COURSE: UCS1502 - MICROPROCESSORS AND INTERFACING

8087 – Math Coprocessor

Dr. K. R. Sarath Chandran
Assistant Professor, Dept. of CSE

This presentation covers

• Details of 8087 coprocessor

Learning Outcome of this module

• To understand the architecture and programming of 8087 coprocessor

Need of a math coprocessor

- Using a general-purpose microprocessor 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 8086 instructions for mathematical functions.

The Math Coprocessor (Numeric Data Processor)

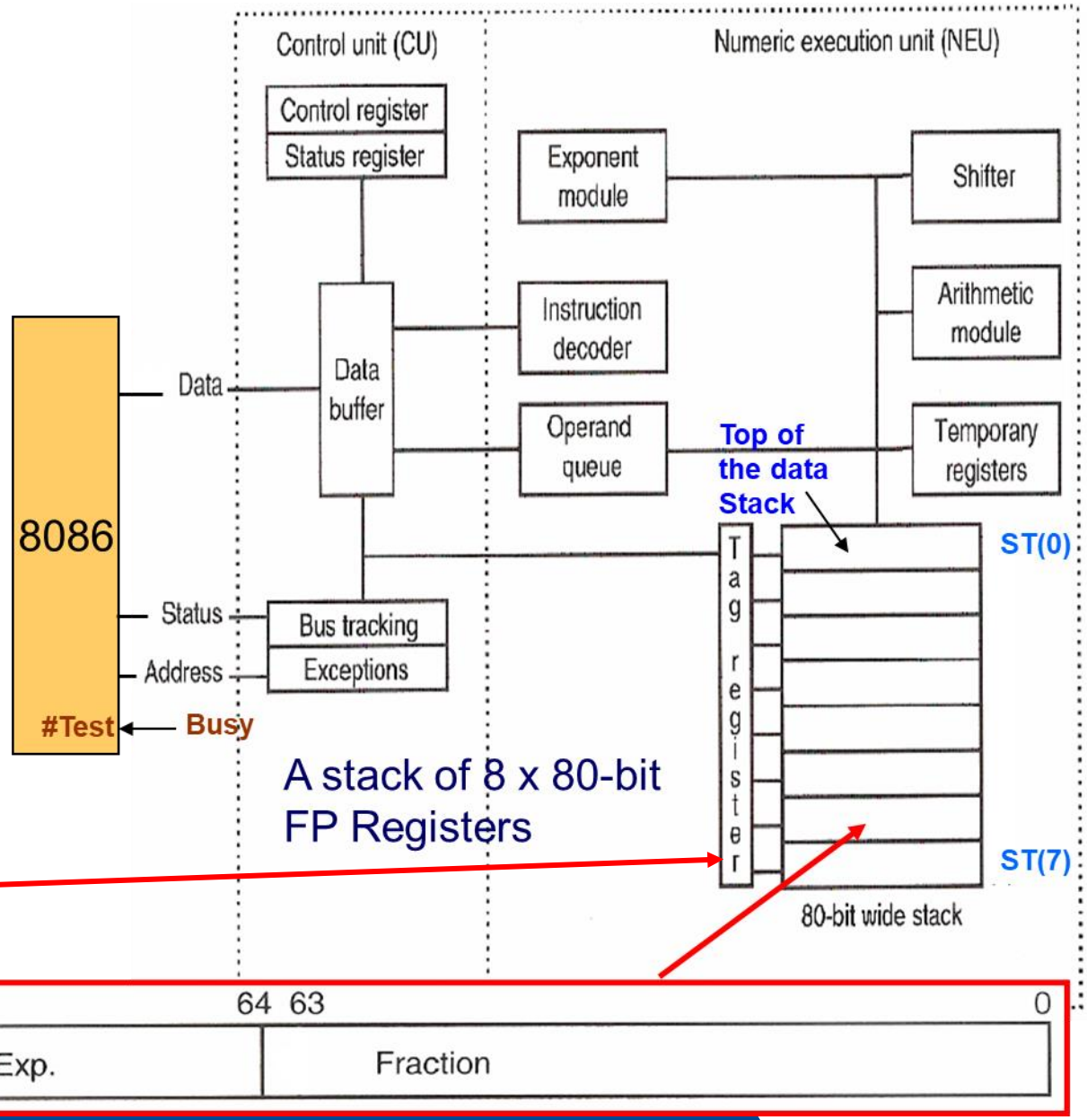
- Complex math operations require large registers, complex circuits and large areas on the chip.
- A general data processor avoids this much burden and delegates such operations to a processor designed specifically for this purpose. e.g. math coprocessor (8087) for the 8086
- The 8086 and the 8087 coprocessors operate in parallel and share the buses 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 factor of 50-100 times

- 8087 Tracks activities of the 8086 by monitoring:
  - Bus status (S0-S2 bits)
  - Queue status (QS0,1)
  - Instruction being fetched (to check if its an ESC instruction)

- Synchronize with WAIT using the BUSY- #TEST signals



Gives the status of registers
00-valid
01-zero
10-invalid
11-empty

The 8087 Coprocessor: Organization

Bus Status Outputs $\overline{S2}$- $\overline{S0}$:

Status bits that encode the type of the current bus cycle

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	*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 $\overline{RQ0}$ / $\overline{GT0}$:

Allow 8087 to request use of the bus

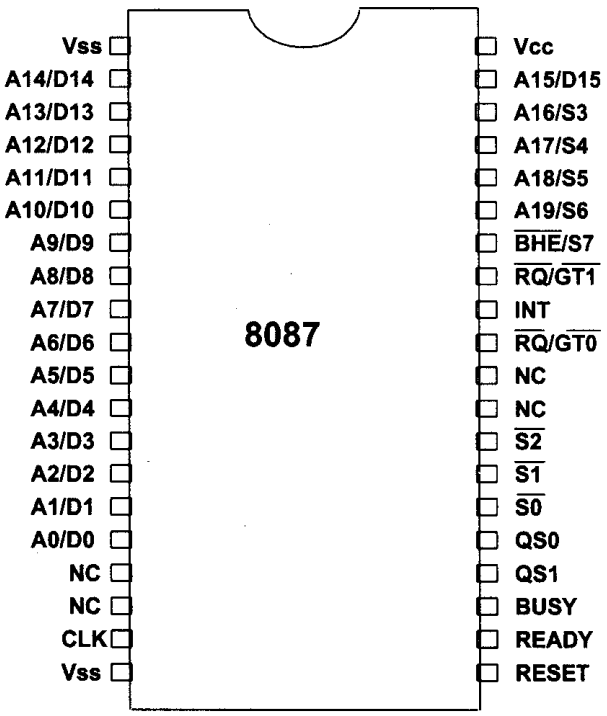
Queue Status Outputs QS1,QS0:

- 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 Signals and connections

1. The 8086 and 8087 receive the same signals, CLK, READY, and RESET, from the 8284. This ensures that they are synchronized.
2. $\overline{S2}$ - $\overline{S0}$ 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 8086 go to the 8087, allowing it to know the status of the queue of the 8086 at any given time.
4. The $\overline{TEST}$ 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. $\overline{RQ}$ / $\overline{GT1}$ (request/grant) of the 8086 is connected to $\overline{RQ}$ / $\overline{GT0}$ of the 8087, allowing them to arbitrate mastery over the buses. There are two sets of RQ/GT: $\overline{RQ}$ / $\overline{GT1}$ and $\overline{RQ}$ / $\overline{GT0}$. $\overline{RQ}$ / $\overline{GT1}$ 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.

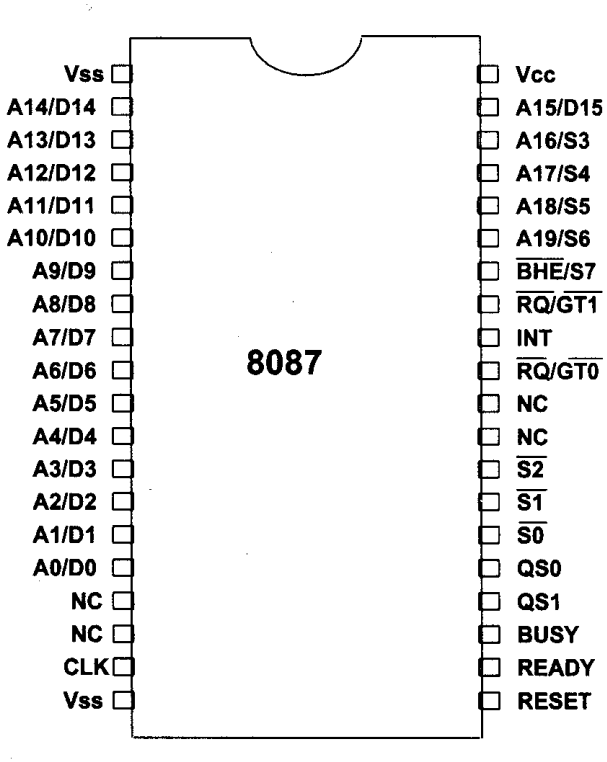


8087 Signals and connections

6. Both the 8086 and 8087 share buses AD0 - AD7 and A8 - A19, allowing either one to access memory. Since the 8087 is designed for both the 8088 and 8086, signal $\overline{BHE}$ is provided for the 8086 processor. It is connected to Vcc if the 8087 is used with the 8088. If the microprocessor used was an 8086, $\overline{BHE}$ from the 8086 is connected to $\overline{BHE}$ 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 8086, often called the host processor, must be connected in maximum mode to be able to accommodate a coprocessor such as the 8087.



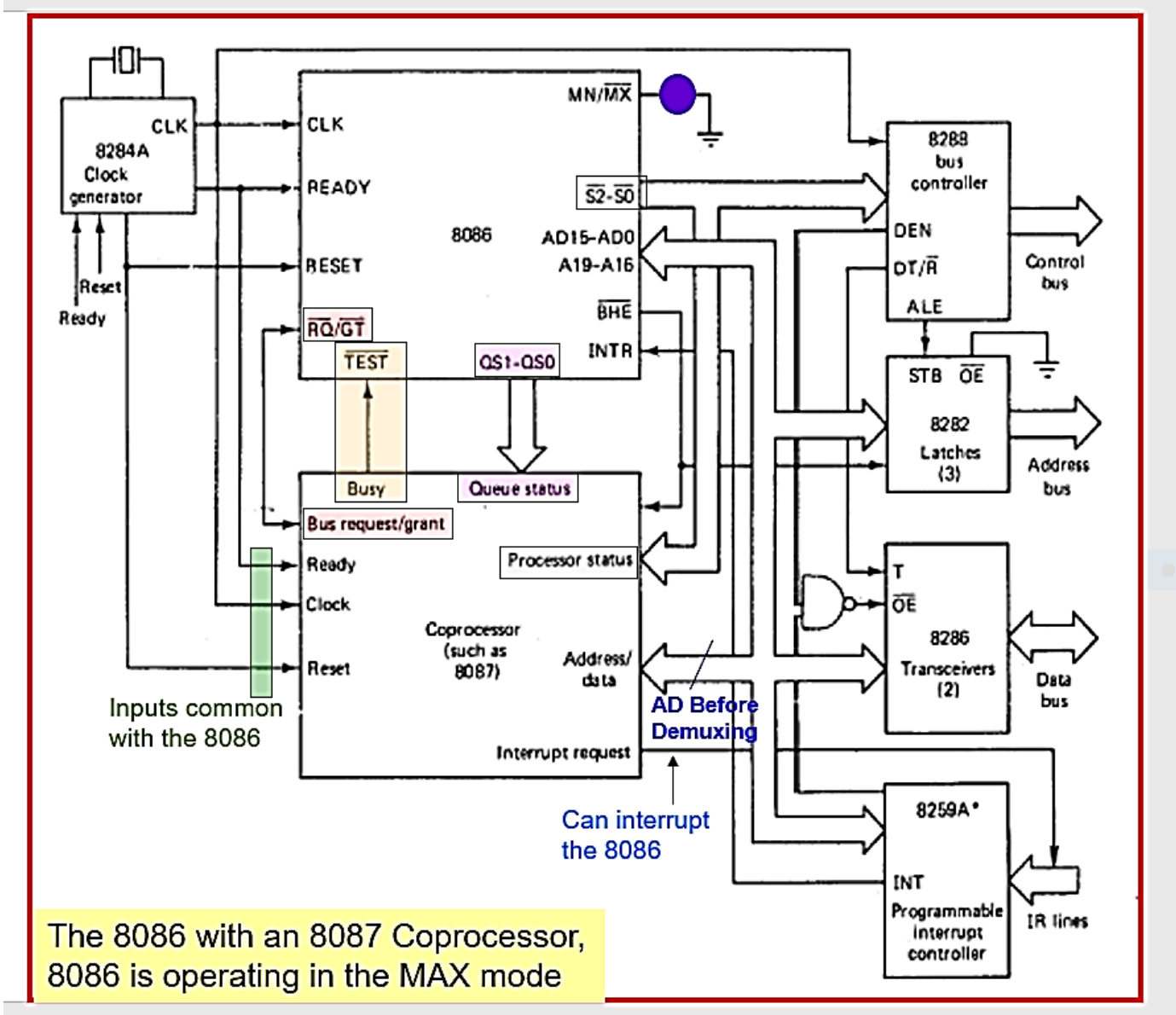
How 8086 and 8087 work together ?

- Both 8086 and 8087 will read instructions from queue.
- Each gets a copy of the instructions as they are fetched from memory.
- Since all the instructions of the 8087 have 11011 in the most significant bits of the opcode, the 8088/86 ignores these instructions.
- In reality, 1001 1101 11011 will be in the initial bits of instructions (first byte 9BH is the opcode for the WAIT instruction).
- Likewise, the 8087 ignores any opcode that lacks this pattern.
- It must be made clear that although both receive a copy of each fetched opcode in their own internal queues, only the 8088/86 can fetch opcodes since it is the only device that has the instruction pointer.
- Each processor decodes all instructions but executes only its own instructions, other instructions will be considered as NOP by each processor.

How 8086 and 8087 work together ?

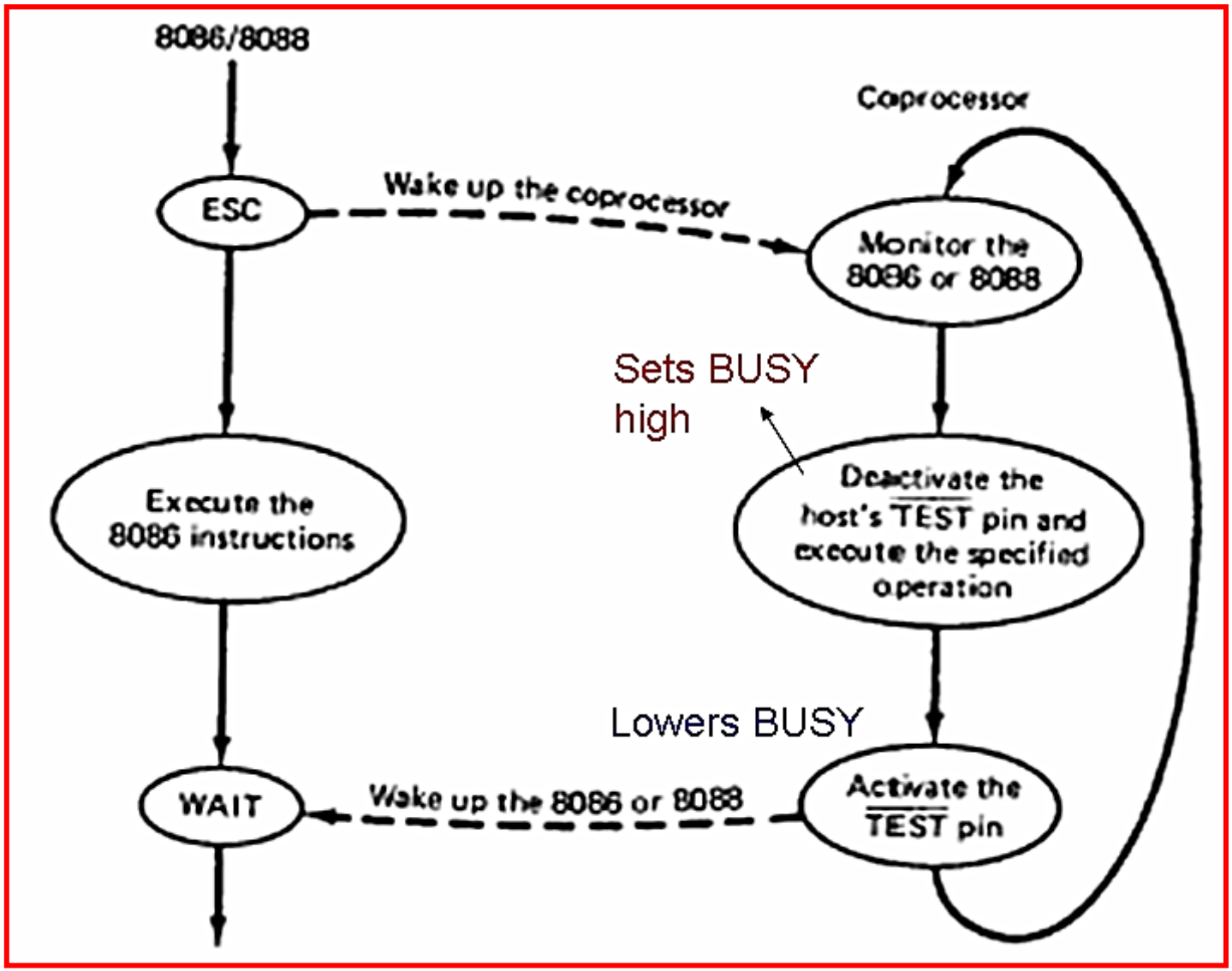
- In addition, when the 8087 is executing an instruction, it activates the BUSY pin automatically by putting high on it.
- This pin is connected to the $\overline{TEST}$ pin of the 8088/86.
- Next, the 8088/86 fetches the next instruction, which is a WAIT instruction(inserted by assembler), and executes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

How 8086 and 8087 work together ?



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

How 8086 and 8087 work together ?



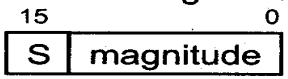
Data formats of 8087

- 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,

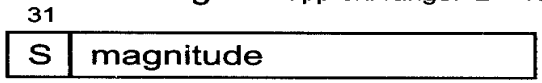
- These forms are sometimes referred to as signed integer numbers.

Data formats of 8087

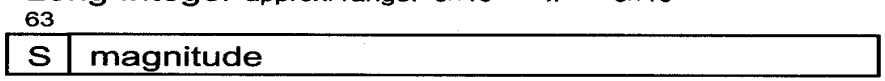
Word Integer approx. range: $-32768 <= x <= +32767$



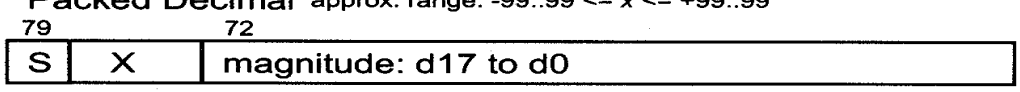
Short Integer approx. range: $-2 \times 10^9 <= x <= +2 \times 10^9$



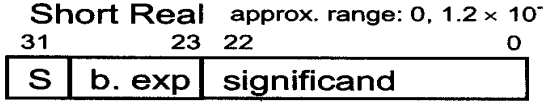
Long Integer approx. range: $-9 \times 10^{18} <= x <= +9 \times 10^{18}$



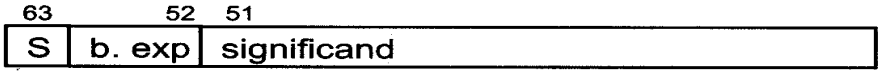
Packed Decimal approx. range: $-99..99 <= x <= +99..99$



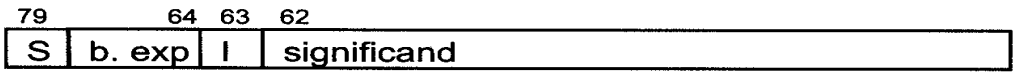
Short Real approx. range: $0,\ 1.2 \times 10^{-38} <= |x| <= +3.4 \times 10^{38}$



Long Real approx. range: $0,2.3 \times 10^{-308} <= |x| <= +1.7 \times 10^{308}$



Temporary Real approx. range: $0,\ 3.4 \times 10^{-4932} <= |x| <= +1.1 \times 10^{4932}$



8087 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.
- All 80x87 mnemonics start with the letter " f " to distinguish them from 80x86 instructions.
- 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.

8087 Programming

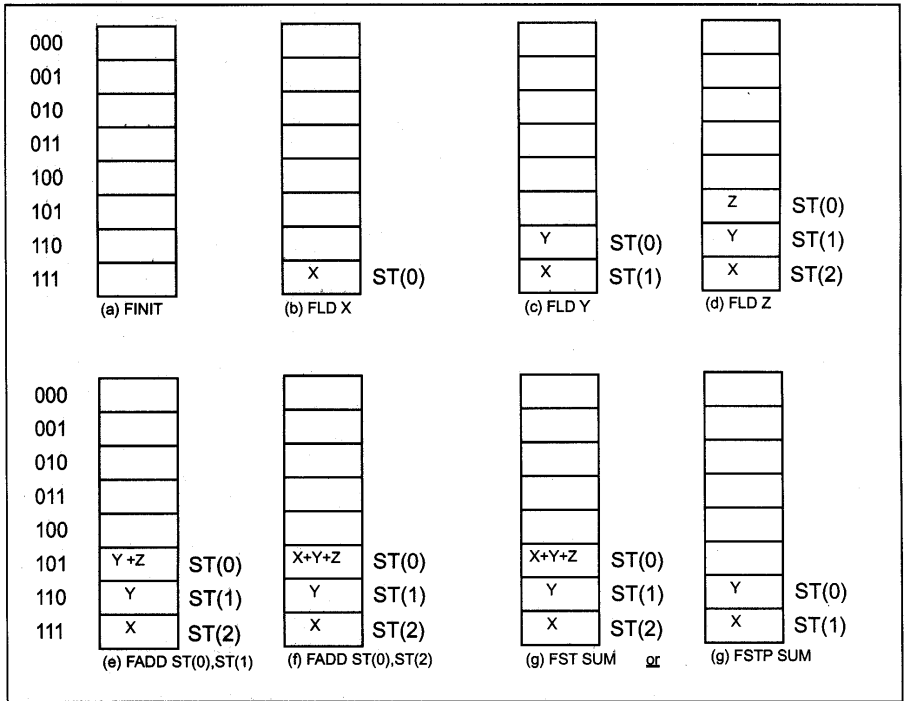
Write an 8087 program that loads three values for X, Y, and Z, adds them, and stores the result.

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.
```

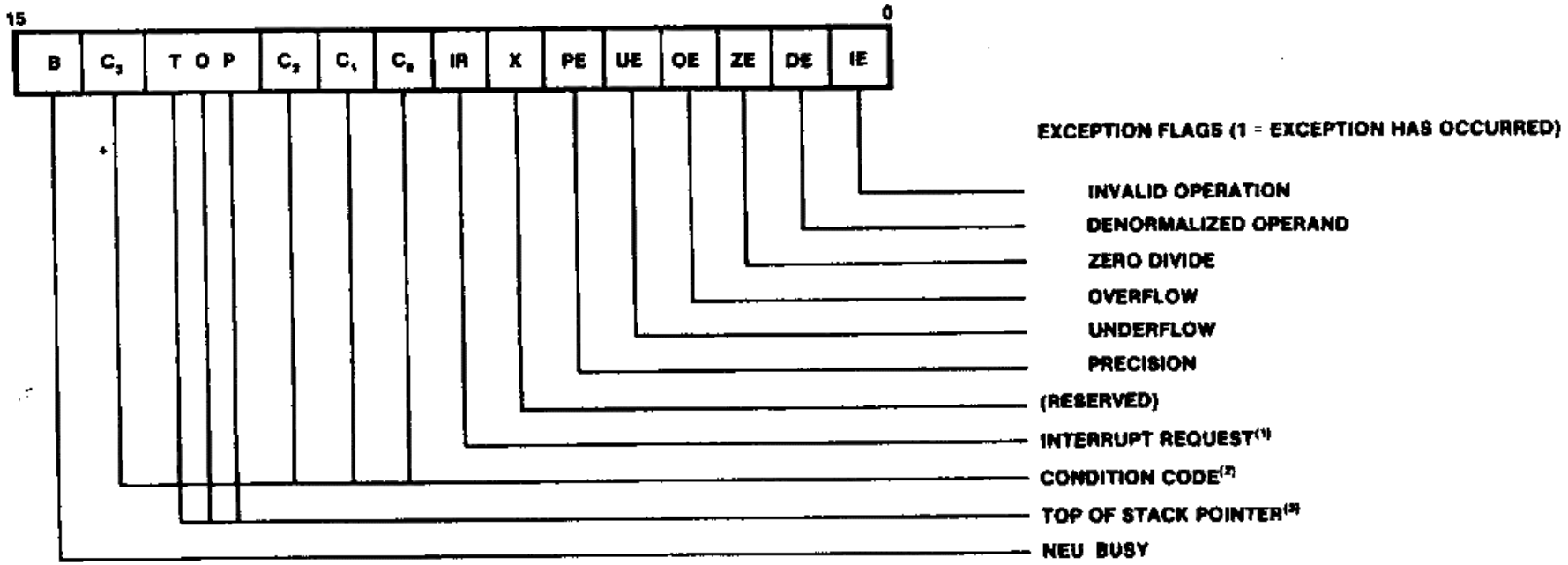
Now the same program can be written as follows:

```
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
```



Stack snapshot during execution

8087 status word



NOTES:

1. IR is set if any unmasked exception bit is set, cleared otherwise.
2. See Table 3 for condition code interpretation.
3. Top Values:
   000 = Register 0 is Top of Stack.
   001 = Register 1 is Top of Stack.
   •
   •
   •
   111 = Register 7 is Top of Stack.

8087 status word

The status word shown in Figure reflects the overall state of the 8087; it may be stored in memory and then inspected by CPU code. The status word is a 16-bit register divided into fields as shown in Figure

The busy bit (bit 15) indicates whether the NEU is either executing an instruction or has an interrupt request pending (B = 1), or is idle (B = 0).

8087 status word

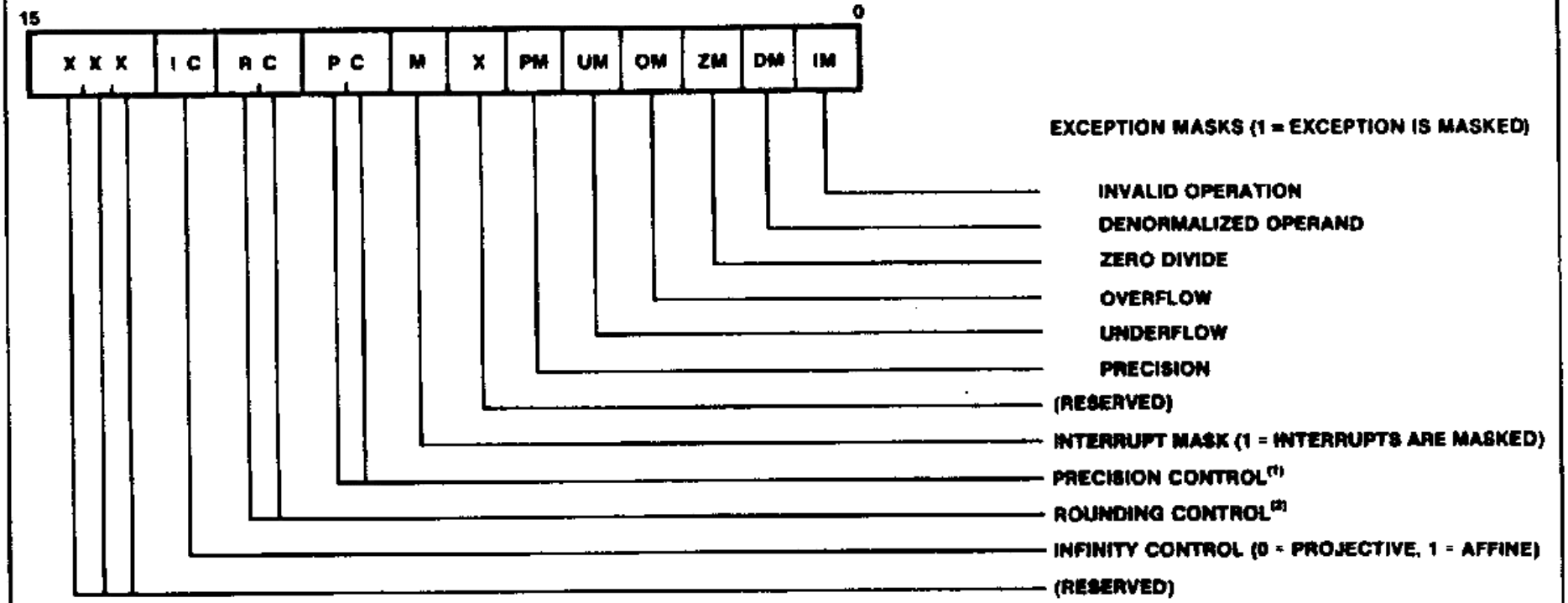
The four numeric condition code bits (C_0–C_3) are similar to flags in a CPU: various instructions update these bits to reflect the outcome of the 8087 operations. The effect of these instructions on the condition code bits is summarized in Table 4. **(8087 intel data sheet)**

Bits 14–12 of the status word point to the 8087 register that is the current top-of-stack (TOP) as described above.

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

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

8087 control word



NOTES:

1. Precision Control
   - 00 = 24 bits
   - 01 = Reserved
   - 10 = 53 bits
   - 11 = 64 bits

2. Rounding Control
   - 00 = Round to Nearest or Even
   - 01 = Round Down (toward $-\infty$)
   - 10 = Round Up (toward $+\infty$)
   - 11 = Chop (truncate toward zero)

8087 control word

The low order byte of this control word configures 8087 interrupts and exception masking. Bits 5–0 of the control word contain individual masks for each of the six exceptions that the 8087 recognizes and bit 7 contains a general mask bit for all 8087 interrupts.

The high order byte of the control word configures the 8087 operating mode including precision, rounding, and infinity controls. The precision control bits (bits 9–8) can be used to set the 8087 internal operating precision at less than the default of temporary real precision. This can be useful in providing compatibility with earlier generation arithmetic processors of smaller precision than the 8087. The rounding control bits (bits 11–10) provide for directed rounding and true chop as well as the unbiased round to nearest mode specified in the proposed IEEE standard. Control over closure of the number space at infinity is also provided (either affine closure, $\pm\infty$, or projective closure, ∞, is treated as unsigned, may be specified).

References

- Doughlas V. Hall, “Microprocessors and Interfacing, Programming and Hardware”, Second Edition, TMH.
- 8087- Intel data sheet

Thank you