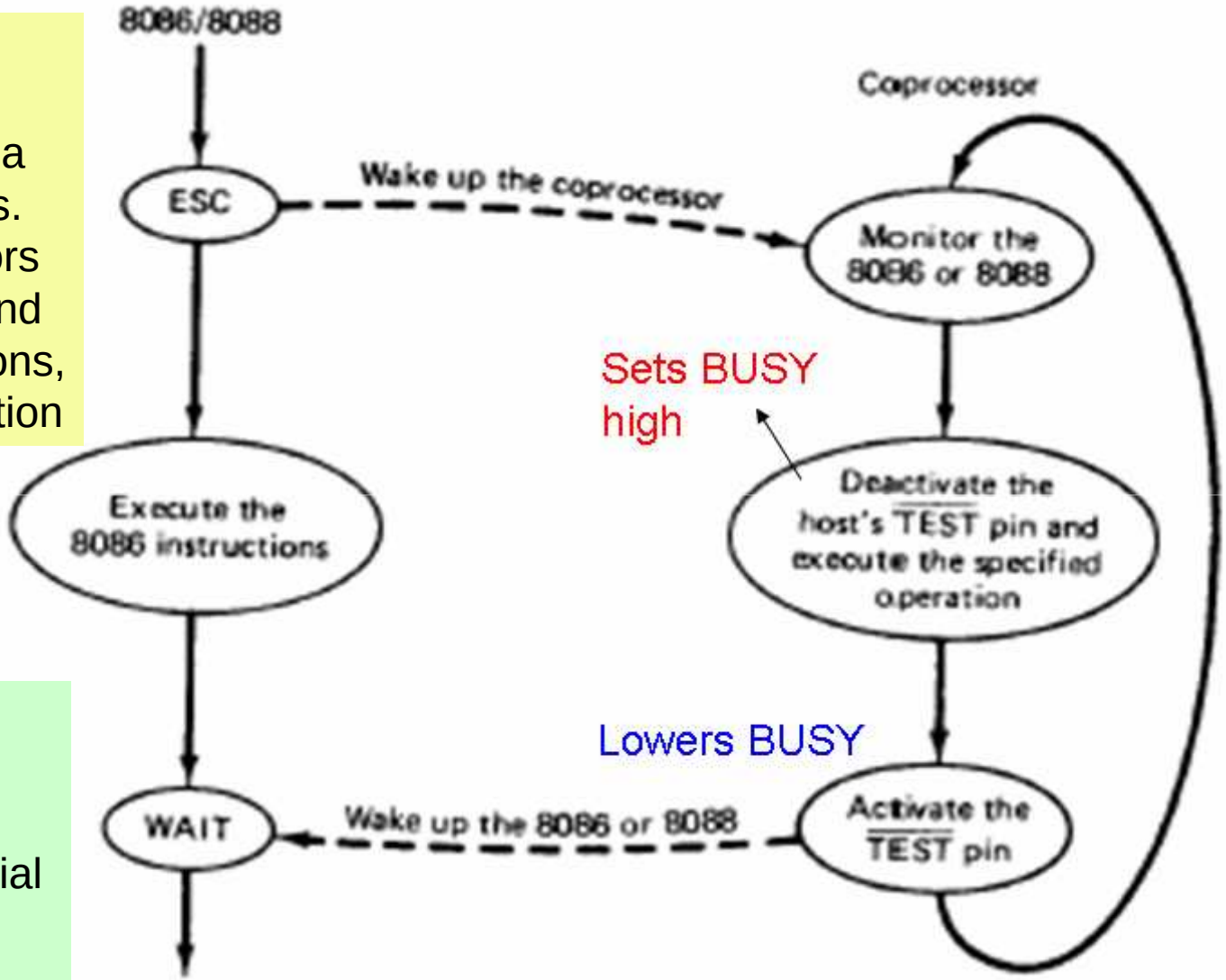
- In addition, when the 8087 is executing an instruction, it activates the BUSY pin automatically by putting high on it.
- This pin is connected to the TEST pin of the 8088/86.
- Next, the 8088/86 fetches the next instruction, which is a WAIT instruction that has been inserted by the assembler, and executes it, thereby going into an internal loop while continuously monitoring the TEST input pin to see when this pin goes low.
- When the 8087 finishes execution of the present instruction, it pulls down (low) the READY pin, indicating through the TEST pin to the 8088/86 that it can now send the next instruction to the 8087.



The 8086 with an 8087 Coprocessor 8086 is operating in the MAX mode

Synchronization between 8086 & the 8087 Coprocessor

The assembler marks all FP instructions as ESC instructions having a special range of opcodes. The Coprocessor monitors the 8086 bus activities and Intercepts such instructions, captures them for execution



WAIT instructions can be used to halt the 8086 to ensure that the 8087 has finished a crucial step, e.g. storing a result in memory.

Comparison of 8087 and 8086 Clock Times

In some cases the differences of run times is hours between PCs with and without math-coprocessor.

Table 20-1: Comparison of 8087 and 8086 Clock Times

Instruction	Approximate Execution Time (μs) (5-MHz clock)	
	8087	8086 Emulation
Multiply (single precision)	19	1,600
Multiply (double precision)	27	2,100
Add	17	1,600
Divide (single precision)	39	3,200
Compare	9	1,300
Load (single precision)	9	1,700
Store (single precision)	18	1,200
Square root	36	19,600
Tangent	90	13,000
Exponentiation	100	17,100

8087 Program

Example 20-5

Write an 8087 program that loads three values for X, Y, and Z, adds them, and stores the result.

Solution:

```
finit           ;initialize the 8087 to start at the top of stack
fld     X       ;load X into ST(0). now ST(0)=X
fld     Y       ;load Y into ST(0). now ST(0)=Y and ST(1)=X
fld     Z       ;load Z into ST(0). now ST(0)=Z,ST(1)=Y,ST(2)=X
fadd    ST(1)   ;add Y to Z and save the result in ST(0)
fadd    ST(2)   ;add X to (Y+Z) and save it in ST(0)
fst     sum     ;store ST(0) in memory location called sum.
```

Now the same program can be written as follows:

```
finit
fld     X               ;load x, now ST(0) = x
fld     Y               ;load y, now ST(0)= y, ST(1) = x
fld     Z               ;load z, now ST(0)=z, ST(1)=y, ST(2)=x
fadd                    ;adds y to z
fadd    ST(2)           ;adds x to (y + z)
fst     sum
```

Program 20-2 shows the actual MASM code and execution. Figure 20-2 shows the registers.

Other data formats of the 8087

- In addition to short real (single precision) and long real (double precision) representations for real numbers, the 8087 also supports 16 , 32 , and 64 bit integers.
- They are referred to as
  - *word integers*,
  - *short integers*, and
  - *long integers*,
- respectively, and are shown in Figure 20-1.
- These forms are sometimes referred to as *signed integer numbers*.
- No decimal points are allowed in integers, in contrast to real numbers, in which decimal points are allowed.

Different Data Representation of 8087

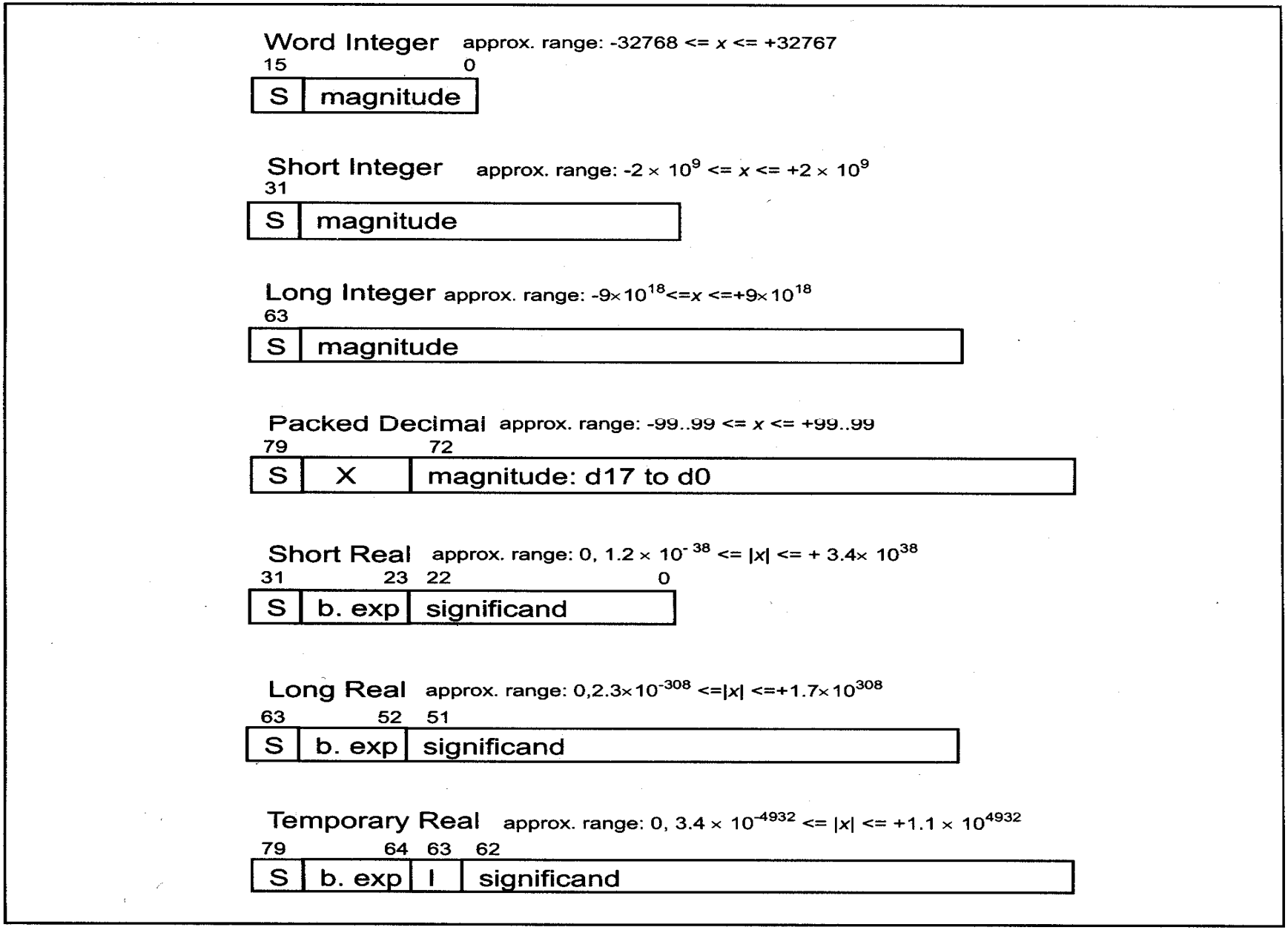


Figure 20-1. 80x87 Data Formats

different data directives

- There are different directives to define the different data types of the coprocessor.

They are as follows:

- DW (Define word) for word integer
- DD (Define double word) for short real (single precision)
  & for short integer
- DQ (Define quad word) for long real (double precision)
  & for long integer
- DT (Define ten bytes) for packed decimal
  & for temporary real

80x87 registers

- There are only 8 general-purpose registers in the 80x87.
- Rather than having different-size registers for different-size operands, all the registers of the 8087 are 80 bits wide.
- Every time the 8087 loads an operand, it automatically converts it to this 80-bit format.
- This gives uniformity to the registers and makes programming, as well as 8087 hardware design, much easier.
- Although these 8 registers have been numbered from 0 to 7, they are accessed like a stack, meaning that a last-in-first-out policy is used.
- At any given time, the top of the stack is referred to as ST(0), or simply ST, and all other registers, regardless of their number, are referred to according to their positions compared to the top of the stack, ST.
- The programming examples below will demonstrate the use of registers in the 8087.
- Example 20-5 will show a complete Assembly language program using the 8087 coprocessor.

80x87 Assembly code and registers

1. All 80x87 mnemonics start with the letter “ f ” to distinguish them from 80x86 instructions.
2. . Whenever a register is not identified specifically, ST [which is ST(0)] is assumed automatically.
3. ST(0) is the top of the stack, ST(1) is one register below that, and ST(2) is two registers below ST(0), and so on. In other words, for register ST(m), the number in parentheses, m, has nothing to do with the register number. There is a way to find out which register number, 0 - 7, is ST(0), the top of the stack. .

How 8087 uses its registers

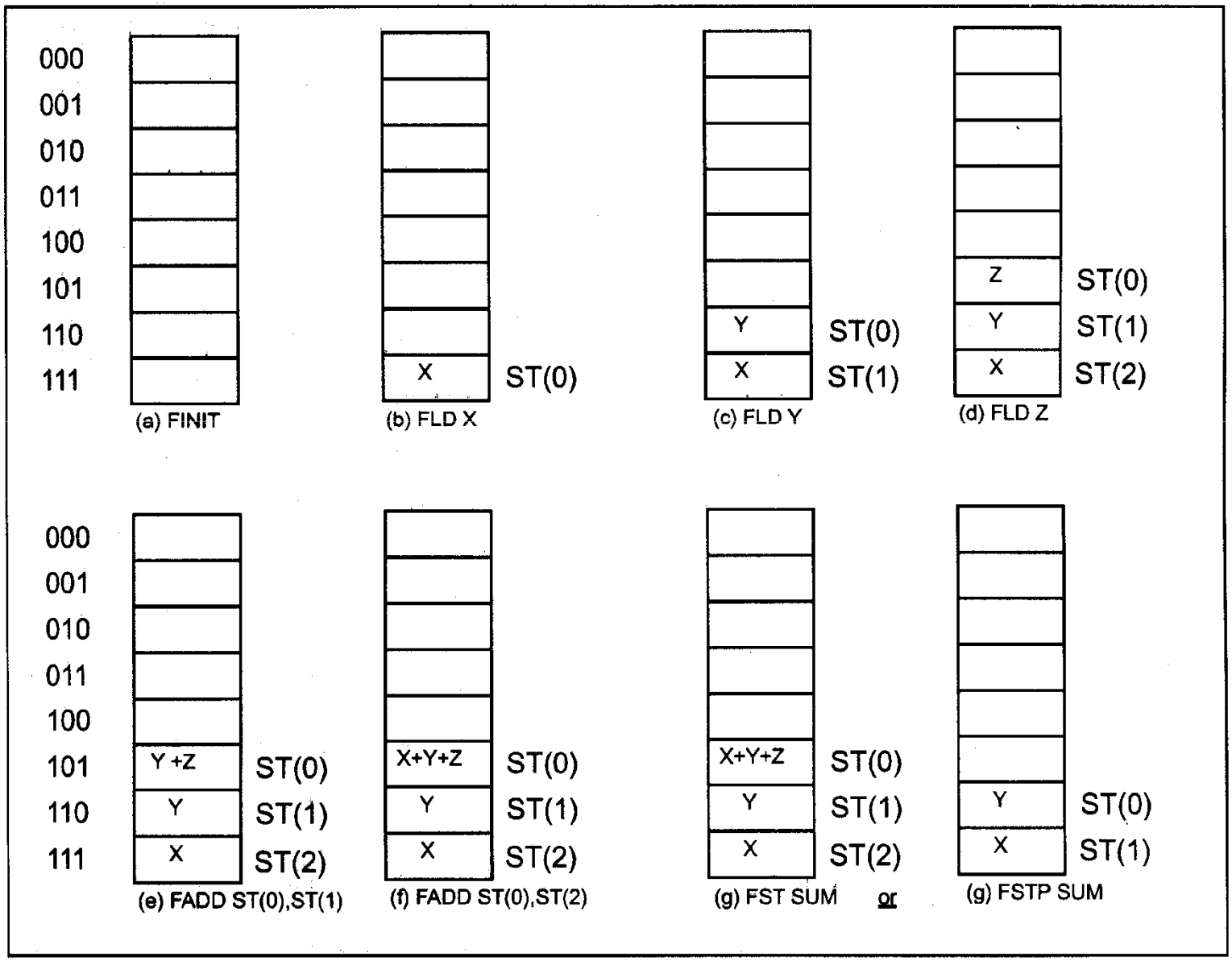


Figure 20-2. Stack Diagram for Example 20-5

Reading an operand by the coprocessor

- Assume that the 8087 needs to read an operand. When the 8088/86 initiates the operand read cycle, the 8087 grabs the 20-bit address and saves it internally.
- If the operand is a single word (like a word integer), the read cycle has been initiated and the word will come into both processors.
- Only the 8087 will use the data; the 8088/86 will ignore it. However, if the operand is 32 bits or longer, the 8087 will take over the buses by sending a low pulse on its RQ/GTO to the RQ/GT1 of the 8088/86

Reading an operand by the coprocessor

- The 8088/86 in turn will send back a low pulse through the same pin, thereby allowing the 8087 to take over the buses.
- Remember, RQ/GT is a bidirectional bus.
- When the 8087 takes over the buses, it will use them until it brings in the last byte of the operand.
- It is only then that by activating RQ/GT (making it low), control of the buses is given back to the 8088/86.
- For example, in the case of a DT operand, the 8087 has control over the buses for the time needed to fetch all 10 bytes and then it gives back the buses.

Writing an operand by the coprocessor

- In the case of writing an operand by the coprocessor (e.g., FST data), the 8088/86 initiates the write cycle, but the 8087 ignores it since the 8086 does not have the operand.
- This is called a *dummy cycle.*
- All the 8087 does during the dummy cycle is grab the address of the first memory location where the operand is to be stored and keep it until the data is ready, and then it requests the use of the buses by activating the RQ/GT pin.
- From then on, the process is the same as the read cycle, meaning that it will use the buses until it writes the last byte of the operand.
- All the cases discussed so far have been taken care of by either the assembler or the hardware and there was no need for the programmer to be worried.

8087 status words

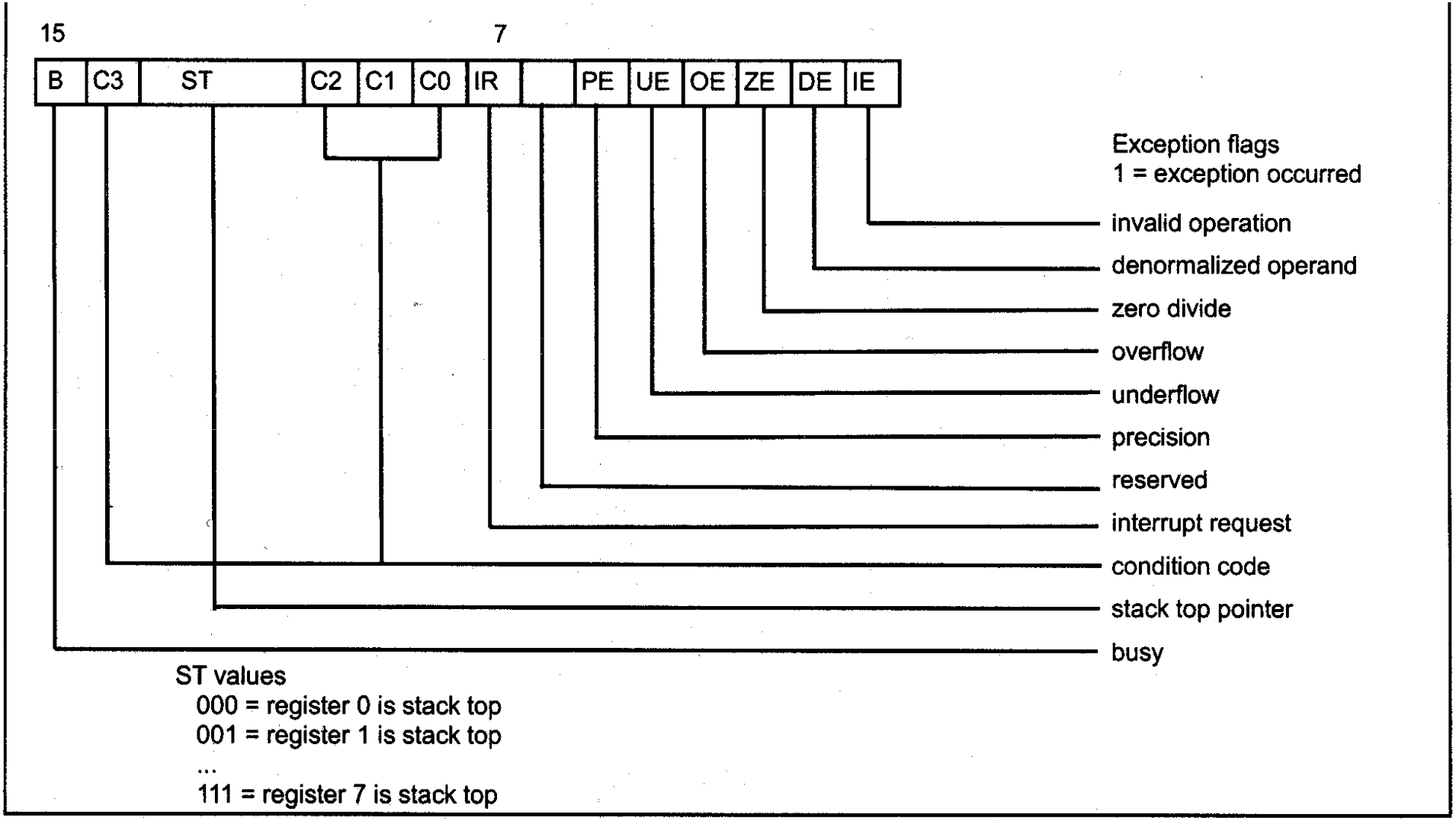


Figure 20-5. 8087 Control and Status Words

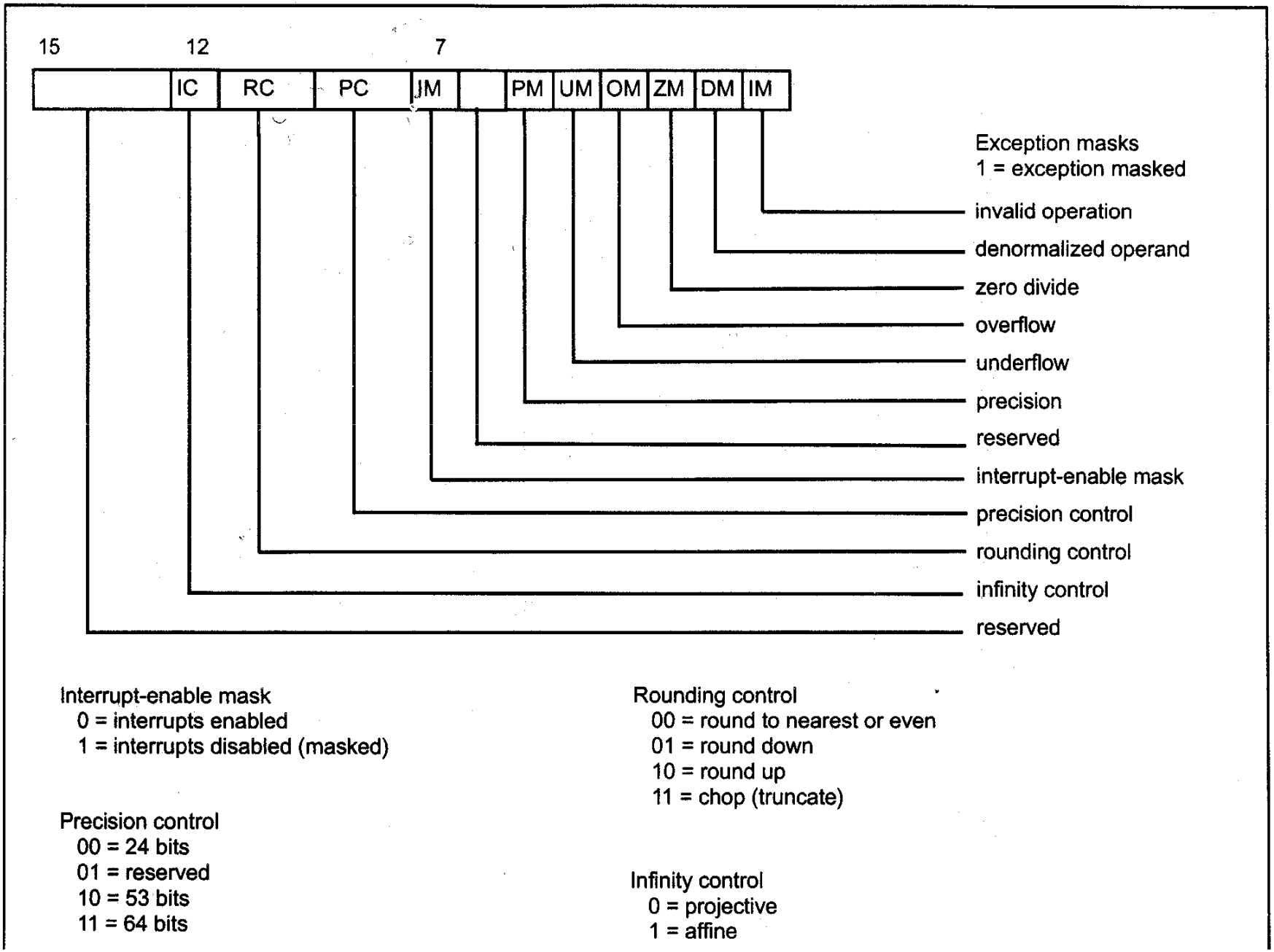
Status Register (cont..)

- Status register reflects the over all operation of the coprocessor.
- B-Busy bit indicates that coprocessor is busy executing a task. Busy can be tested by examining the status or by using the FWAIT instruction. Newer coprocessor automatically synchronize with the microprocessor, so busy flag need not be tested before performing additional coprocessor tasks.
- C_3-C_0 Condition code bits indicates conditions about the coprocessor.

- TOP- Top of the stack (ST) bit indicates the current register address as the top of the stack.
- ES-Error summary bit is set if any unmasked error bit (PE, UE, OE, ZE, DE, or IE) is set. In the 8087 the error summary is also caused a coprocessor interrupt.
- PE- Precision error indicates that the result or operand executes selected precision.
- UE-Under flow error indicates the result is too large to be represent with the current precision selected by the control word.

- OE-Over flow error indicates a result that is too large to be represented. If this error is masked, the coprocessor generates infinity for an overflow error.
- ZE-A Zero error indicates the divisor was zero while the dividend is a non-infinity or non-zero number.
- DE-Denormalized error indicates at least one of the operand is denormalized.
- IE-Invalid error indicates a stack overflow or underflow, indeterminate from (0/0,0,-0, etc) or the use of a NAN as an operand. This flag indicates error such as those produced by taking the square root of a negative number.

8087 control words



Interrupt-enable mask
0 = interrupts enabled
1 = interrupts disabled (masked)

Precision control
00 = 24 bits
01 = reserved
10 = 53 bits
11 = 64 bits

Rounding control
00 = round to nearest or even
01 = round down
10 = round up
11 = chop (truncate)

Infinity control
0 = projective
1 = affine

Control Register

- Control register selects precision, rounding control, infinity control.
- It also masks an unmasks the exception bits that correspond to the rightmost Six bits of status register.
- Instruction FLDCW is used to load the value into the control register.

Control Register (cont..)

- IC –Infinity control selects either affine or projective infinity. Affine allows positive and negative infinity, while projective assumes infinity

  is unsigned.
- RC –Rounding control determines the type of rounding.

INFINITY CONTROL
0 = Projective
1 = Affine

ROUNDING CONTROL
00=Round to nearest or even
01=Round down towards minus infinity
10=Round up towards plus infinity
11=Chop or truncate towards zero

Control Register

- PC- Precision control sets the precision of he result as define in table
- Exception Masks – It Determines whether the error indicated by the exception affects the error bit in the status register. If a logic1 is placed in one of the exception control bits, corresponding status register bit is masked off.

PRECISION CONTROL

00=Single precision (short)

01=Reserved

10=Double precision (long)

11=Extended precision (temporary)

Pentium

- There are few changes as far as instructions and registers are concerned from the 8087 to the math processor inside the Pentium, except for a few new instructions and much lower clock counts for instruction execution.
- The new instructions introduced in the 80387 are FSIN(sine), FCOS (cosine), FSINCOS (sine and cosine), FPREMl (partial remainder), and FUCOM and its variations.