

BIBLIOGRAPHY

Because of the technical level of this book, the major sources of further information on the topics discussed are manufacturer's data books, application notes, and articles in current engineering periodicals. With the foundation you get from this book you should be able to comfortably read these materials. Listed below, by chapter, are some materials that will give you more details for many of the topics we discuss in the book. Following the chapter listings is a list of periodicals which we have found to be particularly helpful in keeping up with the latest advances in microcomputer evolution and applications.

Chapter 1

Hall, Douglas V., *Digital Circuits and Systems*, McGraw-Hill, Inc., New York, 1989.

Chapters 2-6

Mick, John, and Jim Brick; *Bit-Slice Microprocessor Design*, McGraw-Hill, Inc., New York, 1980.

A History of Microprocessor Development at Intel, Intel Article Reprint/AR-173, Intel Corporation, Santa Clara, Calif.

Microprocessors, Intel Corporation, Santa Clara, Calif., latest edition (Databook).

Turbo Assembler User's Guide, Borland International, Scotts Valley, Calif., 1988.

Macro Assembler User's Guide, Microsoft Corp., Beltingham, Wash., 1989.

Peripherals, Intel Corporation, Santa Clara, Calif., latest edition (Databook).

SDK-86 MCS-86 System Design Kit User's Guide, Intel Corporation, Santa Clara, Calif., 1981.

8086 System Design, Application Note, Intel Corporation/AP-67, Intel Corporation, Santa Clara, Calif.

Chapters 9-10

Peripherals, Intel Corporation, Santa Clara, Calif., latest edition (Databook).

IBM PC TECHNICAL REFERENCE MANUAL, IBM Corp., Boca Raton, Fla., 1983.

Dorf, Richard C., *Robotics and Automated Manufacturing*, Reston Publishing Company, Reston, Va. 1983.

AMF Potter & Brumfield Catalog, Potter & Brumfield, Princeton, Ind., latest edition.

Optoelectronics Designer's Catalog, Hewlett-Packard, Palo Alto, Calif., latest edition.

Interfacing Liquid Crystal Displays in Digital Systems, Application Note AN-8, Beckman Instruments, Inc., Scottsdale, Ariz., latest edition.

Optoelectronics Device Data Book, DL118R1, Motorola Semiconductor Products Inc., Phoenix, Ariz., latest edition.

Sandhu, H. S., Hands-On Introduction to ROBOTICS—

The Manual for the XR-Series Robots, Rhino Robots, Champaign, Ill., latest edition.

Slo-Syn DC Stepping Motors Catalog, DCM1078, Superior Electric Company, Bristol, Conn., latest edition.

Auslander, David M., and Paul Sagues, *Microprocessors for Measurement and Control*, Osborne/McGraw-Hill, Berkeley, Calif., 1981.

Allocca, John A., and Allen Stuart, *Transducers Theory and Applications*, Reston Publishing Company, Inc., Reston, Va., 1984.

Seippel, Robert G., *Transducers, Sensors and Detectors*, Reston Publishing Company, Inc., Reston, Va., 1983.

Johnson, Curtis D., *Process Control Instrumentation Technology*, John Wiley & Sons, New York, latest edition.

Sheingold, Daniel H. (ed.), *Transducer Interfacing Handbook — A Guide to Analog Signal Conditioning*, Analog Devices, Inc., Norwood, Mass., latest edition.

Analog Devices Industrial Control Series Data Sheet, Analog Devices, Inc., Norwood, Mass., latest edition.

Texas Instruments, Inc. *Third Generation TMS320 User's Guide*, Dallas, Tex., latest edition.

Chassaing, Rulph, and Darell W. Horning, *Digital Signal Processing with the TMS320C25*, John Wiley & Sons, New York, 1990.

Chapter 11

Peripherals, Intel Corporation, Santa Clara, Calif., latest edition (Databook).

Microprocessors, Intel Corporation, Santa Clara, Calif., latest edition (Databook).

IBM PC Technical Reference Manual, IBM Corp., Boca Raton, Fla., 1983.

IBM PC/AT Technical Reference Manual, IBM Corp., Florida, 1984.

8086 Macro Assembly Language Reference Manual for 8086-Based Development Systems, Intel Corporation, Santa Clara, Calif., latest edition.

Error Detecting and Correcting Codes, Application Note AP-46, Intel Corporation, Santa Clara, Calif., 1979.

Getting Started With the Numeric Data Processor, Application Note AP-113, Intel Corporation, Santa Clara, Calif., 1981.

Hall, Douglas V., *Digital Circuits and Systems*, McGraw-Hill, Inc., New York, 1989. (Chapter 15 on Electronic Design Automation.)

Chapter 12

Turbo C++ User's Manual; Turbo C++ Programmer's Guide; Turbo C++ Library Reference, Borland International, Scotts Valley, Calif., 1990.

Microsoft C 6.0 Optimizing Compiler, Language Reference, Codeview, and Utilities, Microsoft Corporation, Bellingham, Wash., 1987.

Schildt, Herbert, *Turbo C: The Complete Reference*, Borland-Osborne/McGraw-Hill, Berkeley, Calif., 1988.

Walte, Michael, and Stephen Prata, *New C Primer Plus*, Howard W. Sams & Company, Carmel, Ind., 1990.

Chapter 13

Peripherals, Intel Corporation, Santa Clara, Calif., latest edition (Databook).

Lesea, Austin, and Rodney Zaks, *Microprocessor Interfacing Techniques*, Sybex Inc., Berkeley, Calif., latest edition.

An Intelligent Data Base System Using the 8272, Application Note AP-116, Intel Corporation, Santa Clara, Calif., 1981. (Old, but good basics.)

Sutty, George, and Steve Blair, *Programmer's Guide to the EGAVGA*, Brady Books, New York, NY, 1988.

Stevens, Roger T., *Fractal Programming in C*, M&T Books, Redwood City, Calif., 1989. (Demo disk comes with book and provides much fun.)

DOS Technical Reference Manual, Microsoft Corporation, Bellingham, Wash., latest edition.

Jamsa, Kris, *DOS—The Complete Reference*, Osborne/McGraw-Hill, Berkeley, Calif., 1987.

Jamsa, Kris, *DOS Power User's Guide*, Osborne/McGraw-Hill, Berkeley, Calif., 1988.

Luther, Arch, *Digital Video in the PC Environment*, McGraw-Hill, New York, NY, 1989.

Phillips International, Inc. *Compact Disk-Interactive, A Designer's Overview*, McGraw-Hill, New York, NY, 1989.

Chapter 14

Microcommunications Handbook, Intel Corporation, Santa Clara, Calif., latest edition. (Two-volume databook with many application notes.)

Stallings, William D., *Local Networks, an Introduction*, Macmillan, New York, NY, 1984. (Older, but good, clear basics.)

Friend, George E., et al., *Understanding Data Communications*, Howard W. Sams, Indianapolis, Ind., 1987. (Good basics.)

Fike, John L., and George E. Friend, *Understanding Telephone Electronics*, Howard W. Sams, Indianapolis, Ind., 1984. (Good basics.)

Schatt, Stan, *Understanding Local Area Networks*, Howard W. Sams, Indianapolis, Ind., 1989. (Good basics.)

McNamara, John E., *Technical Aspects of Data Communication*, Digital Equipment Corporation, Maynard, Mass., latest edition.

EIA Standard RS-422, Electrical Characteristics of Balanced Voltage Digital Interface Circuits, Electronic Industries Association, Engineering Department, Washington, D.C., 1975.

EIA Standard RS-232-C, Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange, Electronic Industries Association, Engineering Department, Washington, D.C., 1969.

Sterling, Donald J., *Technician's Guide to Fiber Optics*, Delmar Publishers Inc. Albany, NY, 1987. (Good basics with emphasis on cables, connectors, and couplers.)

Chapter 15

Stevens, Al, *Turbo C Memory Resident Utilities, Screen I/O, and Programming Techniques*, MIS Press, Portland, Ore., 1987.

Kaisler, Stephen H., *The Design of Operating Systems for Small Computer Systems*, John Wiley & Sons, Inc., New York, 1983.

Microprocessors, Intel Corporation, Santa Clara, Calif., latest edition (Databook).

ASM386 Assembly Language Reference Manual, Intel Corporation, Santa Clara, Calif., 1986.

80386 System Software Writer's Guide, Intel Corporation, Santa Clara, Calif., 1987.

386 Microprocessor Hardware Reference Manual, Intel Corporation, Santa Clara, Calif., 1988.

80386 Programmer's Reference Manual, Intel Corporation, Santa Clara, Calif., 1986.

Pappas, Chris H., and William H. Murray, *Inside the Model 80*, Osborne/McGraw-Hill, Berkeley, Calif., 1988.

Microsoft Windows User's Guide, Microsoft Corporation, Bellingham, Wash., 1990.

Klimasauskas, Casimir C., *Teaching Your Computer to Learn*, NeuralWare, Inc., Pittsburgh, Penn., 1988 (Booklet).

Periodicals

BYTE. ISSN 0360-5280. Byte Publications, Inc., 70 Main Street, Peterborough, NH 03458.

EDN. ISSN 0012-7515. Cahners Publishing Co., 221 Columbus Avenue, Boston, MA 02116.

Electronic Design. USPS-172-080. Hayden Publishing Co., Inc., 50 Essex Street, Rochelle Park, NJ 07662.

Electronics. ISSN 0013-5070. McGraw-Hill, Inc., 1221 Avenue of the Americas, New York, NY 10020.

Instruments & Control Systems. ISSN 0164-0089. Chilton Company, Chilton Way, Radnor, PA 19089.

Electronic Engineering Times. ISSN 0192-1541. Electronic Engineering Times, 600 Community Drive, Manhasset, NY 11030.

IAPX 86/10 16-BIT HMOS MICROPROCESSOR

8086/8086-2/8086-1

- Direct Addressing Capability 1 MByte of Memory
- Architecture Designed for Powerful Assembly Language and Efficient High Level Languages.
- 14 Word, by 16-Bit Register Set with Symmetrical Operations
- 24 Operand Addressing Modes
- Bit, Byte, Word, and Block Operations
- 8 and 16-Bit Signed and Unsigned Arithmetic in Binary or Decimal Including Multiply and Divide
- Range of Clock Rates:
 - 5 MHz for 8086,
 - 8 MHz for 8086-2,
 - 10 MHz for 8086-1
- MULTIBUS™ System Compatible Interface
- Available in EXPRESS
- Standard Temperature Range
- Extended Temperature Range

The Intel IAPX 86/10 high performance 16-bit CPU is available in three clock rates: 5, 8 and 10 MHz. The CPU is implemented in N Channel depletion load silicon gate technology (HMOS), and packaged in a 40-pin CerDIP package. The IAPX 86/10 operates in both single processor and multiple processor configurations to achieve high performance levels.

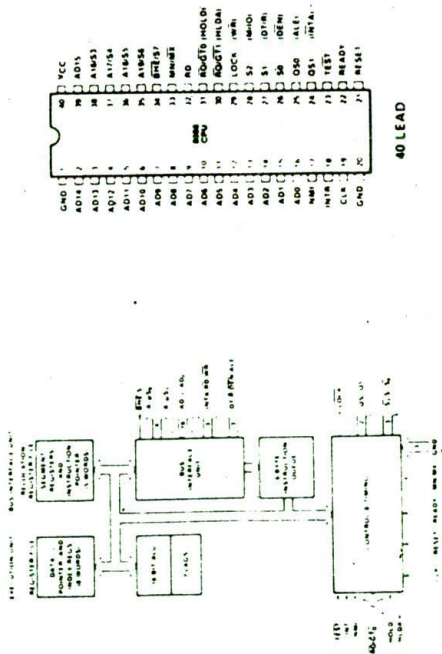


Figure 1. IAPX 86/10 CPU Block Diagram

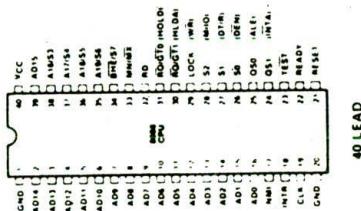


Figure 2. IAPX 86/10 Pin Configuration

Table 1. Pin Description

The following pin function descriptions are for IAPX 86 systems in either minimum or maximum mode. The "Local Bus" in these descriptions is the direct multiplexed bus interface connection to the 8086 (without regard to additional bus buffers).

| Symbol | Pin No. | Type | Name and Function |
|---|----------|------|--|
| AD ₁₅ -AD ₀ | 2-16, 39 | IO | <p>Address Data Bus: These lines constitute the 16-bit multiplexed memory/IO address (T₁) and data (T₂, T₃, T₄, T₅ bus. A₀ is analogous to BHE for the lower byte of the data bus, pins D₇-D₀. It is LOW during T₁ when a byte is to be transferred on the lower portion of the bus in memory or IO operations. Eight-bit oriented devices float to the lower half of the bus in memory or IO operations. Eight-bit oriented devices float to the lower half of the bus normally use A₀ to condition chip select functions. (See BHE.) These lines are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold" acknowledge.</p> <p>Address/Status: During T₁, these are the four most significant address lines for memory operations. During IO operations these lines are LOW. During memory and IO operations, status information is available on these lines during T₂, T₃, T₄, and T₅. The status of the interrupt enable FLAG bit (IS) is updated at the beginning of each CLK cycle. A₁₅, A₁₄, and A₁₃ are encoded as shown. This information indicates which relocation register is presently being used for data accessing.</p> <p>These lines float to 3-state OFF during local bus "hold" acknowledge.</p> |
| A ₁₅ /S ₆ , A ₁₄ /S ₅ , A ₁₃ /S ₄ , A ₁₂ /S ₃ | 35-38 | O | <p>Bus High Enable/Status: During T₁, the bus high enable signal (BHE) should be used to enable data onto the most significant half of the data bus, pins D₁₅-D₈. Eight-bit oriented devices tied to the upper half of the bus would normally use BHE to condition chip select functions. BHE is LOW during T₁ for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. The S₃ status information is available during T₂, T₃, and T₄. The signal is active LOW, and floats to 3-state OFF in "hold." It is LOW during T₁ for the first interrupt acknowledge cycle.</p> |
| BHE/S ₇ | 34 | O | <p>Read Strobe: Read strobe indicates that the processor is performing a memory or IO read cycle, depending on the state of the S₂ pin. This signal is used to read devices which reside on the 8086 local bus. RD is active LOW during T₂, T₃ and T₄ of any read cycle, and is guaranteed to remain HIGH in T₂ until the 8086 local bus has floated. This signal floats to 3-state OFF in "hold acknowledge."</p> <p>READY: is the acknowledge signal from the addressed memory or IO device that it will complete the data transfer. The READY signal from memory is synchronized by the 8284 Clock Generator to form READY. This signal is active HIGH. The 8086 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.</p> |
| RD | 32 | O | <p>Interrupt Request: is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector-lookup table located in system memory. It can be internally masked by software reasserting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.</p> |
| READY | 22 | I | <p>TEST: input is examined by the "Wait" instruction. If the TEST input is LOW execution continues, otherwise the processor waits in an "idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.</p> |
| INTR | 18 | I | |
| TEST | 23 | I | |

intel

IAPX 86/10

Table 1. Pin Description (Continued)

| Symbol | Pin No. | Type | Name and Function |
|-----------------|---------|------|---|
| NMI | 17 | I | Non-maskable interrupt: an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized. |
| RESET | 21 | I | Reset: causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the instruction set description, when RESET returns LOW. RESET is internally synchronized. |
| CLK | 19 | I | Clock: provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing. |
| V _{CC} | 40 | | V _{CC} : +5V power supply pin. |
| GND | 1, 20 | | Ground |
| MINIMX | 33 | I | Minimum/Maximum: indicates what mode the processor is to operate in. The t_0 modes are discussed in the following sections. |

The following pin function descriptions are for the 8086/8288 system in maximum mode (i.e., $\overline{MNIMX} = V_{CC}$). Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

| S_7, S_6, S_5 | S_4 | S_3 | S_2 | S_1 | S_0 | Characteristics |
|-----------------|-------|-------|-------|-------|-------|-----------------|
| 0 | 0 | 0 | 0 | 0 | 0 | Interrupt |
| 0 | 0 | 0 | 0 | 0 | 1 | Read IO Port |
| 0 | 0 | 0 | 0 | 1 | 0 | Write IO Port |
| 0 | 0 | 0 | 1 | 0 | 0 | Code Access |
| 0 | 0 | 1 | 0 | 0 | 0 | Data Access |
| 0 | 1 | 0 | 0 | 0 | 0 | Read Memory |
| 0 | 1 | 0 | 0 | 1 | 0 | Write Memory |
| 0 | 1 | 1 | 0 | 0 | 0 | Parity |

Status: active during T_1 , T_2 , and T_3 and is returned to the passive state (0,1,1) during T_3 or during T_4 when READY is HIGH. This status is used by the 8288 Bus Controller to generate all memory and I/O access control signals.

Any change by S_7, S_6, S_5 , or S_4 during T_1 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_4 is used to indicate the end of a bus cycle.

These signals float to 3-state OFF in "hold acknowledge" edge. These status lines are encoded as shown.

Request/Grant: pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with \overline{RDST} , having higher priority than \overline{RDST} . \overline{RDST} has an internal pull-up resistor so may be left unconnected. The request/grant sequence is as follows (see Figure 9):

- A pulse of 1 CLK wide from another local bus master indicates a local bus request ("hold") to the 8086 (pulse 1).
- During a T_2 or T_3 clock cycle, a pulse 1 CLK wide from the 8086 to the requesting master (pulse 2), indicates that the 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge."
- A pulse 1 CLK wide from the requesting master indicates to the 8086 (pulse 3) that the "hold" request is about to end and that the 8086 can reclaim the local bus at the next CLK.

Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW.

If the request is made while the CPU is performing a memory cycle, it will release the local bus during t_2 of the cycle when all the following conditions are met:

- Request occurs on or before T_2 .
- Current cycle is not the low byte of a word (on an odd address).
- Current cycle is not the first acknowledge of an interrupt acknowledge sequence.
- A locked instruction is not currently executing.

intel

IAPX 86/10

Table 1. Pin Description (Continued)

| Symbol | Pin No. | Type | Name and Function |
|-----------------------------------|---------|------|---|
| LOCK | 29 | O | If the local bus is idle when the request is made the two possible events will follow: 1. Local bus will be released during the next clock. 2. A memory cycle will start within 3 clocks. Now the four rules for a currently active memory cycle apply with condition number 1 already satisfied. LOCK output indicates that other system bus masters are not to gain control of the system bus while LOCK is active LOW. The LOCK signal is activated by the "LOCK" prefix instruction and remains active until the completion of the next instruction. This signal is active LOW, and floats to 3-state OFF in "hold acknowledge." |
| OS ₁ , OS ₀ | 24, 25 | O | Queue Status: The queue status is valid during the CLK cycle after which the queue operation is performed. OS ₁ and OS ₀ provide status to allow external tracking of the internal 8086 instruction queue. |

| OS ₁ | OS ₀ | CHARACTERISTICS |
|-----------------|-----------------|----------------------------------|
| 0 | 0 | No Operation |
| 0 | 1 | First Byte of Op Code from Queue |
| 1 | 0 | Empty the Queue |
| 1 | 1 | Subsequent BYT from Queue |

The following pin function descriptions are for the 8086 in minimum mode (i.e., $\overline{MNIMX} = V_{CC}$). Only the pin functions which are unique to minimum mode are described; all other pin functions are as described above.

| | | | |
|------------|--------|-----|--|
| M/IO | 28 | O | Status line: logically equivalent to S_2 in the maximum mode. It is used to distinguish a memory access from an I/O access. M/IO becomes valid in the T_1 preceding a bus cycle and remains valid until the final T_1 of the cycle ($M = \text{HIGH}$, $IO = \text{LOW}$). M/IO floats to 3-state OFF in local bus "hold acknowledge." |
| WR | 29 | O | Write: indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/IO signal. WR is active for T_2, T_3 and T_4 of any write cycle. It is active LOW, and floats to 3-state OFF in local bus "hold acknowledge." |
| INTA | 24 | O | INTA is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T_2, T_3 and T_4 of each interrupt acknowledge cycle. |
| ALE | 25 | O | Address Latch Enable: provided by the processor to latch the address into the 8282/8283 address latch. It is a HIGH pulse active during T_1 of any bus cycle. Note that ALE is never floated. |
| DT/R | 27 | O | Data Transceiver/Receiver: needed in minimum system that desires to use an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/R is equivalent to S_1 in the maximum mode, and its timing is the same as for M/IO. ($T = \text{HIGH}$, $R = \text{LOW}$). This signal floats to 3-state OFF in local bus "hold acknowledge." |
| DEN | 26 | O | Data Enable: provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. DEN is active LOW during each memory and I/O access and for INTA cycles. For a read or INTA cycle it is active from the middle of T_2 until the middle of T_4 , while for a write cycle it is active from the beginning of T_2 until the middle of T_4 . DEN floats to 3-state OFF in local bus "hold acknowledge." |
| HOLD, HLDA | 31, 30 | I/O | HOLD: indicates that another master is requesting a local bus "hold." To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of a T_1 clock cycle. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will LOWER the HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. |

The same rules as for \overline{RDST} apply regarding when the local bus will be released. HOLD is not an asynchronous input. External synchronization should be provided if the system cannot otherwise guarantee the setup time.

FUNCTIONAL DESCRIPTION

GENERAL OPERATION

The internal functions of the IAPX 86/10 processor are partitioned logically into two processing units. The first is the Bus Interface Unit (BIU) and the second is the Execution Unit (EU) as shown in the block diagram of Figure 1.

These units can interact directly but for the most part perform as separate asynchronous operational processors. The bus interface unit provides the functions related to instruction fetching and queuing, operand fetch and store, and address relocation. This unit also provides the basic bus control. The overlap of instruction pre-fetching provided by this unit serves to increase processor performance through improved bus bandwidth utilization. Up to 6 bytes of the instruction stream can be queued while waiting for decoding and execution.

The instruction stream queuing mechanism allows the BIU to keep the memory utilized very efficiently. When ever there is space for at least 2 bytes in the queue, the BIU will attempt a word fetch memory cycle. This greatly reduces "dead time" on the memory bus. The queue acts as a First In First Out (FIFO) buffer from which the EU extracts instruction bytes as required. If the queue is empty following a branch instruction, for example, the first byte into the queue immediately becomes available to the EU.

The execution unit receives pre-fetched instructions from the BIU queue and provides an relocated operand addresses to the BIU. Memory operands are passed through the BIU for processing by the EU, which passes results to the BIU for storage. See the instruction set description for further register set and architectural descriptions.

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is organized as a linear array of up to 1 million bytes, addressed as 000000(H) to FFFF(H). The memory is logically divided into code, data, extra data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries. (See Figure 3a.)

All memory references are made relative to base addresses contained in high speed segment registers. The segment types were chosen based on the addressing needs of programs. The segment register to be selected is automatically chosen according to the rules of the following table. All information in one segment type share the same logical attributes (e.g. code or data). By structuring memory into relocatable areas of similar characteristics and by automatically selecting segment registers, programs are shorter, faster, and more structured.

Word (16 bit) operands can be located on even or odd address boundaries and are thus not constrained to even boundaries as is the case in many 16-bit computers. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU automatically performs the proper number of memory accesses; one if the word operand is on an even byte boundary and two if it is on an odd byte boundary. Except for the performance penalty, this double access is transparent to the software. This performance penalty does not occur for instruction fetches; only word operands.

Physically, the memory is organized as a high bank (D₁₅-D₀) and a low bank (D₇-D₀) of 512K 8-bit bytes addressed in parallel by the processor's address lines.

'A₁₅' - A₀: Byte data with even addresses is transferred on the D₇-D₀ bus lines while odd addressed byte data (A₀ HIGH) is transferred on the D₁₅-D₇ bus lines. The processor provides two enable signals, BHE and A₀, to selectively allow reading from or writing into either an odd byte location, even byte location, or both. The instruction stream is fetched from memory as words and is addressed internally by the processor to the byte level as necessary.

| Memory Reference Need | Segment Register Used | Segment Selection Rule |
|-----------------------|-----------------------|--|
| Instructions | CODE (CS) | Automatic with all instruction prefetch |
| Stack | STACK (SS) | All stack pushes and pops. Memory references relative to BP base register except data references. |
| Local Data | DATA (DS) | Data references when relative to stack, destination of string operation, or explicitly overridden. |
| External/Global Data | EXTRA (ES) | Destination of string operations. Explicitly selected using a segment override. |

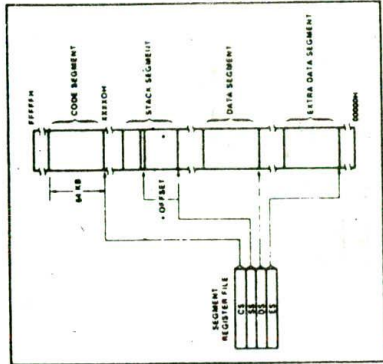


Figure 3a. Memory Organization

In referencing word data the BIU requires one or two memory cycles depending on whether or not the starting byte of the word is on an even or odd address, respectively. Consequently, in referencing word operands performance can be optimized by locating word operands on even address boundaries. This is an especially useful technique for using the stack, since odd address references to the stack may adversely affect the context switching time for interrupt processing or task multi-plexing.

Certain locations in memory are reserved for specific CPU operations (see Figure 3b). Locations from address FFFF(H) through FFFF(H) are reserved for operations following RESET; the CPU will always begin execution at location FFFF(H) where the jump must be executed 0000(H) through 003F(H) are reserved for interrupt operations. Each of the 256 possible interrupt types has its service routine pointed to by a 4 byte pointer element

consisting of a 16-bit segment address and a 16-bit offset address. The pointer elements are assumed to have been stored at the respective places in reserved memory prior to occurrence of interrupts.

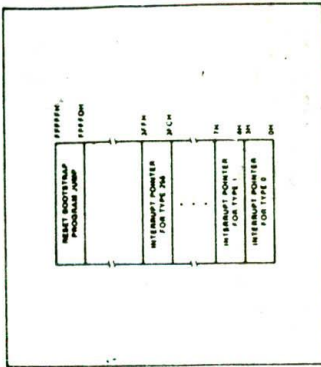


Figure 3b. Reserved Memory Locations

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum IAPX 86/10 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8086 is equipped with a strap pin (MN/MX) which defines the system configuration dependent on a certain subset of the pins changes. The MN/MX pin is strapped to GND; the 8086 treats pins 24 through 31 in maximum mode. An 8288 bus controller interprets status information coded into S₀, S₁, S₂ to generate bus timing and control signals compatible with the MULTIBUS™ architecture. When the MN/MX pin is strapped to Vcc, the 8086 generates bus control signals itself on pins 24 through 31, as shown in parentheses in Figure 2. Examples of minimum mode and maximum mode systems are shown in Figure 4.

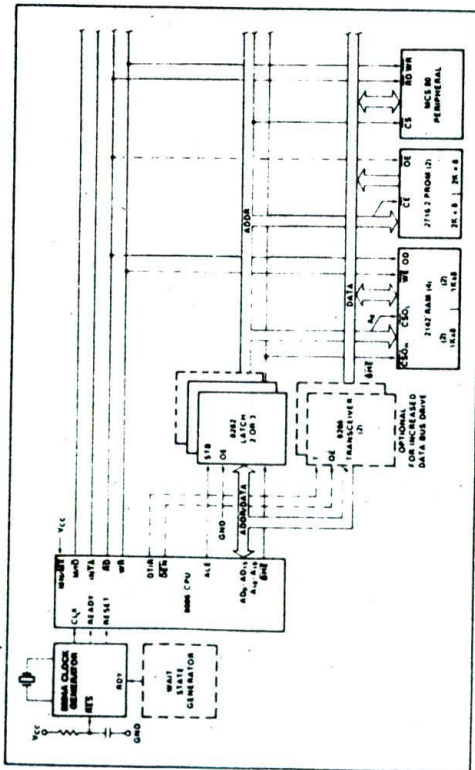


Figure 4a. Minimum Mode IAPX 86/10 Typical Configuration

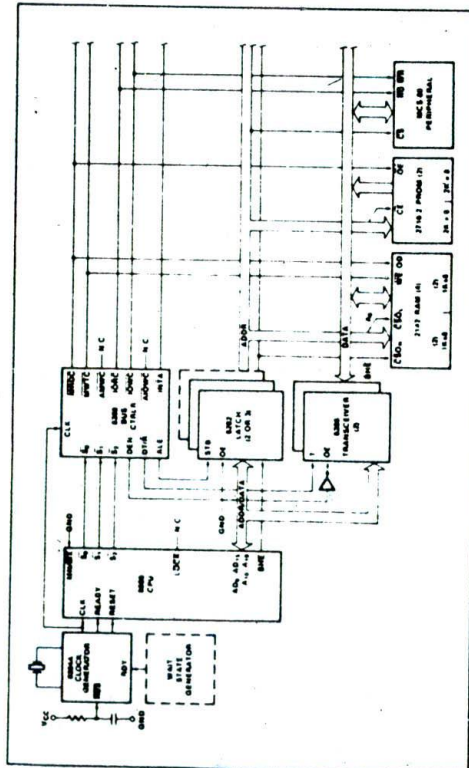


Figure 4b. Maximum Mode IAPX 86/10 Typical Configuration

BUS OPERATION

The 86/10 has a combined address and data bus commonly referred to as a times multiplexed bus. This technique provides the most efficient use of pins on the processor while permitting the use of a standard 40-lead package. This "local bus" can be buffered directly and used throughout the system with address latching provided on memory and I/O modules. In addition, the bus can also be demultiplexed at the processor with a single set of address latches if a standard non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T_1 , T_2 , T_3 , and T_4 (see Figure 5). The address is emitted from the processor during T_1 , and data transfer occurs on the bus during T_3 and T_4 . T_2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "Wait" states (T_w) are inserted between T_3 and T_4 . Each cycle period can occur between 8000 bus cycles. These are referred to as "idle" states (T_i) or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T_1 of any bus cycle the ALE (Address Latch Enable) signal is emitted (by either the processor or the 8288 bus controller, depending on the Min/Max strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits S_0 , S_1 , and S_2 are used, in maximum mode, by the bus controller to identify the type of bus transaction according to the following table.

| S_2 | S_1 | S_0 | CHARACTERISTICS |
|----------|-------|-------|------------------------|
| 0 (LOW) | 0 | 0 | Interrupt Acknowledge |
| 0 | 0 | 1 | Read I/O |
| 0 | 1 | 0 | Write I/O |
| 0 | 1 | 1 | Instruction Fetch |
| 1 (HIGH) | 0 | 0 | Read Data from Memory |
| 1 | 0 | 1 | Write Data to Memory |
| 1 | 1 | 1 | Passive (no bus cycle) |

Status bits S_1 through S_7 are multiplexed with high-order address bits and the BIZ signal, and are therefore valid during T_3 through T_4 . S_0 , S_1 , and S_2 indicate which segment register base instruction set description will be used for this bus cycle in forming the address, according to the following table.

| S_2 | S_1 | CHARACTERISTICS |
|----------|-------|--------------------------------|
| 0 (LOW) | 0 | Alternate Data (extra segment) |
| 0 | 1 | Stack |
| 1 (HIGH) | 0 | Code or None |
| 1 | 1 | Data |

S_8 is a reflection of the PSW interrupt enable bit. $S_9=0$ and S_7 is a spare status bit.

I/O ADDRESSING

In the 86/10, I/O operations can address up to a maximum of 64K I/O byte registers or 32K I/O word registers. The I/O address appears in the same format as the memory address on bus lines A_{15-A_0} . The address lines A_{15-A_16} are zero in I/O operations. The variable I/O instructions which use register DX as a pointer have full address capability while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space.

I/O ports are addressed in the same manner as memory locations. Even addressed bytes are transferred on the D_7-D_0 bus lines and odd addressed bytes on D_7-D_6 . Care must be taken to assure that each register within an 8-bit peripheral located on the lower portion of the bus be addressed as even.

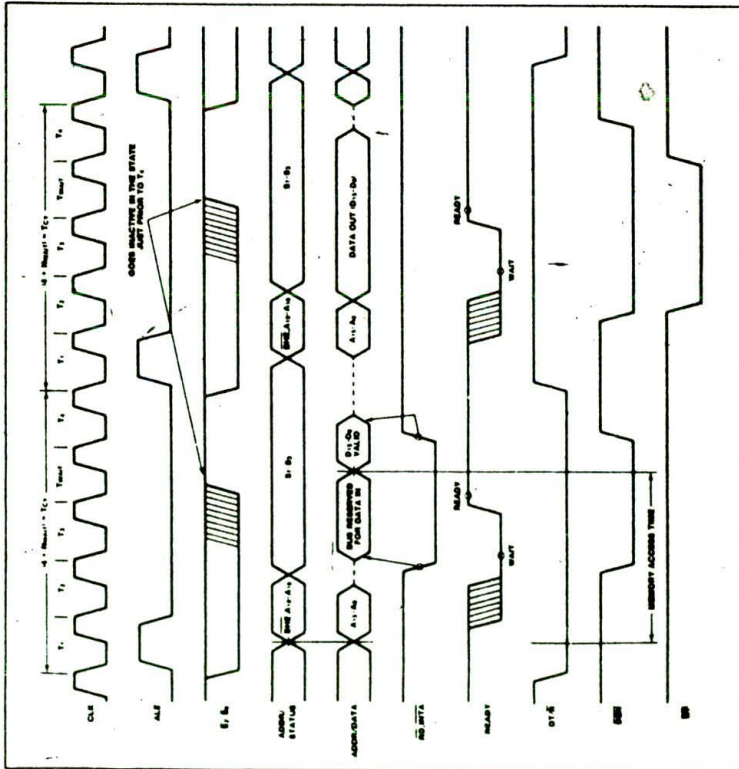


Figure 5. Basic System Timing

EXTERNAL INTERFACE

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8086 RESET is required to be HIGH for greater than 4 CLK cycles. The 8086 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 10 CLK cycles. After this interval the 8086 operates normally beginning with the instruction in absolute location FFFFH (see Figure 3b). The details of this operation are specified in the Instruction Set description of the MCS-86 Family User's Manual. The RESET input is internally synchronized to the processor clock. At initialization the HIGH-to-LOW transition of RESET must occur no sooner than 50 μ s after power-up, to allow complete initialization of the 8086. NMI may not be asserted prior to the 2nd CLK cycle following the end of the RESET.

INTERRUPT OPERATIONS

Interrupt operations fall into two classes, software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the Instruction Set description. Hardware interrupts can be classified as non-maskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256 element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 3b), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type". An interrupting device supplies an 8-bit type number, during the interrupt acknowledge

sequence, which is used to "vector" through the appropriate element to the new interrupt service program location.

NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt pin (NMI) which has higher priority than the maskable interrupt request pin (INTR). A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW-to-HIGH transition. The activation of this pin causes a type 2 interrupt. (See Instruction Set description.)

NMI is required to have a duration in the HIGH state of greater than two CLK cycles, but is not required to be synchronized to the clock. Any high-going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves of a block-type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response, if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE INTERRUPT (INTR)

The 86/10 provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable FLAG status bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block-type instruction. During the interrupt response sequence further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt or single-step), although the

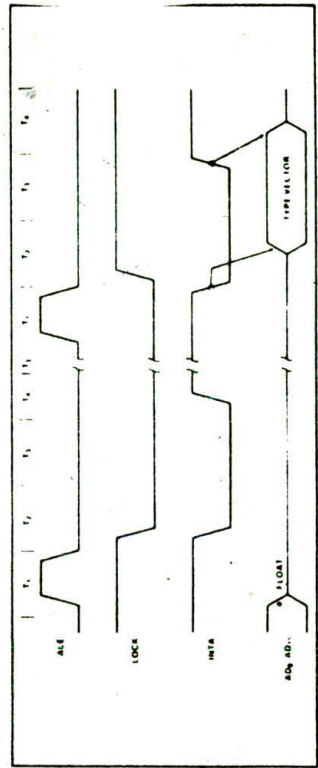


Figure 6. Interrupt Acknowledge Sequence

FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored the enable bit will be zero unless specifically set by an instruction.

During the response sequence (figure 6) the processor executes two successive (back-to-back) interrupt acknowledge cycles. The 8086 emits the LOCK signal from T_2 of the first bus cycle until T_2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle in the second bus cycle a byte is latched from the external interrupt system (e.g., 8259A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector table. An INTR signal level HIGH will be continually responded to within the limitations of the enable bit and sample period. The INTERRUPT RETURN instruction includes a FLAGS pop which returns the status of the original interrupt enable bit when it restores the FLAGS.

HALT

When a software "HALT" instruction is executed the processor indicates that it is entering the "HALT" state in one of two ways depending upon which mode is strapped in minimum mode. The processor issues one ALE with no qualifying bus control signals in Maximum Mode. The processor issues appropriate HALT status on S_2 , S_3 , and the 8288 bus controller issues one ALE. The 8086 will not leave the "HALT" state when a local bus "hold" is entered while in "HALT". In this case, the processor reissues the HALT indicator. An interrupt request or RESET will force the 8086 out of the "HALT" state.

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The LOCK status information is provided by the processor when directly consecutive bus cycles are required during the execution of an instruction. This provides the processor with the capability of performing read/modify/write operations on memory (via the Exchange Register With Memory instruction, for example) without the possibility of another system bus master receiving intervening memory cycles. This is useful in multiprocessor system configurations to accomplish "test and set lock" operations. The LOCK signal is activated (forced LOW) in the clock cycle following the one in which the software "LOCK" prefix instruction is decoded by the EU. It is deactivated at the end of the last bus cycle of the instruction following the "LOCK" prefix instruction. While LOCK is active a request on a RQ/RT pin will be recorded and then honored at the end of the LOCK.

EXTERNAL SYNCHRONIZATION VIA TEST

As an alternative to the interrupts and general I/O capabilities, the 8086 provides a single software testable input known as the TEST signal. At any time the program may execute a WAIT instruction. If at that time the TEST signal is inactive (HIGH), program execution becomes suspended while the processor waits for TEST

to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver (82B6/82B7) is required to buffer the 8086 local bus, signals DTR and DEN are provided by the 8086.

A write cycle also begins with the assertion of ALE and the emission of the address. The MIO signal is again asserted to indicate memory or I/O write operation in the T_2 immediately following the address emission the processor emits the data to be written into the processed location. This data remains valid until the middle of T_4 . During T_2 , T_3 , and T_4 , the processor asserts the write control signal. The write (WR) signal becomes active at the beginning of T_2 as opposed to the read which is delayed somewhat into T_2 to provide time for the bus to float.

The BHE and A0 signals are used to select the proper byte(s) of the memory/I/O word to be read or written according to the following table.

| BHE | A0 | CHARACTERISTICS |
|-----|----|----------------------------------|
| 0 | 0 | Whole word |
| 0 | 1 | Upper byte from/ to odd address |
| 1 | 0 | Lower byte from/ to even address |
| 1 | 1 | None |

IO ports are addressed in the same manner as memory location. Even addressed bytes are transferred on the D_7 - D_0 bus lines and odd addressed bytes on D_{15} - D_8 . The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal (INTA) is asserted in place of the

read (RD) signal and the address bus is floated. (See Figure 6) In the second of two successive INTA cycles, a byte of information is read from bus lines D_7 - D_0 , as supplied by the interrupt system logic (i.e., 8259A Priority Interrupt Controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into an interrupt vector lookup table, as described earlier.

BUS TIMING—MEDIUM SIZE SYSTEMS

For medium size systems the MN/MX pin is connected to V_{SS} and the 8288 Bus Controller is added to the system as well as an 8282/8283 latch for latching the system address and a 8266/8267 transceiver to allow for bus loading greater than the 8086 is capable of handling. Signals ALE, DEN, and DTR are generated by the 8288 (instead of the processor in this configuration) although their timing remains relatively the same. The 8086 status outputs (S_2 , S_3 , and S_4) provide type-of-cycle information and become 8286 inputs. This bus cycle information specifies read (code data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 8286 latches these control signals specifying memory or write, I/O read or write, or interrupt acknowledge. The 8286 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence data is valid at the leading edge of write. The 8266/8267 transceiver receives the usual I and OE inputs from the 8286's DTR and DEN.

The pointer into the interrupt vector table, which is passed during the second INTA cycle, can derive from an 8259A located on either the local bus or the system bus. If the master 8259A Priority Interrupt Controller is positioned on the local bus, a TTL gate is required to disable the 8266/8267 transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll".

to become active. It must remain active for at least 5 CLK cycles. The WAIT instruction is re-executed repeatedly until that time. This activity does not consume bus cycles. The processor remains in an idle state while waiting. All 8086 drivers go to 3-state OFF if bus "hold" is entered. If interrupts are enabled, they may occur while the processor is waiting. When this occurs the processor fetches the WAIT instruction one extra time, processes the interrupt, and then re-latches and re-executes the WAIT instruction upon returning from the interrupt.

BASIC SYSTEM TIMING

Typical system configurations for the processor operating in minimum mode and in maximum mode are shown in Figures 4a and 4b, respectively. In minimum mode, the MN/MX pin is strapped to V_{CC} and the processor emits bus control signals in a manner similar to the 8085 in maximum mode. The MN/MX pin is strapped to V_{SS}, and the processor emits coded status information, which the 8288 bus controller uses to generate MULTIBUS compatible bus control signals. Figure 5 illustrates the signal timing relationships.

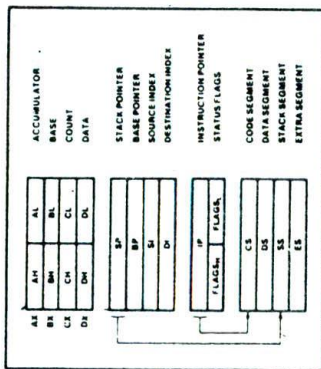


Figure 7. IAPX 86/10 Register Model

SYSTEM TIMING — MINIMUM SYSTEM

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal. The trailing (falling) edge of this signal is used to latch the address information, which is valid on the local bus at this time, into the 8282/8283 latch. The BHE and A0 signals address the low, high, or both bytes. From T_1 to T_2 , the MIO signal indicates a memory or I/O operation. At T_2 the address is removed from the local bus and the bus goes to a high impedance state. The read control signal is also asserted at T_2 . The read (RD) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias 0°C to 70°C
 Storage Temperature -65°C to +150°C
 Voltage on Any Pin with Respect to Ground -1.0 to +7V
 Power Dissipation 2.5 Watt

*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS (8086: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 10\%$)
 (8086-1: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 5\%$)
 (8086-2: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 5\%$)

| Symbol | Parameter | Min. | Max. | Units | Test Conditions |
|----------|--|------|-------------------|---------------|---|
| V_{IL} | Input Low Voltage | -0.5 | +0.8 | V | |
| V_{IH} | Input High Voltage | 2.0 | $V_{CC} + 0.5$ | V | |
| V_{OL} | Output Low Voltage | | 0.45 | V | $I_{OL} = 2.5\text{ mA}$ |
| V_{OH} | Output High Voltage | 2.4 | | V | $I_{OH} = -400\ \mu\text{A}$ |
| I_{CC} | Power Supply Current: 8086-1 8086-2 | | 340 360 350 | mA | $T_A = 25^\circ\text{C}$ |
| I_U | Input Leakage Current | | ± 10 | μA | $0\text{V} \leq V_{IN} \leq V_{CC}$ |
| I_{LO} | Output Leakage Current | | ± 10 | μA | $0.45\text{V} \leq V_{OUT} \leq V_{CC}$ |
| V_{CL} | Clock Input Low Voltage | -0.5 | +0.6 | V | |
| V_{CH} | Clock Input High Voltage | 3.9 | $V_{CC} + 1.0$ | V | |
| C_{IN} | Capacitance of Input Buffer (All input except $AD_0 - AD_{15}$, $RD\overline{ST}$) | | 15 | pF | $f_c = 1\text{ MHz}$ |
| C_{IO} | Capacitance of I/O Buffer ($AD_0 - AD_{15}$, $RD\overline{ST}$) | | 15 | pF | $f_c = 1\text{ MHz}$ |

A.C. CHARACTERISTICS (8086: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 10\%$)
 (8086-1: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 5\%$)
 (8086-2: $T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = 5\text{V} \pm 5\%$)

MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

| Symbol | Parameter | 8086 | | 8086-1 (Preliminary) | | 8086-2 | | Units | Test Conditions |
|--------------|--|------|------|----------------------|------|--------|------|-------|-------------------|
| | | Min. | Max. | Min. | Max. | Min. | Max. | | |
| T_{CLK} | CLK Cycle Period | 200 | 500 | 100 | 500 | 125 | 500 | ns | |
| T_{LCLK} | CLK Low Time | 115 | | 53 | | 68 | | ns | |
| T_{HCLK} | CLK High Time | 88 | | 38 | | 44 | | ns | |
| T_{CH1CH2} | CLK Rise Time | | 10 | | 10 | | 10 | ns | From 1.0V to 3.5V |
| T_{FCLK1} | CLK Fall Time | | 10 | | 10 | | 10 | ns | From 3.5V to 1.0V |
| T_{DQCL} | Data in Setup Time | 30 | | 5 | | 20 | | ns | |
| T_{DQCLX} | Data in Hold Time | 10 | | 10 | | 10 | | ns | |
| T_{RDVCL} | RDY Setup Time into EBM_A (See Notes 1, 2) | 35 | | 35 | | 35 | | ns | |
| T_{CLD1X} | RDY Hold Time into EBM_A (See Notes 1, 2) | 0 | | 0 | | 0 | | ns | |
| T_{RDVCH} | READY Setup Time into 8086 | 118 | | 53 | | 68 | | ns | |
| T_{CHRYX} | READY Hold Time into 8086 | 30 | | 20 | | 20 | | ns | |
| T_{RDVCL} | READY Inactive to CLK (See Note 3) | -8 | | -10 | | 8 | | ns | |
| T_{RDVCH} | HOLD Setup Time | 35 | | 2L | | 20 | | ns | |
| T_{RDVCH} | INTR, INIT, TEST Setup Time (See Note 2) | 30 | | 15 | | 15 | | ns | |
| T_{ILH} | Input Rise Time (Except CLK) | | 20 | | 20 | | 20 | ns | From 0.8V to 2.0V |
| T_{FHL} | Input Fall Time (Except CLK) | | 12 | | 12 | | 12 | ns | From 2.0V to 0.8V |

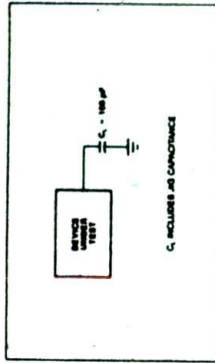
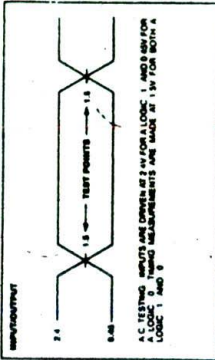
A.C. CHARACTERISTICS (Continued)

TIMING RESPONSES

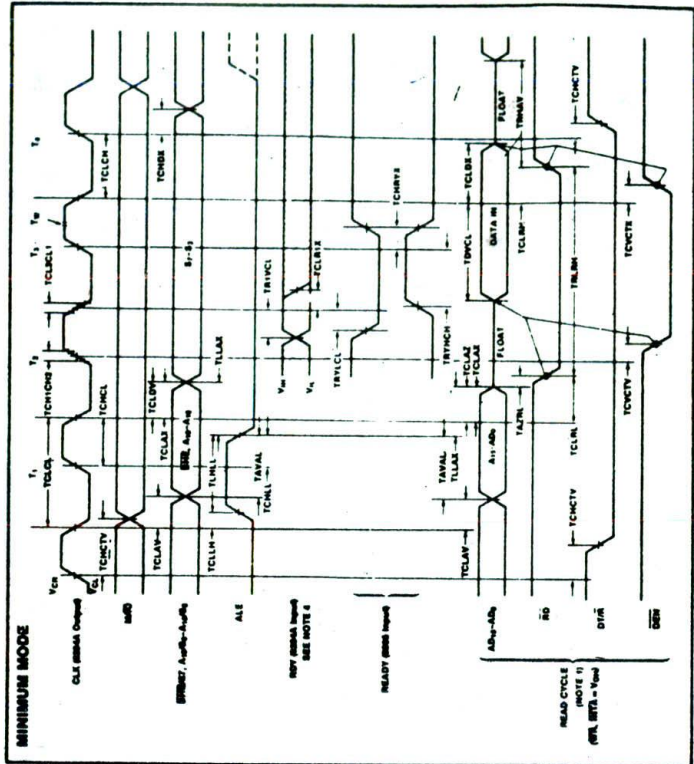
| Symbol | Parameter | ns | | ns (Preliminary) | | ns | | Units | Test Conditions |
|--------|------------------------------------|------|------|------------------|------|----------|------|-------|--|
| | | Min. | Max. | Min. | Max. | Min. | Max. | | |
| TCLAV | Address Valid Delay | 10 | 110 | 10 | 50 | 10 | 60 | ns | |
| TCLAX | Address Hold Time | 10 | | | | 10 | | ns | |
| TCLAE | Address Float Delay | | 60 | 10 | 40 | TCLAX | 50 | ns | |
| TLKLL | ALE Width | | 20 | | 10 | TOLCH-10 | | ns | |
| TALL | ALE Active Delay | 80 | | | 40 | | 50 | ns | |
| TAL | ALE Inactive Delay | 85 | | | 45 | | 55 | ns | |
| TLAX | Address Hold Time to ALE Inactive | | 10 | | 10 | TKNCL-10 | | ns | |
| TCLDV | Data Valid Delay | 10 | 110 | 10 | 50 | | 10 | ns | C _L = 20, 100 pF for all 8288 Outputs (to be taken to 8288 unit load) |
| TCLDH | Data Hold Time | 10 | | 10 | | | 10 | ns | |
| TWHD | Data Hold Time After WR | | 30 | | 25 | TOLCH-30 | | ns | |
| TCVCTV | Control Active Delay 1 | 10 | 110 | 10 | 50 | | 10 | ns | |
| TCVCTV | Control Active Delay 2 | 10 | 110 | 10 | 45 | | 10 | ns | |
| TCVCTI | Control Inactive Delay | 10 | 110 | 10 | 50 | | 10 | ns | |
| TAZRL | Address Float to READ Active | 0 | | 0 | | | 0 | ns | |
| TCLR | RD Active Delay | 10 | 165 | 10 | 70 | | 10 | ns | |
| TCLRH | RD Inactive Delay | 10 | 150 | 10 | 60 | | 10 | ns | |
| TBRV | RD Inactive to Next Address Active | | 45 | | 35 | TCLCL-40 | | ns | |
| TCLMV | MEMA Valid Delay | 10 | 160 | 10 | 60 | | 10 | ns | |
| TRLRH | RD Width | 21 | 75 | 21 | 40 | 21 | 50 | ns | |
| TWLWH | WR Width | 21 | 60 | 21 | 35 | 21 | 40 | ns | |
| TAVL | Address Valid to ALE Low | | 60 | | 35 | TOLCH-40 | | ns | |
| TOLH | Output Rise Time | 20 | | 20 | | | 20 | ns | From 0 BV to 2 DV |
| TOHL | Output Fall Time | 12 | | 12 | | | 12 | ns | From 2 DV to 0 BV |

- NOTES
 1 Signal at 8284A shown for reference only
 2 Setup requirement for asynchronous signal only to guarantee recognition at next CLK
 3 Applies only to T2 state (8 ns into T3)

A.C. TESTING INPUT, OUTPUT WAVEFORM



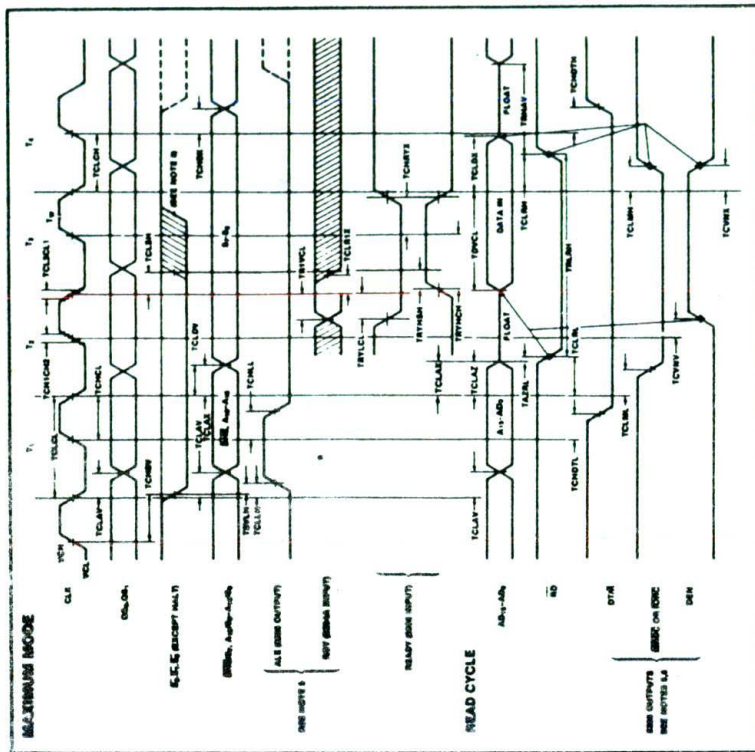
WAVEFORMS



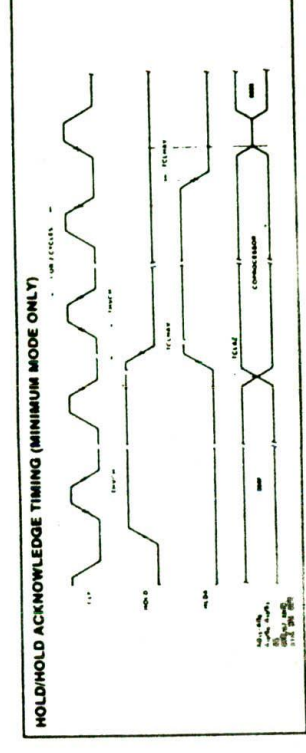
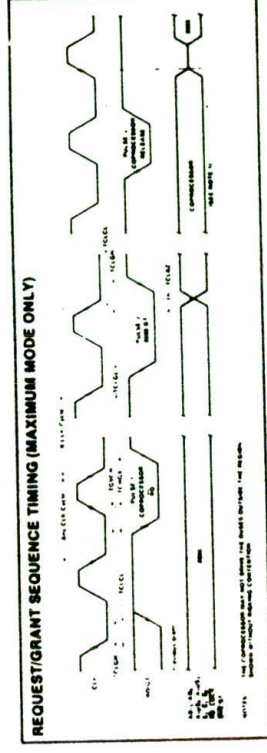
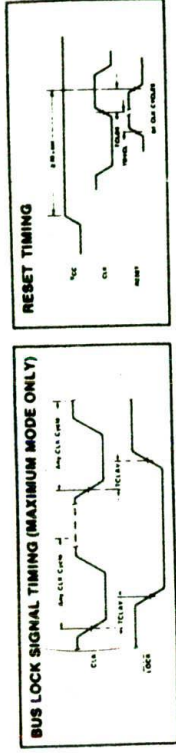
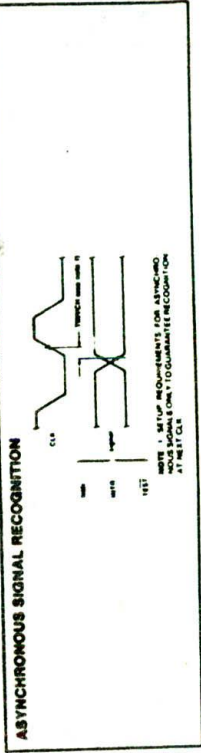
A.C. CHARACTERISTICS (Continued)
TIMING RESPONSES

| Symbol | Parameter | ns | | ns | | ns | | ns | | Total Contribution |
|--------|---|------|------------|------------|------------|------------|------------|------------|------|--------------------|
| | | Min. | Max. | Min. | Max. | Min. | Max. | Min. | Max. | |
| TCLM | Command Active Delay (See Note 1) | 10 | 35 | 10 | 30 | | | | | |
| TCLM | Command Inactive Delay (See Note 1) | 10 | 30 | 10 | 30 | | | | | |
| TRYSH | READY Active to Status Passes (See Note 3) | | 110 | | 45 | | | | | |
| TCHSV | Status Active Delay | 10 | 110 | 10 | 45 | | | | | |
| TCLSH | Status Inactive Delay | 10 | 130 | 10 | 55 | 10 | 70 | | | |
| TCLAV | Address Valid Delay | 10 | 110 | 10 | 50 | 15 | 30 | | | |
| TCLAJ | Address Hold Time | 10 | | 10 | | | | | | |
| TCLAJ | Address Float Delay | | TCLAJ | | 40 | TCLAJ | 50 | | | |
| TSVLIH | Status Valid to ALE High (See Note 1) | | 15 | | 15 | | 15 | | | |
| TSVMLH | Status Valid to MCE High (See Note 1) | | 15 | | 15 | | 15 | | | |
| TCLLW | CLK Low to ALE Valid (See Note 1) | | 15 | | 15 | | 15 | | | |
| TCLMCH | CLK Low to MCE High (See Note 1) | | 15 | | 15 | | 15 | | | |
| TCHLL | ALE Inactive Delay (See Note 1) | | 15 | | 15 | | 15 | | | |
| TCLMCL | MCE Inactive Delay (See Note 1) | | 15 | | 15 | | 15 | | | |
| TCLDV | Data Valid Delay | 10 | 110 | 10 | 50 | 10 | 30 | | | |
| TCHDA | Data Hold Time | 10 | | 10 | | | | | | |
| TCHVA | Control Active Delay (See Note 1) | 5 | 45 | 5 | 45 | 5 | 45 | | | |
| TCHVI | Control Inactive Delay (See Note 1) | 10 | 45 | 10 | 45 | 10 | 45 | | | |
| TAPRL | Address Float to Read Active | 0 | | 0 | | | | | | |
| TCLRV | RD Active Delay | 10 | 185 | 10 | 70 | 10 | 100 | | | |
| TCLRH | RD Inactive Delay | 10 | 150 | 10 | 80 | 10 | 80 | | | |
| TRHAV | RD Inactive to Next Address Active | | | TCLCL - 45 | | TCLCL - 35 | | TCLCL - 40 | | |
| TCHDTL | Direction Control Active Delay (See Note 1) | | 50 | | 50 | | 50 | | | |
| TCHDTH | Direction Control Inactive Delay (See Note 1) | | 30 | | 30 | | 30 | | | |
| TCLGL | DT Active Delay | 0 | 85 | 0 | 45 | 0 | 50 | | | |
| TCLGH | DT Inactive Delay | 0 | 85 | 0 | 45 | 0 | 50 | | | |
| TRLPH | RD Width | | TCLCL - 75 | | TCLCL - 40 | | TCLCL - 50 | | | |
| TOLCH | Output Rise Time | | 20 | | 20 | | 20 | | | From 0.8V to 2.0V |
| TCHOL | Output Fall Time | | 12 | | 12 | | 12 | | | From 2.0V to 0.8V |

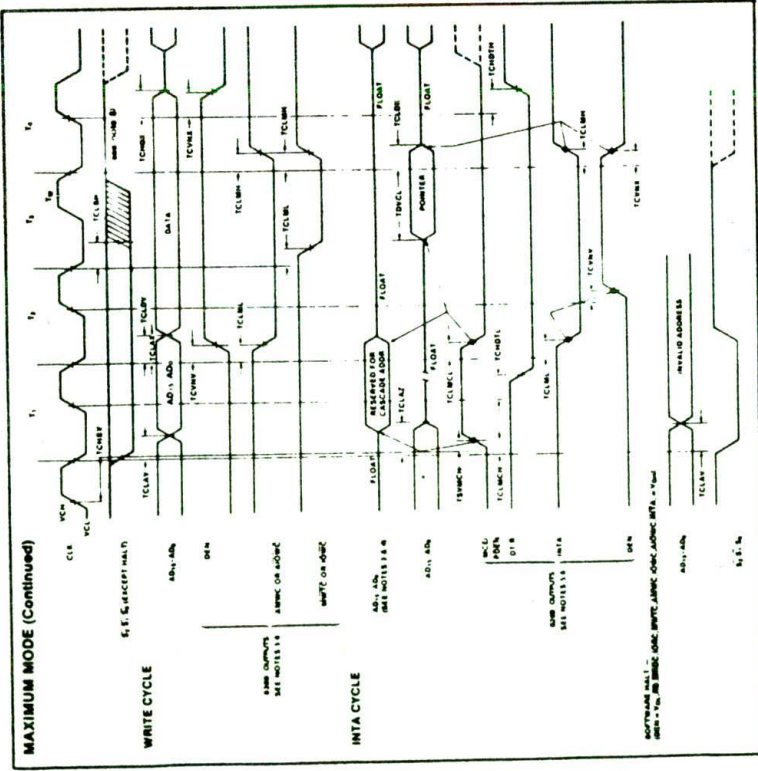
t₁ ~ 20-100 pF for an 8086/286 pin (see note 1) for each pin (see note 1)

WAVEFORMS
MULTIBUS MODE


WAVEFORMS (Continued)



WAVEFORMS (Continued)



- NOTES:**
- 1 All signals switch between V_{OH} and V_{OL} unless otherwise specified.
 - 2 RDY is sampled near the end of T_1 , T_2 , T_3 , T_4 to detect the machine state to be inserted.
 - 3 Cascade address is valid between T_1 and T_2 of the first INTA cycle.
 - 4 Two INTA cycles are valid between T_1 and T_2 of the second INTA cycle.
 - 5 Signals at $E288A$ or $E288B$ are shown for reference only.
 - 6 The issuance of the $E288$ command and control signals ($MRDC$, $MWTC$, $AMWTC$, $IORC$, $IOWC$, $ADWOC$, $INTA$ and DEH) lags the active high $E288$ CEN .
 - 7 All timing measurements are made at 1.5V unless otherwise noted.
 - 8 Status inactive in state just prior to T_4 .

Table 2. In-Instruction Set Summary

| 1 | | 2 | |
|--|--------------------------|------------------|--------------------------|
| Instruction Code | Mnemonic and Description | Instruction Code | Mnemonic and Description |
| DATA TRANSFER | | 76543210 | 76543210 |
| MOV - Move: | | 1000100w | mod reg r/m |
| Immediate to Register/Memory | | 1100011w | mod 000r/m |
| Immediate to Register | | 1011w reg | data data # w = 1 |
| Memory to Accumulator | | 1010000w | addr-low addr-high |
| Accumulator to Memory | | 1010001w | addr-low addr-high |
| Register/Memory to Segment Register | | 10001110 | mod 0 reg r/m |
| Segment Register to Register/Memory | | 10001100 | mod 0 reg r/m |
| PUSH - Push: | | 11111111 | mod 110r/m |
| Register/Memory | | 01010 reg | |
| Register | | 000 reg 110 | |
| Segment Register | | | |
| POP - Pop: | | 10001111 | mod 000r/m |
| Register/Memory | | 01011 reg | |
| Register | | 000 reg 111 | |
| Segment Register | | | |
| ICNG - Exchange: | | 1000011w | mod reg r/m |
| Register/Memory with Register | | 10010 reg | |
| Register with Accumulator | | | |
| IN - Input from: | | 1110010w | port |
| Fixed Port | | 1110110w | port |
| Variable Port | | | |
| OUT - Output to: | | 1110011w | port |
| Fixed Port | | 1110111w | port |
| Variable Port | | | |
| XLAT - Translate Byte to AL | | 1010111 | |
| LEA - Load EA to Register | | 10001101 | mod reg r/m |
| LDS - Load Pointer to DS | | 11000101 | mod reg r/m |
| LES - Load Pointer to ES | | 11000100 | mod reg r/m |
| LAHF - Load AH with Flags | | 10011111 | |
| SAHF - Store AH into Flags | | 10011110 | |
| PUSHF - Push Flags | | 10011100 | |
| POPF - Pop Flags | | | |
| ARITHMETIC | | | |
| ADD - Add: | | 0000000w | mod reg r/m |
| Register/Memory with Register to Either | | 1000000w | mod 000r/m |
| Immediate to Register/Memory | | 0000010w | data data # w = 1 |
| Immediate to Accumulator | | | |
| ADC - Add with Carry: | | 0001000w | mod reg r/m |
| Register/Memory with Register to Either | | 1000000w | mod 010r/m |
| Immediate to Register/Memory | | 0001010w | data data # w = 1 |
| Immediate to Accumulator | | | |
| INC - Increment: | | 1111111w | mod 000r/m |
| Register/Memory | | 01000 reg | |
| Register | | | |
| 76543210 | | 00110111 | |
| AAA - ASCII Adjust for Add | | 00100111 | |
| DAA - Decimal Adjust for Add | | | |
| SUB - Subtract: | | 0010100w | mod reg r/m |
| Register/Memory and Register to Either | | 1000000w | mod 101r/m |
| Immediate from Register/Memory | | 0010110w | data data # w = 1 |
| Immediate from Accumulator | | | |
| SBB - Subtract with Borrow | | 0001100w | mod reg r/m |
| Register/Memory and Register to Either | | 1000000w | mod 011r/m |
| Immediate from Register/Memory | | 0001110w | data data # w = 1 |
| Immediate from Accumulator | | | |
| DEC - Decrement: | | 1111111w | mod 001r/m |
| Register/Memory | | 01001 reg | |
| Register | | 1111011w | mod 011r/m |
| NEG - Change sign | | | |
| CBP - Compare: | | 0011100w | mod reg r/m |
| Register/Memory and Register | | 1000000w | mod 111r/m |
| Immediate with Register/Memory | | 0011110w | data data # w = 1 |
| Immediate with Accumulator | | | |
| AAS - ASCII Adjust for Subtract | | 00101111 | |
| DAS - Decimal Adjust for Subtract | | | |
| IMUL - Multiply (Unsigned) | | 1111011w | mod 100r/m |
| IMUL - Integer Multiply (Signed) | | 1111011w | mod 101r/m |
| AMM - ASCII Adjust for Multiply | | 11010100 | 00001010 |
| DIV - Divide (Unsigned) | | 1111011w | mod 110r/m |
| IDIV - Integer Divide (Signed) | | 1111011w | mod 111r/m |
| AAD - ASCII Adjust for Divide | | 11010101 | 00001010 |
| CBW - Convert Byte to Word | | 10011000 | |
| CWD - Convert Word to Double Word | | 10011001 | |
| LOGIC | | | |
| NOT - Invert | | 1111011w | mod 101r/m |
| SHL/SAL - Shift Logical/Arithmetic Left | | 1101000w | mod 100r/m |
| SHR - Shift Logical Right | | 1101000w | mod 101r/m |
| SAR - Shift Arithmetic Right | | 1101000w | mod 111r/m |
| RCL - Rotate Left | | 1101000w | mod 000r/m |
| ROR - Rotate Right | | 1101000w | mod 001r/m |
| RCL - Rotate Through Carry Flag Left | | 1101000w | mod 010r/m |
| RCR - Rotate Through Carry Right | | 1101000w | mod 011r/m |
| AND - And: | | | |
| Register/Memory and Register to Either | | 0010000w | mod reg r/m |
| Immediate to Register/Memory | | 1000000w | mod 100r/m |
| Immediate to Accumulator | | 0010010w | data data # w = 1 |
| TEST - And Function to Flags; No Result: | | | |
| Register/Memory and Register | | 1000010w | mod reg r/m |
| Immediate Data and Register/Memory | | 1111011w | mod 000r/m |
| Immediate Data and Accumulator | | 1010100w | data data # w = 1 |
| 76543210 | | 76543210 | 76543210 |
| 76543210 | | 76543210 | 76543210 |
| 1000100w | mod reg r/m | | |
| 1100011w | mod 000r/m | data | data # w = 1 |
| 1011w reg | data | data # w = 1 | |
| 1010000w | addr-low | addr-high | |
| 1010001w | addr-low | addr-high | |
| 10001110 | mod 0 reg r/m | | |
| 10001100 | mod 0 reg r/m | | |
| 11111111 | mod 110r/m | | |
| 01010 reg | | | |
| 000 reg 110 | | | |
| 10001111 | mod 000r/m | | |
| 01011 reg | | | |
| 000 reg 111 | | | |
| 1000011w | mod reg r/m | | |
| 10010 reg | | | |
| 1110010w | port | | |
| 1110110w | port | | |
| 1110011w | port | | |
| 1110111w | port | | |
| 1010111 | | | |
| 10001101 | mod reg r/m | | |
| 11000101 | mod reg r/m | | |
| 11000100 | mod reg r/m | | |
| 10011111 | | | |
| 10011110 | | | |
| 10011101 | | | |
| 0000000w | mod reg r/m | | |
| 1000000w | mod 000r/m | data | data # w = 01 |
| 0000010w | data | data # w = 1 | |
| 0001000w | mod reg r/m | | |
| 1000000w | mod 010r/m | data | data # w = 01 |
| 0001010w | data | data # w = 1 | |
| 1111111w | mod 000r/m | | |
| 01000 reg | | | |

3

3 Measurand and Description

Instruction Code

| | | | |
|--|----------|---------------|---------------|
| OR - Or | 76543210 | 76843210 | 76543210 |
| Reg / Memory and Register to Either Immediate to Register / Memory | 0000100w | mod reg / r/m | |
| Immediate to Accumulator | 1000000w | mod 011/r/m | data if w = 1 |
| XCOR - Exclusive or | 0000110w | data | data if w = 1 |
| Reg / Memory and Register to Either Immediate to Register / Memory | 0011000w | mod reg / r/m | data if w = 1 |
| Immediate to Accumulator | 1000000w | mod 110/r/m | data if w = 1 |
| Immediate to Accumulator | 0011010w | data | data if w = 1 |
| STRING MANIPULATION | | | |
| REP - Repeat | 1111001z | | |
| MOVB - Move Byte/Word | 1010010w | | |
| CMPS - Compare Byte/Word | 1010011w | | |
| SCAS - Scan Byte/Word | 1010111w | | |
| LODS - Load Byte/Word to AL/AX | 1010110w | | |
| STOS - Store Byte/Word from AL/A | 1010101w | | |
| CONTROL TRANSFER | | | |
| CALL - Call | 11110000 | disp-low | disp-high |
| Direct within Segment | 11111111 | mod 010/r/m | |
| Indirect within Segment | 10011010 | offset-low | offset-high |
| Direct Intersegment | | sig-low | sig-high |
| Indirect Intersegment | 11111111 | mod 011/r/m | |
| JMP - Unconditional Jump | 11101001 | disp-low | disp-high |
| Direct within Segment | 11101011 | disp | |
| Indirect within Segment | 11111111 | mod 100/r/m | |
| Direct Intersegment | 11101010 | offset-low | offset-high |
| Indirect Intersegment | | sig-low | sig-high |
| Indirect Intersegment | 11111111 | mod 011/r/m | |
| RET - Return from CALL | 11000011 | | |
| Within Segment | 11000010 | data-low | data-high |
| Within Seg Adding limited to SP Intersegment | 11001011 | data-low | data-high |
| Intersegment Adding Immediate to SP | 01110100 | disp | |
| JL/JZ - Jump on Less/Zero | 01111100 | disp | |
| JL/JGE - Jump on Less/Not Greater or Equal | 01111110 | disp | |
| JLE/JNG - Jump on Less or Equal/Not Greater | 01111110 | disp | |
| JM/JNAE - Jump on Below/Not Above or Equal | 01110010 | disp | |
| JM/JMA - Jump on Below or Equal/Not Above | 01110110 | disp | |
| JP/JPE - Jump on Parity/Parity Even | 01110000 | disp | |
| JO - Jump on Overflow | 01111000 | disp | |
| JS - Jump on Sign | 01110101 | disp | |
| JM/JNE - Jump on Not Equal/Not Zero | 01111101 | disp | |
| JM/JDE - Jump on Not Less/Greater or Equal | 01111111 | disp | |
| JM/JD - Jump on Not Less or Equal/Greater | | | |

4

4 Measurand and Description

Instruction Code

| | | | |
|---|----------|-------------|--|
| JMB/JAE - Jump on Not Below/Above or Equal | 01110011 | disp | |
| JMB/JA - Jump on Not Below or Equal/Above | 01110111 | disp | |
| JMP/JPO - Jump on Not Par/Par Odd | 01111011 | disp | |
| JMO - Jump on Not Overflow | 01110001 | disp | |
| JMS - Jump on Not Sign | 01111001 | disp | |
| LOOP - Loop CX Times | 11100010 | disp | |
| LOOPZ/LOOPE - Loop While Zero/Equal | 11100001 | disp | |
| LOOPNZ/LOOPNE - Loop While Not Zero/Not Equal | 11100000 | disp | |
| JCZ - Jump on CX Zero | 11100011 | disp | |
| INT - Interrupt | 11001101 | . type | |
| Type Specified | 11001100 | | |
| Type 3 | 11001110 | | |
| IRET - Interrupt on Overflow | 11001110 | | |
| IRETD - Interrupt Return | 11001111 | | |
| PROCESSOR CONTROL | | | |
| CLC - Clear Carry | 11111000 | | |
| CMC - Complement Carry | 11110101 | | |
| STC - Set Carry | 11111001 | | |
| CLD - Clear Direction | 11111100 | | |
| STD - Set Direction | 11111101 | | |
| CLE - Clear Interrupt | 11111010 | | |
| STI - Set Interrupt | 11111011 | | |
| HLT - Halt | 11110100 | | |
| WAIT - Wait | 10011011 | | |
| ESC - Escape (to Extenu. Devices) | 11011xxx | mod xxx/r/m | |
| LOCK - Bus Lock Prefix | 11110000 | | |

if s w = 01 then 16 bit of immediate data form the operand and
if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1, if v = 1 then "count" in (CL) x = don't care
z is used for string primitives for comparison with ZF FLAG

SEGMENT OVERRIDE PREFIX
001 reg 110

REG is assigned according to the following table

| 16-Bit (w = 1) | 8-Bit (w = 0) | Segment |
|----------------|---------------|---------|
| 000 AX | 000 AL | 00 ES |
| 001 CX | 001 CL | 01 CS |
| 010 DX | 010 DL | 10 SS |
| 011 BX | 011 BL | 11 DS |
| 100 SP | 100 AH | |
| 101 BP | 101 CH | |
| 110 SI | 110 DH | |
| 111 DI | 111 BH | |

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:
FLAGS = x.x.x.x.(OF)(DF)(IF)(TF)(SF)(ZF).X.(AF).X.(CF)

NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
Above/below refers to unsigned value
Greater = more positive.
Less = less positive (more negative) signed values
if d = 1 then "reg. if d = 0 then "from" reg
if w = 1 then word instruction, if w = 0 then byte instruction
if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0', disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then EA = (BP) + (SI) + DISP
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)
*except if mod = 00 and r/m = 110 then EA = disp-high; disp-low.

APPENDIX B

8086/8088 Instructions

Notes for 8086/8088 Instructions

The individual instruction descriptions are shown by a format box such as the following.

| | | | | |
|--------|----------|--|--|------|
| Opcode | m/op/r/m | | | Data |
|--------|----------|--|--|------|

These are byte-wise representations of the object code generated by the assembler and are interpreted as follows:

- Opcode is the 8-bit opcode for the instruction. The actual opcode generated is defined in the "Opcode" column of the instruction table that follows each format box.
- m/op/r/m is the byte that specifies the operands of the instruction. It contains a 2-bit mode field (m), a 3-bit register field (op), and a 3-bit register or memory (r/m) field.
- Dashed blank boxes following the m/op/r/m box are for any displacement required by the mode field.
- Data is for a byte of immediate data.
- A dashed blank box following a Data box is used whenever the immediate operand is a word quantity.

Operand Summary

"reg" field Bit Assignments:

| Word Operand | Byte Operand | Segment |
|--------------|--------------|---------|
| 000 AX | 000 AL | 00 ES |
| 001 CX | 001 CL | 01 CS |
| 010 DX | 010 DL | 10 SS |
| 011 BX | 011 BL | 11 DS |
| 100 SP | 100 AH | |
| 101 BP | 101 CH | |
| 110 SI | 110 DH | |
| 111 DI | 111 BH | |

Second Instruction Byte Summary

| |
|-------------|
| mod xxx r/m |
|-------------|

| mod | Displacement |
|-----|---|
| 00 | DISP = 0; dsp-low and dsp-high are absent |
| 01 | DISP = dsp-low sign-extended to 16-bits, dsp-high is absent |
| 10 | DISP = dsp-high; dsp-low absent |
| 11 | r/m is treated as a "reg" field |

| r/m | Operand Address |
|-----|--------------------|
| 000 | (BX) + (SI) + DISP |
| 001 | (BX) + (DI) + DISP |
| 010 | (BP) + (SI) + DISP |
| 011 | (BP) + (DI) + DISP |
| 100 | (SI) + DISP |
| 101 | (DI) + DISP |
| 110 | (BP) + DISP |
| 111 | (BX) + DISP |

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = dsp-high; dsp-low

Flags

- AF: AUXILIARY CARRY — BCD
- CF: CARRY FLAG
- DF: DIRECTION FLAG (STRINGS)
- IF: INTERRUPT ENABLE FLAG
- OF: OVERFLOW FLAG (CF, SF)
- PF: PARITY FLAG
- SF: SIGN FLAG
- TF: TRAP (SINGLE STEP) FLAG
- ZF: ZERO FLAG

Instructions that reference the Reg register file as a 16-bit object use the symbol FLAGS to represent the file

| | | | | | | | | | | | | | | | | | | |
|----|---|---|---|---|---|---|---|----|----|----|----|----|---|----|---|----|---|----|
| 15 | 8 | | | | | | | | | | | | | | | | | |
| X | X | X | X | X | X | X | X | DF | IF | TF | SF | ZF | X | AF | X | PF | X | CF |

X = Don't Care

Segment Override Prefix

| |
|-----------------|
| 0 0 1 reg 1 1 0 |
|-----------------|

Timing: 2 clocks

Use of Segment Override

| Operand Register | Default | With Override Prefix |
|--|---------|----------------------|
| IP (code address) | CS | Never |
| SP (stack address) | SS | Never |
| BP (stack address or stack marker) | SS | BP + DS or ES, or CS |
| SI or DI (not incl. strings) | DS | ES, SS, or CS |
| SI (implicit source addr. for strings) | DS | ES, SS, or CS |
| DI (implicit dest addr. for strings) | ES | Never |

Operand Address (EA) Timing (Clocks):

Add 4 clocks for word operands at ODD ADDRESSES

Immed Offset = 8

Base (BX, BP, SI, DI) = 5

Base + DISP = 9

Base + Index (BP + DI, BX + SI) = 7

Base + Index (BP + SI, BX + DI) = 8

Base + Index (BP + DI, BX + SI) + DISP = 11

Base + Index (BP + SI, BX + DI) + DISP = 12

AAA = ASCII Adjust for Addition

| |
|--------|
| Opcode |
|--------|

| Opcode | Clocks | Operation |
|--------|--------|----------------------|
| 37 | 4 | adjust AL, flags, AH |

AAD = ASCII Adjust for Division

| |
|-------------|
| Long—Opcode |
|-------------|

| Opcode | Clocks | Operation |
|--------|--------|---------------------------------|
| D5, 0A | 60 | Adjust AL, AH prior to division |

AAM = ASCII Adjust for Multiplication

| |
|-------------|
| Long—Opcode |
|-------------|

| Opcode | Clocks | Operation |
|--------|--------|------------------------------------|
| D4, 0A | 83 | Adjust AL, AH after multiplication |

AAS = ASCII Adjust for Subtraction

| |
|--------|
| Opcode |
|--------|

| Opcode | Clocks | Operation |
|--------|--------|----------------------|
| 3F | 4 | adjust AL, flags, AH |

ADC = Integer Add with Carry

Memory/Reg + Reg

| | | | |
|--------|-------------|--|--|
| Opcode | mod reg r/m | | |
|--------|-------------|--|--|

| | Opcode | Clocks | Operation |
|------|--------|---------|----------------------------|
| Byte | 12 | 3 | Reg8 - CF + Reg8 + Reg8 |
| | 12 | 9 - EA | Reg8 - CF + Reg8 + Mem8 |
| | 10 | 16 - EA | Mem8 - CF + Mem8 + Reg8 |
| Word | 13 | 3 | Reg16 - CF + Reg16 + Reg16 |
| | 13 | 9 - EA | Reg16 - CF + Reg16 + Mem16 |
| | 11 | 16 - EA | Mem16 - CF + Mem16 + Reg16 |

Immed to AX/AL

| | | |
|--------|------|--|
| Opcode | Data | |
|--------|------|--|

| | Opcode | Clocks | Operation |
|------|--------|--------|------------------------|
| Byte | 14 | 4 | AL ← CF + AL + Immed8 |
| Word | 15 | 4 | AX ← CF + AX + Immed16 |

Immed to Memory/Reg

| Opcode | mod 010 r/m | Data |
|--------|-------------|------|
| | | |

| | Opcode | Clocks | Operation |
|------|--------|---------|------------------------------|
| Byte | 80 | 4 | Reg8 ← CF + Reg8 + Immed8 |
| | 80 | 17 + EA | Mem8 ← CF + Mem8 + Immed8 |
| Word | 81 | 4 | Reg16 ← CF + Reg16 + Immed16 |
| | 81 | 17 + EA | Mem16 ← CF + Mem16 + Immed16 |
| | 83 | 4 | Reg16 ← CF + Reg16 + Immed8 |
| | 83 | 17 + EA | Mem16 ← CF + Mem16 + Immed8 |

ADD = Integer Addition

Memory/Reg + Reg

| Opcode | mod reg r/m |
|--------|-------------|
| | |

| | Opcode | Clocks | Operation |
|------|--------|---------|-----------------------|
| Byte | 02 | 3 | Reg8 ← Reg8 + Reg8 |
| | 02 | 9 - EA | Reg8 ← Reg8 + Mem8 |
| | 00 | 16 - EA | Mem8 ← Mem8 + Reg8 |
| Word | 03 | 3 | Reg16 ← Reg16 + Reg16 |
| | 03 | 9 - EA | Reg16 ← Reg16 + Mem16 |
| | 01 | 16 - EA | Mem16 ← Mem16 + Reg16 |

Immed to AX/AL

| Opcode | Data |
|--------|------|
| | |

| | Opcode | Clocks | Operation |
|--|--------|--------|-------------------|
| | 04 | 4 | AL ← AL + Immed8 |
| | 05 | 4 | AX ← AX + Immed16 |

Immed to Memory/Reg

| Opcode | mod 000 r/m | Data |
|--------|-------------|------|
| | | |

| | Opcode | Clocks | Operation |
|------|--------|---------|-------------------------|
| Byte | 80 | 4 | Reg8 ← Reg8 + Immed8 |
| | 80 | 17 + E/ | Mem8 ← Mem8 + Immed8 |
| Word | 81 | 4 | Reg16 ← Reg16 + Immed16 |
| | 81 | 17 + EA | Mem16 ← Mem16 + Immed16 |
| | 83 | 4 | Reg16 ← Reg16 + Immed8 |
| | 83 | 17 + EA | Mem16 ← Mem16 + Immed8 |

AND = Logical AND

Memory/Reg with Reg

| Opcode | mod reg r/m |
|--------|-------------|
| | |

| | Opcode | Clocks | Operation |
|------|--------|---------|-------------------------|
| Byte | 22 | 3 | Reg8 ← Reg8 AND Reg8 |
| | 22 | 9 - EA | Reg8 ← Reg8 AND Mem8 |
| | 20 | 16 - EA | Mem8 ← Mem8 AND Reg8 |
| Word | 23 | 3 | Reg16 ← Reg16 AND Reg16 |
| | 23 | 9 - EA | Reg16 ← Reg16 AND Mem16 |
| | 21 | 16 - EA | Mem16 ← Mem16 AND Reg16 |

Immed to AX/AL

| Opcode | Data |
|--------|------|
| | |

| | Opcode | Clocks | Operation |
|------|--------|--------|---------------------|
| Byte | 24 | 4 | AL ← AL AND Immed8 |
| Word | 25 | 4 | AX ← AX AND Immed16 |

Immed to Memory Reg

| Opcode | mod 100 r/m | Data |
|--------|-------------|------|
| | | |

| | Opcode | Clocks | Operation |
|------|--------|---------|---------------------------|
| Byte | 80 | 4 | Reg8 ← Reg8 AND Immed8 |
| | 80 | 17 - EA | Mem8 ← Mem8 AND Immed8 |
| Word | 81 | 4 | Reg16 ← Reg16 AND Immed16 |
| | 81 | 17 - EA | Mem16 ← Mem16 AND Immed16 |

CALL = Call

Within segment or group, IP relative

| Opcode | Displ. | DispH |
|--------|--------|-------|
| | | |

| | Opcode | Clocks | Operation |
|--|--------|--------|---------------------------------------|
| | E8 | 19 | IP ← IP + Disp16 — (SP) ← return link |

Within segment or group, Indirect

| Opcode | mod 010 r/m |
|--------|-------------|
| | |

| | Opcode | Clocks | Operation |
|--|--------|---------|---------------------------------|
| | FF | 16 | IP ← Reg16 — (SP) ← return link |
| | FF | 21 + EA | IP ← Mem16 — (SP) ← return link |

Inter-segment or group, Direct

| Opcode | offset | offset | segbase | segbase | segbase |
|--------|--------|--------|---------|---------|---------|
| | | | | | |

| | Opcode | Clocks | Operation |
|--|--------|--------|-----------------------------|
| | 9A | 28 | CS ← segbase IP ← offset |

Inter-segment or group, Indirect

| Opcode | mod 011 r/m |
|--------|-------------|
| | |

| | Opcode | Clocks | Operation |
|--|--------|---------|-----------------------------|
| | FF | 37 + EA | CS ← segbase IP ← offset |

CBW = Convert Byte to Word

| Opcode |
|--------|
| |

| | Opcode | Clocks | Operation |
|--|--------|--------|----------------------------------|
| | 98 | 2 | convert byte in AL to word in AX |

CLC = Clear Carry Flag

| Opcode |
|--------|
| |

| | Opcode | Clocks | Operation |
|--|--------|--------|----------------------|
| | F8 | 2 | clear the carry flag |

CLD = Clear Direction Flag

| Opcode |
|--------|
| |

| | Opcode | Clocks | Operation |
|--|--------|--------|----------------------|
| | FC | 2 | clear direction flag |

CLI = Clear Interrupt Enable Flag

| Opcode |
|--------|
| |

| | Opcode | Clocks | Operation |
|--|--------|--------|----------------------|
| | FA | 2 | clear interrupt flag |

CMC = Complement Carry Flag

| Opcode |
|--------|
| |

| | Opcode | Clocks | Operation |
|--|--------|--------|-----------------------|
| | F5 | 2 | complement carry flag |

CMP = Compare Two Operands

Memory/Reg with Reg

| Opcode | mod reg r/m |
|--------|-------------|
| | |

| | Opcode | Clocks | Operation |
|------|--------|--------|-----------------------|
| Byte | 38 | 3 | Flags ← Reg8 - Reg8 |
| | 38 | 9 - EA | Flags ← Reg8 - Mem8 |
| | 3A | 9 - EA | Flags ← Mem8 - Reg8 |
| Word | 39 | 3 | Flags ← Reg16 - Reg16 |
| | 39 | 9 - EA | Flags ← Reg16 - Mem16 |
| | 3B | 9 - EA | Flags ← Mem16 - Reg16 |

Immed to AX/AL

| Opcode | Data |
|--------|------|
| | |

| | Opcode | Clocks | Operation |
|------|--------|--------|--------------------|
| Byte | 3C | 4 | flags AL - Immed8 |
| Word | 3D | 4 | flags AX - Immed16 |

Immed to Memory/Reg



| | Opcode | Clocks | Operation |
|------|--------|---------|-------------------------|
| Byte | 80 | 4 | flags - Reg8 - Immed8 |
| | 80 | 10 - EA | flags - Mem8 - Immed8 |
| Word | 81 | 4 | flags - Reg16 - Immed16 |
| | 81 | 10 + EA | flags - Mem16 - Immed16 |
| | 83 | 4 | flags - Reg16 - Immed8 |
| | 83 | 10 - EA | flags - Mem16 - Immed8 |

CWD = Convert Word to Doubleword



| Opcode | Clocks | Operation |
|--------|--------|---|
| 99 | 5 | convert word in AX to doubleword in DX AX |

DAA = Decimal Adjust for Addition



| Opcode | Clocks | Operation |
|--------|--------|----------------------|
| 27 | 4 | adjust AL, flags, AH |

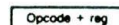
DAS = Decimal Adjust for Subtraction



| Opcode | Clocks | Operation |
|--------|--------|----------------------|
| 2F | 4 | adjust AL, flags, AH |

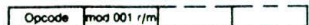
DEC = Decrement by 1

Word Register



| Opcode | Clocks | Operation |
|----------|--------|-------------------|
| 48 + reg | 2 | Reg16 - Reg16 - 1 |

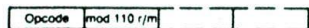
Memory/Byte Register



| | Opcode | Clocks | Operation |
|------|--------|---------|-------------------|
| Byte | FE | 3 | Reg8 - Reg8 - 1 |
| | FE | 15 + EA | Mem8 - Mem8 - 1 |
| Word | FF | 15 + EA | Mem16 - Mem16 - 1 |

DIV = Unsigned Division

Memory/Reg with AX or DX AX



| | Opcode | Clocks | Operation |
|------|--------|----------------|-----------------------|
| Byte | F6 | 80-90 | AH,AL - AX / Reg8 |
| | F6 | (86-96) + EA | AH,AL - AX / Mem8 |
| Word | F7 | 144-162 | DX,AX - DX AX / Reg16 |
| | F7 | (150-168) + EA | DX,AX - DX AX / Mem16 |

ESC = Escape



| Opcode | Clocks | Operation |
|--------|--------|-----------------|
| D8 + i | 8 - EA | data bus ← (EA) |
| D8 + i | 2 | data bus ← (EA) |

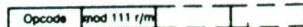
HLT = Halt



| Opcode | Clocks | Operation |
|--------|--------|----------------|
| F4 | 2 | halt operation |

IDIV = Signed Division

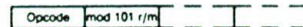
Memory/Reg with AX or DX AX



| | Opcode | Clocks | Operation |
|------|--------|----------------|-----------------------|
| Byte | F6 | 101-112 | AH,AL - AX / Reg8 |
| | F6 | (107-118) + EA | AH,AL - AX / Mem8 |
| Word | F7 | 165-184 | DX,AX - DX AX / Reg16 |
| | F7 | (171-190) + EA | DX,AX - DX AX / Mem16 |

IMUL = Signed Multiplication

Memory/Reg with AL or AX



| | Opcode | Clocks | Operation |
|------|--------|----------------|--------------------|
| Byte | F6 | 80-98 | AX - AL * Reg8 |
| | F6 | (86-104) + EA | AX - AL * Mem8 |
| Word | F7 | 128-154 | DX,AX - AX * Reg16 |
| | F7 | (134-160) + EA | DX,AX - AX * Mem16 |

IN = Input Byte, Word

Fixed port



| | Opcode | Clocks | Operation |
|------|--------|--------|------------|
| Byte | E4 | 10 | AL ← Port8 |
| | E5 | 10 | AX ← Port8 |

Variable port



| | Opcode | Clocks | Operation |
|------|--------|--------|--------------------|
| Word | EC | 8 | AL ← Port16(in DX) |
| | ED | 8 | AX ← Port16(in DX) |

INC = Increment by 1

Word Register



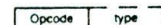
| Opcode | Clocks | Operation |
|----------|--------|-------------------|
| 40 + reg | 2 | Reg16 - Reg16 + 1 |

Memory/Byte Register



| | Opcode | Clocks | Operation |
|------|--------|---------|-------------------|
| Byte | FE | 3 | Reg8 - Reg8 + 1 |
| | FE | 15 - EA | Mem8 - Mem8 + 1 |
| Word | FF | 15 - EA | Mem16 - Mem16 + 1 |

INT INTO = Interrupt



| Opcode | Clocks | Operation |
|--------|---------|-------------------------------------|
| CC | 52 | Interrupt 3 |
| CD | 51 | Interrupt type |
| CE | 53 or 4 | Interrupt4 if FLAGS OF = 1 else NOP |

IRET = Return from Interrupt



| Opcode | Clocks | Operation |
|--------|--------|-----------------------|
| CF | 24 | Return from interrupt |

Jcond = Jump on Condition

Operation

if condition is true then do
 sign-extend displacement to 16 bits.
 IP ← IP + sign-extended displacement
 end if.

Format



| Opcode | Clocks | Operation | cond |
|--------|---------|--|------|
| 77 | 16 or 4 | jump if above | JAE |
| 73 | 16 or 4 | jump if above or equal | JBE |
| 72 | 16 or 4 | jump if below | JBE |
| 76 | 16 or 4 | jump if below or equal | JBE |
| 72 | 16 or 4 | jump if carry set | JBC |
| 74 | 16 or 4 | jump if equal | JE |
| 7F | 16 or 4 | jump if greater | JG |
| 7D | 16 or 4 | jump if greater or equal | JGE |
| 7C | 16 or 4 | jump if less | JL |
| 76 | 16 or 4 | jump if less or equal | JLE |
| 7E | 16 or 4 | jump if not above | JNA |
| 72 | 16 or 4 | jump if neither above nor equal | JNAE |
| 73 | 16 or 4 | jump if not below | JNB |
| 77 | 16 or 4 | jump if neither below nor equal | JNBE |
| 73 | 16 or 4 | jump if no carry | JNC |
| 75 | 16 or 4 | jump if not equal | JNE |
| 7E | 16 or 4 | jump if not greater | JNG |
| 7C | 16 or 4 | jump if neither greater nor equal | JNGE |
| 7D | 16 or 4 | jump if not less | JNL |
| 7F | 16 or 4 | jump if neither less nor equal | JNLE |
| 71 | 16 or 4 | jump if no overflow | JNO |
| 7B | 16 or 4 | jump if no parity | JNP |
| 79 | 16 or 4 | jump if positive | JNS |
| 75 | 16 or 4 | jump if not zero | JNZ |
| 70 | 16 or 4 | jump if overflow | JO |
| 7A | 16 or 4 | jump if parity | JP |
| 7A | 16 or 4 | jump if parity even | JPE |
| 7B | 16 or 4 | jump if parity odd | JPO |
| 78 | 16 or 4 | jump if sign | JS |
| 74 | 18 or 6 | jump if zero | JZ |
| E3 | 18 or 6 | jump if CX is zero (does not test flags) | JCXZ |

JMP = Jump

Within segment or group, IP relative

| Opcode | DispL | DispH |
|--------|-------|-------|
| | | |

| Opcode | Clocks | Operation |
|--------|--------|---------------------------------------|
| E9 | 15 | IP = IP + Disp16 |
| EB | 15 | IP = IP + Disp8 (Disp8 sign-extended) |

Within segment or group, Indirect

| Opcode | mod 100 r/m |
|--------|-------------|
| | |

| Opcode | Clocks | Operation |
|--------|---------|------------|
| FF | 11 | IP = Reg16 |
| FF | 18 - EA | IP = Mem16 |

Inter-segment or group, Direct

| Opcode | offset | offset | segbase | segbase |
|--------|--------|--------|---------|---------|
| | | | | |

| Opcode | Clocks | Operation |
|--------|--------|-----------------------------|
| EA | 15 | CS = segbase IP = offset |

Inter-segment or group, Indirect

| Opcode | mod 101 r/m |
|--------|-------------|
| | |

| Opcode | Clocks | Operation |
|--------|---------|-----------------------------|
| FF | 24 - EA | CS = segbase IP = offset |

LAHF = Load AH from Flags

| Opcode |
|--------|
| |

| Opcode | Clocks | Operation |
|--------|--------|-----------------------------------|
| 9F | 4 | copy low byte of flags word to AH |

LDS/LES = Load Pointer to DS/ES and Register

| Opcode | mod reg r/m |
|--------|-------------|
| | |

| Opcode | Clocks | Operation |
|--------|---------|--|
| C4 | 16 - EA | dword pointer at EA goes to reg16 (1st word) and ES (2nd word) |
| C5 | 16 - EA | dword pointer at EA goes to reg16 (1st word) and DS (2nd word) |

LEA = Load Effective Address

| Opcode | mod reg r/m |
|--------|-------------|
| | |

| Opcode | Clocks | Operation |
|--------|--------|------------|
| BD | 2 - EA | Reg16 = EA |

LOCK = Assert Bus Lock

| Opcode |
|--------|
| |

| Opcode | Clocks | Operation |
|--------|--------|--------------------------------------|
| F0 | 2 | assert the bus lock next instruction |

LOOPxx = Loop Control

| Opcode | Disp |
|--------|------|
| | |

| Opcode | Clocks | Operation | xx = |
|--------|---------|--|--------|
| E1 | 18 or 6 | dec CX, loop if equal and CX not 0 | LOOPE |
| E0 | 19 or 5 | dec CX, loop if not equal and CX not 0 | LOOPNE |
| E1 | 18 or 6 | dec CX, loop if zero and CX not 0 | LOOPZ |
| E0 | 19 or 5 | dec CX, loop if not zero and CX not 0 | LOOPNZ |
| E2 | 17 or 5 | dec CX, loop if CX not 0 | LOOP |

MOV = Move Data

Memory/Reg to or from Reg

| Opcode | mod reg r/m |
|--------|-------------|
| | |

| Byte | Opcode | Clocks | Operation |
|------|--------|--------|---------------|
| | 8B | 9 + EA | Mem8 = Reg8 |
| | 8B | 2 | Reg8 = Reg8 |
| | 8A | 8 + EA | Reg8 = Mem8 |
| Word | Opcode | Clocks | Operation |
| | 8B | 9 + EA | Mem16 = Reg16 |
| | 8B | 2 | Reg16 = Reg16 |
| | 8B | 8 + EA | Reg16 = Mem16 |

Direct-Addressed Memory to or from AX/AL

| Opcode | AddrL | AddrH |
|--------|-------|-------|
| | | |

| Byte | Opcode | Clocks | Operation |
|------|--------|--------|------------|
| | A0 | 10 | AL = Mem8 |
| | A2 | 10 | Mem8 = AL |
| Word | Opcode | Clocks | Operation |
| | A1 | 10 | AX = Mem16 |
| | A3 | 10 | Mem16 = AX |

Immed to Reg

| Opcode | Data |
|--------|------|
| | |

| Byte | Opcode | Clocks | Operation |
|------|----------|--------|-----------------|
| | B0 + reg | 4 | Reg8 = immed8 |
| Word | Opcode | Clocks | Operation |
| | B8 + reg | 4 | Reg16 = immed16 |

Immed to Memory/Reg

| Opcode | mod 000 r/m | Data |
|--------|-------------|------|
| | | |

| Opcode | Clocks | Operation |
|--------|---------|-----------------|
| C6 | 4 | Reg8 = immed8 |
| C6 | 10 - EA | Mem8 = immed8 |
| C7 | 4 | Reg16 = immed16 |
| C7 | 10 - EA | Mem16 = immed16 |

Memory/Reg to or from SReg

| Opcode | mod s/reg r/m |
|--------|---------------|
| | |

| Word | Opcode | Clocks | Operation |
|------|--------|--------|--------------|
| | BC | 9 + EA | Mem16 = SReg |
| | BC | 2 | Reg16 = SReg |
| | BE | 8 + EA | SReg = Mem16 |
| | BE | 2 | SReg = Reg16 |

MUL = Unsigned Multiplication

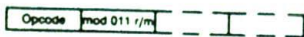
Memory/Reg with AL or AX

| Opcode | mod 100 r/m |
|--------|-------------|
| | |

| Byte | Opcode | Clocks | Operation |
|------|--------|----------------|--------------------|
| | F6 | 70-77 | AX = AL * Reg8 |
| | F6 | (76-83) + EA | AX = AL * Mem8 |
| Word | Opcode | Clocks | Operation |
| | F7 | 118-133 | DX AX = AX * Reg16 |
| | F7 | (124-139) - EA | DX AX = AX * Mem16 |

NEG = Negate an Integer

Memory/Reg



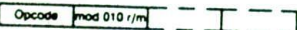
| Opcode | Clocks | Operation |
|--------|---------|-----------------------|
| F6 | 3 | Reg8 ← 00H · Reg8 |
| F7 | 3 | Reg16 ← 0000H · Reg16 |
| F8 | 13 - EA | Mem8 ← 00H · Mem8 |
| F9 | 16 - EA | Mem16 ← 0000H · Mem16 |

NOP = No Operation



| Opcode | Clocks | Operation |
|--------|--------|--------------|
| 90 | 3 | no operation |

NOT = Form One's Complement
Memory/Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|------------------------|
| F6 | F6 | 3 | Reg8 ← 0FFH · Reg8 |
| Word | F7 | 3 | Reg16 ← 0FFFFH · Reg16 |
| | F7 | 16 + EA | Mem16 ← 0FFFFH · Mem16 |

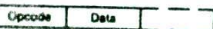
OR = Logical Inclusive OR

Memory/Reg with Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|------------------------|
| 0A | 0A | 3 | Reg8 ← Reg8 OR Reg8 |
| Word | 0B | 3 | Reg16 ← Reg16 OR Reg16 |
| | 08 | 9 - EA | Reg8 ← Reg8 OR Mem8 |
| | 09 | 16 + EA | Reg16 ← Reg16 OR Mem16 |
| | 0E | 16 + EA | Mem8 ← Mem8 OR Reg8 |
| | 0F | 16 + EA | Mem16 ← Mem16 OR Reg16 |

Immed to AX/AL



| Opcode | Clocks | Operation |
|--------|--------|--------------------|
| 0C | 4 | AL ← AL OR Immed8 |
| 0D | 4 | AX ← AX OR Immed16 |

Immed to Memory/Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|--------------------------|
| 80 | 80 | 4 | Reg8 ← Reg8 OR Immed8 |
| Word | 81 | 4 | Reg16 ← Reg16 OR Immed16 |
| | 80 | 17 + EA | Mem8 ← Mem8 OR Immed8 |
| | 81 | 17 + EA | Mem16 ← Mem16 OR Immed16 |

OUT = Output Byte, Word

Fixed port



| Byte | Opcode | Clocks | Operation |
|------|--------|--------|------------|
| E6 | E6 | 10 | Port8 ← AL |
| E7 | E7 | 10 | Port8 ← AX |

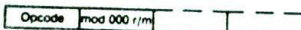
Variable port



| Word | Opcode | Clocks | Operation |
|------|--------|--------|---------------------|
| EE | EE | 8 | Port16 (in DX) ← AL |
| EF | EF | 8 | Port16 (in DX) ← AX |

POP = Pop a Word from the Stack

Word Memory



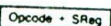
| Opcode | Clocks | Operation |
|--------|---------|------------------|
| 8F | 17 - EA | Mem16 ← (SP) + 2 |

Word Register



| Opcode | Clocks | Operation |
|----------|--------|------------------|
| 58 + reg | 8 | Reg16 ← (SP) + 2 |

Segment Register



| Opcode | Clocks | Operation |
|-----------|--------|-----------------|
| 07 + SReg | 8 | SReg ← (SP) + 2 |

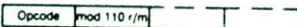
POPF = Pop the TOS into the Flags



| Opcode | Clocks | Operation |
|--------|--------|------------------|
| 9D | 8 | FLAGS ← (SP) + 2 |

PUSH = Push a Word onto the Stack

Memory/Reg



| Opcode | Clocks | Operation |
|--------|---------|---------------|
| FF | 16 - EA | -(SP) ← Mem16 |

Word Register



| Opcode | Clocks | Operation |
|----------|--------|---------------|
| 50 + reg | 11 | -(SP) ← Reg16 |

Segment Register



| Opcode | Clocks | Operation |
|-----------|--------|--------------|
| 06 + SReg | 10 | -(SP) ← SReg |

PUSHF = Push the Flags to the Stack



| Opcode | Clocks | Operation |
|--------|--------|---------------|
| 9C | 10 | -(SP) ← FLAGS |

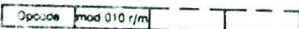
RCL = Rotate Left Through Carry

Memory or Reg by 1



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|-------------------|
| D0 | D0 | 2 | rotate Reg8 by 1 |
| Word | D1 | 2 | rotate Reg16 by 1 |
| | D0 | 15 + EA | rotate Mem8 by 1 |
| | D1 | 15 + EA | rotate Mem16 by 1 |

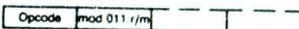
Memory or Reg by count in CL



| Byte | Opcode | Clocks | Operation |
|------|--------|-----------------|--------------------|
| D2 | D2 | 8 + 4/bit | rotate Reg8 by CL |
| Word | D3 | 8 + 4/bit | rotate Reg16 by CL |
| | D2 | 20 + EA - 4/bit | rotate Mem8 by CL |
| | D3 | 20 + EA + 4/bit | rotate Mem16 by CL |

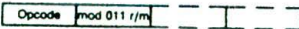
RCR = Rotate Right Through Carry

Memory or Reg by 1



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|-------------------|
| D0 | D0 | 2 | rotate Reg8 by 1 |
| Word | D1 | 2 | rotate Reg16 by 1 |
| | D0 | 15 + EA | rotate Mem8 by 1 |
| | D1 | 15 + EA | rotate Mem16 by 1 |

Memory or Reg by count in CL



| Byte | Opcode | Clocks | Operation |
|------|--------|-----------------|--------------------|
| D2 | D2 | 8 + 4/bit | rotate Reg8 by CL |
| Word | D3 | 8 + 4/bit | rotate Reg16 by CL |
| | D2 | 20 + EA + 4/bit | rotate Mem8 by CL |
| | D3 | 20 + EA + 4/bit | rotate Mem16 by CL |

REP_x = Repeat Prefix



| Opcode | Clocks | Operation | REP _x |
|--------|--------|--|------------------|
| F3 | 2 | repeat next instruction until CX = 0 | REP |
| F3 | 2 | repeat next instruction until CX = 0 or ZF = 0 | REPE REPZ |
| F2 | 2 | repeat next instruction until CX = 0 or ZF = 1 | REPNE REPNZ |

RET = Return from Subroutine

| Opcode | Clocks | Operation |
|--------|--------|----------------------|
| C3 | 8 | intra-segment return |
| CB | 18 | inter-segment return |

Return and add constant to SP

| Opcode | Data1 | Data2 | Operation |
|--------|-------|-------|---------------------------|
| C2 | | | intra-segment ret and add |
| CA | | | inter-segment ret and add |

ROL = Rotate Left

Memory or Reg by 1

| Opcode | mod | 000 r/m | | | |
|--------|-----|---------|-------------------|--|--|
| Byte | D0 | 2 | rotate Reg8 by 1 | | |
| Word | D0 | 15 + EA | rotate Mem8 by 1 | | |
| | D1 | 2 | rotate Reg16 by 1 | | |
| | D1 | 15 + EA | rotate Mem16 by 1 | | |

Memory or Reg by count in CL

| Opcode | mod | 000 r/m | | | |
|--------|-----|----------------|--------------------|--|--|
| Byte | D2 | 8 + 4/bt | rotate Reg8 by CL | | |
| | D2 | 20 + Ea + 4/bt | rotate Mem8 by CL | | |
| Word | D3 | 8 + 4/bt | rotate Reg16 by CL | | |
| | D3 | 20 + Ea + 4/bt | rotate Mem16 by CL | | |

ROR = Rotate Right

Memory or Reg by 1

| Opcode | mod | 001 r/m | | | |
|--------|-----|---------|-------------------|--|--|
| Byte | D0 | 2 | rotate Reg8 by 1 | | |
| | D0 | 15 + EA | rotate Mem8 by 1 | | |
| Word | D1 | 2 | rotate Reg16 by 1 | | |
| | D1 | 15 + EA | rotate Mem16 by 1 | | |

Memory or Reg by count in CL

| Opcode | mod | 001 r/m | | | |
|--------|-----|----------------|--------------------|--|--|
| Byte | D2 | 8 + 4/bt | rotate Reg8 by CL | | |
| | D2 | 20 + Ea + 4/bt | rotate Mem8 by CL | | |
| | D3 | 8 + 4/bt | rotate Reg16 by CL | | |
| | D3 | 20 + Ea + 4/bt | rotate Mem16 by CL | | |

SAHF = Store AH in Flags

| Opcode | Clocks | Operation |
|--------|--------|-----------------------------------|
| 9E | 4 | copy AH to low byte of flags word |

SAL/SHL = Arithmetic/Logical Left Shift

Memory or Reg by 1

| Opcode | mod | 100 r/m | | | |
|--------|-----|---------|------------------|--|--|
| Byte | D0 | 2 | shift Reg8 by 1 | | |
| | D0 | 15 + EA | shift Mem8 by 1 | | |
| Word | D1 | 2 | shift Reg16 by 1 | | |
| | D1 | 15 + EA | shift Mem16 by 1 | | |

Memory or Reg by count in CL

| Opcode | mod | 100 r/m | | | |
|--------|-----|----------------|-------------------|--|--|
| Byte | D2 | 8 + 4/bt | shift Reg8 by CL | | |
| | D2 | 20 + Ea + 4/bt | shift Mem8 by CL | | |
| Word | D3 | 8 + 4/bt | shift Reg16 by CL | | |
| | D3 | 20 + Ea + 4/bt | shift Mem16 by CL | | |

SAR = Arithmetic Right Shift

Memory or Reg by 1

| Opcode | mod | 111 r/m | | | |
|--------|-----|---------|-----------------|--|--|
| Byte | D0 | 2 | shift Reg8 by 1 | | |
| | D0 | 15 + EA | shift Mem8 by 1 | | |

| Word | D1 | 2 | shift Reg16 by 1 |
|------|----|---------|------------------|
| | D1 | 15 + EA | shift Mem16 by 1 |

Memory or Reg by count in CL

| Opcode | mod | 111 r/m | | | |
|--------|-----|----------------|-------------------|--|--|
| Byte | D2 | 8 + 4/bt | shift Reg8 by CL | | |
| | D2 | 20 + Ea + 4/bt | shift Mem8 by CL | | |
| Word | D3 | 8 + 4/bt | shift Reg16 by CL | | |
| | D3 | 20 + Ea + 4/bt | shift Mem16 by CL | | |

SBB = Integer Subtraction with Borrow

Memory/Reg with Reg

| Opcode | mod | reg r/m | | | |
|--------|-----|---------|----------------------------|--|--|
| Byte | 1A | 3 | Reg8 = Reg8 - Reg8 - CF | | |
| | 1A | 9 - EA | Reg8 = Reg8 - Mem8 - CF | | |
| | 1B | 16 + EA | Mem8 = Mem8 - Reg8 - CF | | |
| Word | 1B | 3 | Reg16 = Reg16 - Reg16 - CF | | |
| | 1B | 9 - EA | Reg16 = Reg16 - Mem16 - CF | | |
| | 19 | 16 + EA | Mem16 = Mem16 - Reg16 - CF | | |

Immed from AX/AL

| Opcode | Data | | | | |
|--------|------|---|------------------------|--|--|
| 1C | | 4 | AL = AL - Immed8 - CF | | |
| 1D | | 4 | AX = AX - Immed16 - CF | | |

Immed from Memory/Reg

| Opcode | mod | 011 r/m | | | Data |
|--------|-----|---------|------------------------------|--|------|
| 80 | | 4 | Reg8 = Reg8 - Immed8 - CF | | |
| 80 | | 17 + EA | Mem8 = Mem8 - Immed8 - CF | | |
| 81 | | 4 | Reg16 = Reg16 - Immed16 - CF | | |
| 81 | | 17 + EA | Mem16 = Mem16 - Immed16 - CF | | |
| 83 | | 4 | Reg16 = Reg16 - Immed8 - CF | | |
| 83 | | 17 + EA | Mem16 = Mem16 - Immed8 - CF | | |

(Immed8 is sign-extended before subtract)

SHR = Logical Right Shift

Memory or Reg by 1

| Opcode | mod | 101 r/m | | | |
|--------|-----|---------|------------------|--|--|
| Byte | D0 | 2 | shift Reg8 by 1 | | |
| | D0 | 15 + EA | shift Mem8 by 1 | | |
| Word | D1 | 2 | shift Reg16 by 1 | | |
| | D1 | 15 + EA | shift Mem16 by 1 | | |

Memory or Reg by count in CL

| Opcode | mod | 101 r/m | | | |
|--------|-----|----------------|-------------------|--|--|
| Byte | D2 | 8 + 4/bt | shift Reg8 by CL | | |
| | D2 | 20 + Ea + 4/bt | shift Mem8 by CL | | |
| Word | D3 | 8 + 4/bt | shift Reg16 by CL | | |
| | D3 | 20 + Ea + 4/bt | shift Mem16 by CL | | |

STC = Set Carry Flag

| Opcode | Clocks | Operation |
|--------|--------|--------------------|
| F9 | 2 | set the carry flag |

STD = Set Direction Flags

| Opcode | Clocks | Operation |
|--------|--------|--------------------|
| FD | 2 | set direction flag |

STI = Set Interrupt Enable Flag

| Opcode | Clocks | Operation |
|--------|--------|--------------------|
| FB | 2 | set interrupt flag |

String = String Operations

| Opcode | Clocks | Operation | String |
|--------|--------|---------------------|--------|
| A6 | 22 | flags = (SI) - (DI) | CMPS |

| | | | |
|----|----|---------------------|------|
| A7 | 22 | flags ← (SI) - (DI) | CMPS |
| A4 | 18 | (DI) ← (SI) | MOVS |
| A5 | 18 | (DI) ← (SI) | MOVS |
| AE | 15 | flags ← (DI) - AL | SCAS |
| AF | 15 | flags ← (DI) - AX | SCAS |
| AC | 12 | AL ← (SI) | LODS |
| AD | 12 | AX ← (SI) | LODS |
| AA | 11 | (DI) ← AL | STOS |
| AB | 11 | (DI) ← AX | STOS |

SUB = Integer Subtraction

Memory/Reg with Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|-----------------------|
| | 2A | 3 | Reg8 ← Reg8 - Reg8 |
| | 2A | 9 - EA | Reg8 ← Reg8 - Mem8 |
| | 2B | 16 - EA | Mem8 ← Mem8 - Reg8 |
| Word | Opcode | Clocks | Operation |
| | 2B | 3 | Reg16 ← Reg16 - Reg16 |
| | 2B | 9 - EA | Reg16 ← Reg16 - Mem16 |
| | 29 | 16 - EA | Mem16 ← Mem16 - Reg16 |

Immed to AX/AL



| Byte | Opcode | Clocks | Operation |
|------|--------|--------|-------------------|
| | 2C | 4 | AL ← AL - Immed8 |
| Word | 2D | 4 | AX ← AX - Immed16 |

Immed to Memory/Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|-------------------------|
| | 80 | 4 | Reg8 ← Reg8 - Immed8 |
| | 80 | 17 - EA | Mem8 ← Mem8 - Immed8 |
| Word | Opcode | Clocks | Operation |
| | 81 | 4 | Reg16 ← Reg16 - Immed16 |
| | 81 | 17 - EA | Mem16 ← Mem16 - Immed16 |
| | 83 | 4 | Reg16 ← Reg16 - Immed8 |
| | 83 | 17 - EA | Mem16 ← Mem16 - Immed8 |

TEST = Logical Compare

Memory/Reg with Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|--------|-------------------------|
| | 84 | 3 | flags ← Reg8 AND Reg8 |
| | 84 | 9 - EA | flags ← Reg8 AND Mem8 |
| Word | 85 | 3 | flags ← Reg16 AND Reg16 |
| | 85 | 9 - EA | flags ← Reg16 AND Mem16 |

Immed to AX/AL



| Byte | Opcode | Clocks | Operation |
|------|--------|--------|------------------------|
| | A8 | 4 | flags ← AL AND Immed8 |
| Word | A9 | 4 | flags ← AX AND Immed16 |

Immed to Memory/Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|---------------------------|
| | F6 | 5 | flags ← Reg8 AND Immed8 |
| | F6 | 11 - EA | flags ← Mem8 AND Immed8 |
| Word | F7 | 5 | flags ← Reg16 AND Immed16 |
| | F7 | 11 - EA | flags ← Mem16 AND Immed16 |

WAIT = Wait While TEST Pin Not Asserted



| Opcode | Clocks | Operation |
|--------|--------|-----------|
| 9B | 3 + 5n | none |

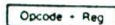
XCHG = Exchange Memory/Register with Register

Memory/Reg with Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|---------------|
| | 86 | 4 | Reg8 ↔ Reg8 |
| | 86 | 17 - EA | Mem8 ↔ Reg8 |
| Word | 87 | 4 | Reg16 ↔ Reg16 |
| | 87 | 17 - EA | Mem16 ↔ Reg16 |

Word Register with AX



| Opcode - Reg | Clocks | Operation |
|--------------|--------|------------|
| 90 - Reg | 3 | AX ↔ Reg16 |

XLAT = Table Look-up Translation



| Opcode | Clocks | Operation |
|--------|--------|-----------------------------|
| D7 | 11 | replace AL with table entry |

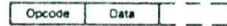
XOR = Logical Exclusive OR

Memory/Reg with Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|-------------------------|
| | 32 | 3 | Reg8 ← Reg8 XOR Reg8 |
| | 32 | 9 - EA | Reg8 ← Reg8 XOR Mem8 |
| | 30 | 16 - EA | Mem8 ← Mem8 XOR Reg8 |
| Word | Opcode | Clocks | Operation |
| | 33 | 3 | Reg16 ← Reg16 XOR Reg16 |
| | 33 | 9 - EA | Reg16 ← Reg16 XOR Mem16 |
| | 31 | 16 - EA | Mem16 ← Mem16 XOR Reg16 |

Immed to AX/AL



| Opcode | Clocks | Operation |
|--------|--------|---------------------|
| 34 | 4 | AL ← AL XOR Immed8 |
| 35 | 4 | AX ← AX XOR Immed16 |

Immed to Memory/Reg



| Byte | Opcode | Clocks | Operation |
|------|--------|---------|---------------------------|
| | 80 | 4 | Reg8 ← Reg8 XOR Immed8 |
| | 80 | 17 - EA | Mem8 ← Mem8 XOR Immed8 |
| Word | Opcode | Clocks | Operation |
| | 81 | 4 | Reg16 ← Reg16 XOR Immed16 |
| | 81 | 17 - EA | Mem16 ← Mem16 XOR Immed16 |

186 INSTRUCTIONS

Notes for iAPX 186 Instructions

These instructions can be used only if the MOD186 control is specified. When MOD186 is specified, clocks for all instructions are as stated under "Clocks for MOD186 Operation".

BOUND = Check Array Against Bounds



| Opcode | Operation |
|--------|---|
| 82 | if Reg16 < Mem16 at EA, or Reg16 > Mem16 at EA + 2 then INTERRUPT 5 |

ENTER = High Level Procedure Entry



| Opcode | Operation |
|--------|-----------------------|
| C8 | build new stack frame |

IMUL = Signed Multiplication

Mem/Reg* Immediate to Reg



| Opcode | Operation |
|--------|-------------------------|
| 68 | Reg16 ← Reg16 * Immed8 |
| 6E | Reg16 ← Reg16 * Immed8 |
| 6B | Reg16 ← Mem16 * Immed8 |
| 69 | Reg16 ← Reg16 * Immed16 |
| 69 | Reg16 ← Reg16 * Immed16 |
| 69 | Reg16 ← Mem16 * Immed16 |

LEAVE = High Level Procedure Exit



| Opcode | Operation |
|--------|--|
| C9 | release current stack frame and return to prior frame |

POP A = Pop All Registers



| Opcode | Operation |
|--------|---------------------------------|
| 61 | restore registers from stack |

PUSH = Push a Word onto the Stack

Word Immediate

| | | | | |
|--------|------|--|--|--|
| Opcode | Data | | | |
|--------|------|--|--|--|

Opcode Operation
 6A $-(SP) = \text{immed8}$
 (sign extended)
 68 $-(SP) = \text{immed16}$

PUSHA = Push All Registers

| |
|--------|
| Opcode |
|--------|

Opcode Operation
 60 save registers on the stack

RCL = Rotate Left Through Carry

Mem or Reg by Immed8

| | | | | | |
|--------|--------|--|--|--|-------|
| Opcode | ModRM* | | | | count |
|--------|--------|--|--|--|-------|

*—(Reg field = 011)

Opcode Operation
 C0 rotate Reg8 by Immed8
 C0 rotate Mem8 by Immed8
 C1 rotate Reg16 by Immed8
 C1 rotate Mem16 by Immed8

RCR = Rotate Right Through Carry

Mem or Reg by Immed8

| | | | | | |
|--------|--------|--|--|--|-------|
| Opcode | ModRM* | | | | count |
|--------|--------|--|--|--|-------|

*—(Reg field = 011)

Opcode Operation
 C0 rotate Reg8 by Immed8
 C0 rotate Mem8 by Immed8
 C1 rotate Reg16 by Immed8
 C1 rotate Mem16 by Immed8

ROL = Rotate Left

Mem or Reg by Immed8

| | | | | | |
|--------|--------|--|--|--|-------|
| Opcode | ModRM* | | | | count |
|--------|--------|--|--|--|-------|

*—(Reg field = 000)

Opcode Operation
 C0 rotate Reg8 by Immed8
 C0 rotate Mem8 by Immed8
 C1 rotate Reg16 by Immed8
 C1 rotate Mem16 by Immed8

ROR = Rotate Right

Mem or Reg by Immed8

| | | | | | |
|--------|--------|--|--|--|-------|
| Opcode | ModRM* | | | | count |
|--------|--------|--|--|--|-------|

*—(Reg field = 001)

Opcode Operation
 C0 rotate Reg8 by Immed8
 C0 rotate Mem8 by Immed8
 C1 rotate Reg16 by Immed8
 C1 rotate Mem16 by Immed8

SAL/SHL = Arithmetic/Logical Left Shift

Mem or Reg by immediate count

| | | | | | |
|--------|--------|--|--|--|-------|
| Opcode | ModRM* | | | | count |
|--------|--------|--|--|--|-------|

*—(Reg field = 100)

Opcode Operation
 C0 rotate Reg8 by Immed8
 C0 rotate Mem8 by Immed8
 C1 rotate Reg16 by Immed8
 C1 rotate Mem16 by Immed8

SAR = Arithmetic Right Shift

Mem or Reg by Immed8

| | | | | | |
|--------|--------|--|--|--|-------|
| Opcode | ModRM* | | | | count |
|--------|--------|--|--|--|-------|

*—(Reg field = 111)

Opcode Operation
 C0 rotate Reg8 by Immed8
 C0 rotate Mem8 by Immed8
 C1 rotate Reg16 by Immed8
 C1 rotate Mem16 by Immed8

SHR = Logical Right Shift

Mem or Reg by Immed8

| | | | | | |
|--------|--------|--|--|--|-------|
| Opcode | ModRM* | | | | count |
|--------|--------|--|--|--|-------|

*—(Reg field = 101)

Opcode Operation
 C0 rotate Reg8 by Immed8
 C0 rotate Mem8 by Immed8
 C1 rotate Reg16 by Immed8
 C1 rotate Mem16 by Immed8

String = String Operations (INS/OUTS)

| |
|--------|
| Opcode |
|--------|

| Opcode | Clocks | Operation |
|--------|--------|------------------------------------|
| 6E | INS | (DI) \rightarrow port(DX) |
| 6F | INS | (DI) \rightarrow port(DX DX + 1) |
| 6C | OUTS | port(DX) \rightarrow (SI) |
| 6D | OUTS | port(DX DX + 1) \rightarrow (SI) |

8087 INSTRUCTIONS

Notes for 8087 Instructions

The individual instruction descriptions are shown by a format box such as the following:

| | | | | |
|------|-----|----------|-------|-------|
| WAIT | op1 | m/op/r/m | addr1 | addr2 |
|------|-----|----------|-------|-------|

These are the byte-wise representations of the object code generated by the assembler and are interpreted as follows:

- WAIT is an 8086 wait instruction, NOP or emulator instruction.
- op1 is the opcode, possibly taking two bytes.
- m/op/r/m byte (middle 3-bits is part of the opcode).
- addr1 and addr2 are offsets of either 8 or 16 bits.

For integer functions, m = 0 for short-integer memory operand; 1 for word-integer memory operand.
 For real functions, m = 0 for short-real memory operand; 1 for long-real memory operand
 i = stack element index

If mod = 00 then DISP = 0, disp-lo and disp-hi are absent.
 If mod = 01 then DISP = disp-lo sign-extended to 16 bits, disp-hi is absent.

If mod = 10 then DISP = disp-hi, disp-lo is absent.

If mod = 11 then r/m is treated as an ST(i) field

If r/m = 000 then EA = (BX) + (SI) + DISP

If r/m = 001 then EA = (BX) + (DI) + DISP

If r/m = 010 then EA = (BP) + (SI) + DISP

If r/m = 011 then EA = (BP) - (DI) + DISP

If r/m = 100 then EA = (SI) + DISP

If r/m = 101 then EA = (DI) + DISP

If r/m = 110 then EA = (BP) + DISP

If r/m = 111 then EA = (BX) + DISP

*Except if mod = 000 and r/m = 110 then EA = disp-hi, disp-lo.

ST(0) = Current stack top

ST(i) = register below stack top

d = Destination

0 = Destination is ST(0)

1 = Destination is ST(i)

P = Pop

0 = No pop

1 = Pop ST(0)

R = Reverse

0 = Destination (op) source

1 = Source (op) destination

For FSQRT $-0 \leq ST(0) < +\infty$

For FSCALE $-2^i \leq ST(1) < -2^{i-1}$ and ST(1) integer

For F2XM1 $0 \leq ST(0) < 2$

For FYL2X $0 \leq ST(0) < \infty$

For FYL2XP1 $-\infty < ST(1) < +\infty$

$0 \leq |ST(0)| < (2 - \sqrt{2})/2$

$-\infty < ST(1) < \infty$

For FPATAN $0 \leq ST(0) < \pi/4$

For FPATAN $0 \leq ST(0) < ST(1) < +\infty$

F2XMI = Compute $2^i - 1$

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
|---------------|-------------------|---------------|------------------------|
| 9B D9 F0 | CD 19 F0 | 500 | ST = 2 ^{ST-1} |
| | | 310-630 | |

FABS = Absolute Value

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
|---------------|-------------------|---------------|-----------|
| 9B D9 E1 | CD 19 E1 | 14 | ST = ST |
| | | 10-17 | |

FADD = Add Real

Stack top + Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + 1 |
|------|-----|---------|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|--------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D8 C0+ | CD 18 C0+ | 85 70-100 | ST ← ST + ST(i) | |
| 9B DC C0+ | CD 1C C0+ | 85 70-100 | ST(i) ← ST + ST(i) | |

Stack top + memory operand

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 000 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| | | Execution Clocks | | |
|---------------|-------------------|---------------------------|----------------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D8 m0m | CD 18 m0m | 105 - EA (90-120) - EA | ST ← ST + mem-op (short-real) | |
| 9B DC m0m | CD 1C m0m | 110 - EA (95-125) - EA | ST ← ST + mem-op (long-real) | |

FADDP = Add Real and Pop

Stack top + Stack Element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|---------------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B DE C1 | CD 1E C1 | 90 75-105 | ST(i) ← ST + ST(i) pop stack | |
| 9B DE C0+ | CD 1E C0+ | 90 75-105 | ST(i) ← ST + ST(i) pop stack | |

FBLD = Packed Decimal (BCD) Load

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 100 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| | | Execution Clocks | | |
|---------------|-------------------|----------------------------|---------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B DF m4m | CD 1F m4m | 300 - EA (290-310) - EA | push stack ST ← mem-op | |

FBSTP = Packed Decimal (BCD) Store and Pop

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 110 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| | | Execution Clocks | | |
|---------------|-------------------|----------------------------|--------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B DF m8m | CD 1F m8m | 530 - EA (520-540) - EA | mem-op ← ST pop stack | |

FCNS = Change Sign

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|-----------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D9 E0 | CD 19 E0 | 15 10-17 | ST ← -ST | |

**FCLEX
FNCLEX = Clear Exceptions**

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|------------------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B DB E2 | CD 1B E2 | 5 2-8 | clear 8087 exceptions | |
| 90 DB E2 | CD 1B E2 | 5 2-8 | clear 8087 exceptions (no wait) | |

FCOM = Compare Real

Compare Stack top and Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D8 D1 | CD 18 D1 | 45 40-50 | ST ← ST(i) | |

9B D8 D0+; CD 18 D0+; 45
40-50 ST ← ST(i)

Compare Stack top and memory operands

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 010 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| | | Execution Clocks | | |
|---------------|-------------------|-------------------------|----------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D8 m2m | CD 18 m2m | 65 - EA (60-70) - EA | ST ← memop (short-real) | |
| 9B DC m2m | CD 1C m2m | 70 - EA (65-75) - EA | ST ← memop (long-real) | |

FCOMP = Compare Real and Pop

Compare Stack top and Stack element and pop

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|-------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D8 D9 | CD 18 D9 | 47 42-52 | ST ← ST(i) pop stack | |
| 9B D8 D8+ | CD 18 D8+ | 47 42-52 | ST ← ST(i) pop stack | |

Compare Stack top and memory operand and pop

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 011 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| | | Execution Clocks | | |
|---------------|-------------------|-------------------------|--|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D8 m3m | CD 18 m3m | 68 - EA (63-73) - EA | ST ← mem-op pop stack (short-real) | |
| 9B DC m3m | CD 1C m3m | 72 - EA (67-77) - EA | ST ← mem-op pop stack (long-real) | |

FCOMPP = Compare Real and Pop Twice

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|--------------------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B DE D9 | CD 1E D9 | 50 45-55 | ST ← ST(i) pop stack pop stack | |

FDECSTP = Decrement Stack Pointer

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|--------------------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D9 F6 | CD 19 F6 | 9 6-12 | stack pointer ← stack pointer - 1 | |

**FDISI
FNDISI = Disable Interrupts**

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|--------------------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B DB E1 | CD 1B E1 | 5 2-8 | Set 8087 interrupt mask | |
| 90 DB E1 | CD 1B E1 | 5 2-8 | Set 8087 interrupt mask (no wait) | |

FDIV = Divide Real

Stack top and Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| | | Execution Clocks | | |
|---------------|-------------------|------------------|---------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation | |
| 9B D8 F0+ | CD 18 F0+ | 198 193-203 | ST ← ST:ST(i) | |

9B DC F8+; CD 1C F8+; 196 ST(i) - ST(i)/ST
193-203

Stack top and memory operand

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 110 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------------|--------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B D8 m8rm | CD 18 m8rm | 220 - EA (215-225) - EA | ST ← ST/mem-op (short-real) |
| 9B DC m8rm | CD 1C m8rm | 225 - EA (220-230) - EA | ST ← ST/mem-op (long-real) |

FDIVP = Divide Real and Pop

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|-------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DE F9 | CD 1E F9 | 202 | ST(1) ← ST(1)/ST pop stack |
| 9B DE F8+; | CD 1E F8+; | 202 | ST(i) ← ST(i)/ST pop stack |

FDIVR = Divide Real Reversed

Stack top and Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B D8 F8+; | CD 18 F8+; | 199 | ST ← ST(i)/ST |
| 9B DC F0+; | CD 1C F0+; | 199 | ST(i) ← ST/ST(i) |

Stack top and memory operand

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 111 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------------|--------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B D8 m7rm | CD 18 m7rm | 221 + EA (216-226) + EA | ST ← mem-op/ST (short-real) |
| 9B DC m7rm | CD 1C m7rm | 226 + EA (221-231) + EA | ST ← mem-op/ST (long-real) |

FDIVRP = Divide Real Reversed and Pop

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|-------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DE F1 | CD 1E F1 | 203 | ST(1) ← ST/ST(1) pop stack |
| 9B DE F0+; | CD 1E F0+; | 203 | ST(i) ← ST/ST(i) |

FENI FNENI = Enable Interrupts

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|-------------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DB E0 | CD 1B E0 | 5 | clear 8087 interrupt mask |
| 9B DB E0 | CD 1B E0 | 2-8 | clear 8087 interrupt mask (no wait) |

FFREE = Free Register

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|---------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DD C0+; | CD 1D C0+; | 11 | TAG(i) masked empty |

FIADD = Integer Add

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 000 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------------|-------------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DA m0rm | CD 1A m0rm | 125 - EA (108-143) - EA | ST ← ST + mem-op (short integer) |
| 9B DE m0rm | CD 1E m0rm | 120 - EA (102-137) - EA | ST ← ST + mem-op (word integer) |

FICOM = Integer Compare

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 010 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|-------------------------|--------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DA m2rm | CD 1A m2rm | 85 - EA (78-91) + EA | ST ← mem-op (short integer) |
| 9B DE m2rm | CD 1E m2rm | 80 - EA (72-86) + EA | ST ← mem-op (word integer) |

FICOMP = Integer Compare and Pop

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 011 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|-------------------------|---|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DA m3rm | CD 1A m3rm | 87 + EA (80-93) + EA | ST ← mem-op pop stack (short integer) |
| 9B DE m3rm | CD 1E m3rm | 82 + EA (74-88) + EA | ST ← mem-op pop stack (word integer) |

FIDIV = Integer Divide

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 110 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------------|-----------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DA m6rm | CD 1A m6rm | 236 + EA (230-243) + EA | ST ← ST/mem-op (short integer) |
| 9B DE m6rm | CD 1E m6rm | 230 + EA (224-238) + EA | ST ← ST/mem-op (word integer) |

FIDIVR = Integer Divide Reversed

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 111 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------------|-----------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DA m7rm | CD 1A m7rm | 237 + EA (231-245) + EA | ST ← mem-op/ST (short integer) |
| 9B DE m7rm | CD 1E m7rm | 230 + EA (225-239) + EA | ST ← mem-op/ST (word integer) |

FILD = Integer Load Word Integer or Short Integer

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 000 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|-------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DB m0rm | CD 1B m0rm | 96 - EA (82-80) + EA | push stack ST ← mem-op (short integer) |
| 9B DF m0rm | CD 1F m0rm | 50 - EA (46-54) + EA | push stack ST ← mem-op (word integer) |

Long Integer

| | | | | |
|------|-----|---------|-------|-------|
| WAIT | op1 | mod 101 | addr1 | addr2 |
|------|-----|---------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|-------------------------|---|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 9B DF m5rm | CD 1F m5rm | 64 - EA (60-68) + EA | push stack ST ← mem-op (long integer) |

FIMUL = Integer Multiply

| | | | | |
|------|-----|------------|-------|-------|
| WAIT | op1 | mod001 r/m | addr1 | addr2 |
|------|-----|------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|-------------------------|----------------------------------|
| 80B7 Encoding | Emulator Encoding | 136 + EA (130-144) + EA | ST ← ST * mem-op (short integer) |
| 9B DA m1rm | CD 1A m1rm | 130 + EA (124-138) + EA | ST ← ST * mem-op (word integer) |

FINCSTP = Increment Stack Pointer

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|-----------------------------------|
| 80B7 Encoding | Emulator Encoding | 9 | stack pointer ← stack pointer + 1 |
| 9B D9 F7 | CD 19 F7 | 6-12 | |

FINIT = Initialize Processor

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|---------------------------|
| 80B7 Encoding | Emulator Encoding | 5 | initialize 80B7 |
| 9B DB E3 | CD 1B E3 | 2-8 | |
| 9B DB E3 | CD 1B E3 | 5 | initialize 80B7 (no wait) |
| | | 2-8 | |

FIST = Integer Store

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 010 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------|-----------------------------|
| 80B7 Encoding | Emulator Encoding | 88 - EA (82-92) - EA | mem-op → ST (short integer) |
| 9B DB m2rm | CD 1B m2rm | 86 - EA (80-90) - EA | mem-op → ST (word integer) |

FISTP = Integer Store and Pop

Short Integer or Word Integer

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 011 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------|--|
| 80B7 Encoding | Emulator Encoding | 90 - EA (84-94) - EA | mem-op → ST pop stack (short integer) |
| 9B DF m3rm | CD 1F m3rm | 88 - EA (82-92) - EA | mem-op → ST pop stack (word integer) |

Long Integer

| | | | | |
|------|-----|---------|-------|-------|
| WAIT | op1 | mod 111 | addr1 | addr2 |
|------|-----|---------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|------------------------|---|
| 80B7 Encoding | Emulator Encoding | 100 - EA (94-105) - EA | mem-op → ST pop stack (long integer) |

FISUB = Integer Subtract

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 100 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|-------------------------|----------------------------------|
| 80B7 Encoding | Emulator Encoding | 125 - EA (108-143) - EA | ST ← ST - mem-op (short integer) |
| 9B DA m4rm | CD 1A m4rm | 120 - EA (102-137) - EA | ST ← ST - mem-op (word integer) |

FISUBR = Integer Subtract Reversed

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 101 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

Execution Clocks

| 80B7 Encoding | Emulator Encoding | Typical Range | Operation |
|---------------|-------------------|-------------------------|----------------------------------|
| 9B DA m5rm | CD 1A m5rm | 125 - EA (109-144) - EA | ST ← mem-op - ST (short integer) |
| 9B DE m5rm | CD 1E m5rm | 120 - EA (103-139) - EA | ST ← mem-op - ST (word integer) |

FLD = Load Real

Stack element to Stack top

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|----------------------|
| 80B7 Encoding | Emulator Encoding | 20 | T ← ST(0) |
| 9B D9 C0 - | CD 19 C0 - | 17-22 | push stack ST ← T |

Memory operand to Stack top
Short Integer or Long Integer

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 000 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------|---|
| 80B7 Encoding | Emulator Encoding | 43 - EA (38-56) - EA | push stack ST ← mem-op (short integer) |
| 9B D9 m0rm | CD 19 m0rm | 46 - EA (40-60) - EA | push stack ST ← mem-op (long integer) |

Temp Real

| | | | | |
|------|-----|---------|-------|-------|
| WAIT | op1 | mod 101 | addr1 | addr2 |
|------|-----|---------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------|---------------------------------------|
| 80B7 Encoding | Emulator Encoding | 57 - EA (53-65) - EA | push stack ST ← mem-op (temp real) |

FLD1 = Load + 1.0

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|------------------------|
| 80B7 Encoding | Emulator Encoding | 18 | push stack ST ← 1.0 |
| 9B D9 E8 | CD 19 E8 | 15-21 | |

FLDCW = Load Control Word

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 101 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------------|---------------------------------|
| 80B7 Encoding | Emulator Encoding | 10 - EA (7-14) - EA | processor control word ← mem-op |
| 9B D9 m5rm | CD 19 m5rm | | |

FLDENV = Load Environment

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 100 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|----------------------|---------------------------|
| 80B7 Encoding | Emulator Encoding | 40 - EA (35-45) - EA | 80B7 environment ← mem-op |
| 9B D9 m4rm | CD 19 m4rm | | |

FLDL2E = Load Log_ee

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | Typical Range | Operation |
|------------------|-------------------|---------------|---------------------------------------|
| 80B7 Encoding | Emulator Encoding | 18 | push stack ST ← log _e e |
| 9B D9 EA | CD 19 EA | 15-21 | |

FLDL2T = Load Log_e10

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|--------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 E9 | CD 19 E9 | 19 16-22 | push stack ST ← op 10 |

FLDLG2 = Load Log₁₀2

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|-------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 EC | CD 19 EC | 21 18-24 | push stack ST ← op 2 |

FLDPI = Load π

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|----------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 EB | CD 19 EB | 19 16-22 | push stack ST ← r |

FLDZ = Load + 0.0

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 EE | CD 19 EE | 14 11-17 | push stack ST ← 0.0 |

FMUL = Multiply Real

Stack top and Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 ← r |
|------|-----|---------|

| Execution Clocks | | | Operation |
|------------------|-------------------|----------------|--------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D8 C8 ← r | CD 18 C8 ← r | 138 130-145 | ST ← ST * ST(r) |
| 9B DC C8 ← r | CD 1C C8 ← r | 138 130-145 | ST(r) ← ST(r) * ST |

Stack top and memory operand

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 001 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | | Operation |
|------------------|-------------------|----------------------------|----------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D8 m1r m | CD 18 m1r m | 118 - EA (110-125) - EA | ST ← ST * mem-op (short real) |
| 9B DC m1r m | CD 1C m1r m | 161 - EA (154-168) - EA | ST ← ST * mem-op (long real) |

FMULP = Multiply Real and Pop

| | | |
|------|-----|---------|
| WAIT | op1 | op2 ← r |
|------|-----|---------|

| Execution Clocks | | | Operation |
|------------------|-------------------|----------------|---------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B DE C9 ← r | CD 1E C9 ← r | 142 134-148 | ST(r) ← ST(r) * ST pop stack |

FNOP = No Operation

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|-----------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 D0 | CD 19 D0 | 13 10-16 | ST ← ST |

FPATAN = Partial Arc tangent

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|----------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 F3 | CD 19 F3 | 650 250-800 | T ← arctan(ST ₁ /ST) pop stack ST ← T |

FPREM = Partial Remainder

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 F8 | CD 19 F8 | 125 15-190 | ST ← REPEAT (ST - ST ₁) |

FPTAN = Partial Tangent

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|---|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 F2 | CD 19 F2 | 450 30-540 | Y/X ← TAN(ST) ST ← Y push stack ST ← X |

FRNDINT = Round to Integer

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 FC | CD 19 FC | 45 16-50 | ST ← nearest integer (ST) |

FRSTOR = Restore Saved State

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 100 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | | Operation |
|------------------|-------------------|----------------------------|---------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B DD m4r m | CD 1D m4r m | 202 - EA (197-207) - EA | 8087 state ← mem-op |

FSAVE = Save State

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 110 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | | Operation |
|------------------|-------------------|----------------------------|----------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B DD m4r m | CD 1D m4r m | 202 - EA (197-207) - EA | mem-op ← 8087 state |
| 9B DD m4r m | CD 1D m4r m | 202 - EA (197-207) - EA | mem-op ← 8087 state (no wait) |

FSCALE = Scale

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|---------------|-------------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 FD | CD 19 FD | 35 32-38 | ST ← ST * 2 |

FSQRT = Square Root

| | | |
|------|-----|-----|
| WAIT | op1 | op2 |
|------|-----|-----|

| Execution Clocks | | | Operation |
|------------------|-------------------|----------------|-----------|
| 8087 Encoding | Emulator Encoding | Typical Range | |
| 9B D9 FA | CD 19 FA | 183 180-186 | ST ← √ ST |

FST = Store Real

Stack top to Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------|------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 DD 00 + i | CD 1D 00 + i | 18 15-22 | ST(i) - ST |

Stack top to memory operand

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 010 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------------------|-----------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 D9 m2m | CD 19 m2m | 87 + EA (84-90) + EA | mem-op - ST (short-real) |
| 98 D0 m2m | CD 1D m2m | 100 + EA (96-104) + EA | mem-op - ST (long-real) |

FSTCW = Store Control Word

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 111 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Operation | |
|------------------|-------------------|-------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 D6 m7m | CD 19 m7m | 15 + EA (12-18) + EA | mem-op - processor control word |
| 98 D9 m7m | CD 19 m7m | 15 + EA (12-18) + EA | mem-op - processor control word (no wait) |

FSTENV = Store Environment

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 110 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Operation | |
|------------------|-------------------|-------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 D9 m6m | CD 19 m6m | 45 + EA (40-50) + EA | mem-op - 6087 environment |
| 98 D9 r6m | CD 19 m6m | 45 + EA (40-50) + EA | mem-op - 6087 environment (no wait) |

FSTP = Store Real and Pop

Stack top to Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------|-------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 DD D8 + i | CD 1D D8 + i | 20 17-24 | ST(i) - ST pop stack |

Stack top to memory operand

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 011 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

Long Real or Short Real

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 D9 m3m | CD 19 m3m | 89 + EA (86-92) + EA | mem-op - ST pop stack (short-real) |
| 98 DB m3m | CD 1B m3m | 102 + EA (98-106) + EA | mem-op - ST pop stack (long-real) |

Temp Real

| | | | | |
|------|-----|-------------|---------|---------|
| WAIT | op1 | mod 111 r/m | disp-lo | disp-hi |
|------|-----|-------------|---------|---------|

| Execution Clocks | | Operation | |
|------------------|-------------------|-------------------------|---|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 DD m7m | CD 1D m7m | 55 + EA (52-58) + EA | mem-op - ST pop stack (temp-real) |

FSTSW = Store Status Word

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 111 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Operation | |
|------------------|-------------------|-------------------------|--|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 DD m7m | CD 1D m7m | 15 + EA (12-18) + EA | mem-op - 8087 status word |
| 98 DD m7m | CD 1D m7m | 15 + EA (12-18) + EA | mem-op - 8087 status word (no wait) |

FSUB = Subtract Real

Stack top and Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------|--------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 D8 E0 + i | CD 18 E0 + i | 85 70-100 | ST - ST - ST(i) |
| 98 DC E8 + i | CD 1C E8 + i | 85 70-100 | ST(i) - ST(i) - ST |

Stack top and memory operand

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 100 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------------------|----------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 D8 m4m | CD 18 m4m | 105 + EA (90-120) + EA | ST - ST - mem-op (short-real) |
| 98 DC m4m | CD 1C m4m | 110 + EA (95-125) + EA | ST - ST - mem-op (long-real) |

FSUBP = Subtract Real and Pop

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------|---------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 DE E9 | CD 1E E9 | 90 75-105 | ST(i) - ST(i) - ST pop stack |
| 98 DE E8 + i | CD 1E E8 + i | 90 75-105 | ST(i) - ST(i) - ST pop stack |

FSUBR = Subtract Real Reversed

Stack top and Stack element

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------|--------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 D8 E8 + i | CD D8 E8 + i | 87 70-100 | ST - ST(i) - ST |
| 98 DC E0 + i | CD 1C E0 + i | 87 70-100 | ST(i) - ST - ST(i) |

Stack top and memory operand

| | | | | |
|------|-----|-------------|-------|-------|
| WAIT | op1 | mod 101 r/m | addr1 | addr2 |
|------|-----|-------------|-------|-------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------------------|----------------------------------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |
| 98 D8 m5m | CD 18 m5m | 105 + EA (90-120) + EA | ST - mem-op - ST (short-real) |
| 98 DC m5m | CD 1C m5m | 110 + EA (95-125) + EA | ST - mem-op - ST (long-real) |

FSUBRP = Subtract Real Reversed and Pop

| | | |
|------|-----|---------|
| WAIT | op1 | op2 + i |
|------|-----|---------|

| Execution Clocks | | Operation | |
|------------------|-------------------|---------------|-----------|
| 8087 Encoding | Emulator Encoding | Typical Range | Operation |

9B DE E1 CD 1E E1 90 ST(i) ← ST - ST(i)
 75-105 pop stack
 9B DE E0 ← CD 1E E0 ← 90 ST(i) ← ST - ST(i)
 75-105 pop stack

FTST = Test Stack Top Against + 0.0

| WAIT | op1 | op2 |
|------|-----|-----|
|------|-----|-----|

| 8087 Encoding | Emulator Encoding | Execution Clocks Typical Range | Operation |
|---------------|-------------------|--------------------------------|---------------|
| 9B D9 E4 | CD 19 E4 | 42 38-48 | ST ← ST - 0.0 |

FWAIT = (CPU) Wait While 8087 is Busy

| WAIT |
|------|
|------|

| 8087 Encoding | Emulator Encoding | Execution Clocks Typical Range | Operation |
|---------------|-------------------|--------------------------------|-----------------------|
| 9B | 90 | 3-5n 3-5n | 8086 wait instruction |

FXAM = Examine Stack Top

| WAIT | op1 | op2 |
|------|-----|-----|
|------|-----|-----|

| 8087 Encoding | Emulator Encoding | Execution Clocks Typical Range | Operation |
|---------------|-------------------|--------------------------------|--------------------|
| 9B D9 E5 | CD 19 E5 | 17 12-23 | set condition code |

FXCH = Exchange Registers

| WAIT | op1 | op2 + i |
|------|-----|---------|
|------|-----|---------|

| 8087 Encoding | Emulator Encoding | Execution Clocks Typical Range | Operation |
|---------------|-------------------|--------------------------------|---|
| 9B D9 C8 | CD 19 C8 | 12 10-15 | T _i ← ST(i) ST(i) ← ST ST ← T _i |
| 9B D9 C8 ← i | CD 19 C8 ← i | 12 10-15 | T _i ← ST(i) ST(i) ← ST ST ← T _i |

EXTRACT = Extract Exponent and Significand

| WAIT | op1 | op2 |
|------|-----|-----|
|------|-----|-----|

| 8087 Encoding | Emulator Encoding | Execution Clocks Typical Range | Operation |
|---------------|-------------------|--------------------------------|---|
| 9B D9 F4 | CD 19 F4 | 50 27-55 | T _i ← exponent (ST) T _i ← significand (ST) ST ← T _i push stack ST ← T _i |

FYL2X = Compute Y * Log₂ X

| WAIT | op1 | op2 |
|------|-----|-----|
|------|-----|-----|

| 8087 Encoding | Emulator Encoding | Execution Clocks Typical Range | Operation |
|---------------|-------------------|--------------------------------|--|
| 9B D9 F1 | CD 19 F1 | 950 900-1100 | T _i ← ST(i) * log ₂ (ST) pop stack ST ← T _i |

FYL2XP1 = Compute Y * Log₂ (X + 1)

| WAIT | op1 | op2 |
|------|-----|-----|
|------|-----|-----|

| 8087 Encoding | Emulator Encoding | Execution Clocks Typical Range | Operation |
|---------------|-------------------|--------------------------------|---|
| 9B D9 F9 | CD 19 F9 | 850 700-1000 | T _i ← ST + 1 T _i ← ST(i) * log ₂ T _i pop stack ST ← T _i |

INDEX

- 2, powers of, 2
- 2's complement, 6-8, 43, 55, 56, 148
- 4-bit microprocessors, 26, 27
- 4N33 optical coupler, 228
- 5-minute rule, 68, 185, 197, 315, 574
- 7-segment display code, 4
- 7-segment light-emitting diodes (LEDs), 4, 267-276
- 8-bit microprocessors, 26-27
- 10BaseT (thin Ethernet) networks, 525
- 16-bit microprocessors, 27-28
- 18-segment light-emitting diodes (LEDs), 267, 276
- 32-bit microprocessors, 27
- 74LS14 inverter, 217, 219, 220
- 74LS138 decoder, 186-189, 222, 223
- 74LS164 wait-state generator, 174, 175, 177, 196-197
- 74LS181 ALU, 16-17
- 74LS244 drivers, 174, 179
- 74LS393 baud rate generator, 174, 175, 178, 184
- 74S373 address latch, 174, 175, 178, 185
- 555 timer, 220
- 741 operational amplifier, 291, 293
- 2142 SRAM, 174, 175, 181, 192-193
- 2316/2716 PROM, 174, 175, 176, 191-192
- 2900 family of bit-slice processors, 27
- 3625 I/O decoder, 174, 182, 193-195
- 3625 off-board decoder, 174, 180, 195-196
- 3625 PROM decoder, 174, 175, 176, 191-192
- 3625 RAM decoder, 174, 175, 181, 192-193
- 4004 microprocessor, 26
- 6502 microprocessor, 27
- 6800 microprocessor, 27
- 6801 microcontroller, 27
- 6809 microprocessor, 27
- 6845 CRT controller, 442-443
- 7445 decoder, 174, 182, 269
- 7447 decoder, 267-268, 269
- 8008 microprocessor, 26-27
- 8048 microcontroller, 27, 265-266
- 8051 microcontroller family, 27, 332, 333
- 8080 microprocessor, 27
- 8085 microprocessor, 27
- 8086 assembly language
 - breakpoints in, 61
 - coding sheets for, 45-47
 - comments in, 32, 45, 46
 - converting algorithms to, 67
 - destinations in, 32-33
 - fields in, 32
 - instructions for. *See* 8086 instructions
 - labels in, 45, 46, 58
 - mnemonics in, 32, 45, 46
 - opcodes (operation codes) for, 32, 45, 46, 48, 590-598
 - operands in, 32, 45, 46
 - overview of, 32-35
 - program development tools for, 59-62
 - programs. *See* 8086 assembly language programs
- 8086 assembly language (Cont.)
 - sources in, 32-33
 - statements in, 32-33, 45-47
 - syntax of, 47
- 8086 assembly language programs
 - 8086 initialization, 237-240
 - 8250 initialization, 513-514
 - 8251A data transmission and reception, 493
 - 8251A initialization, 492
 - 8254 initialization, 237-240
 - 8259A initialization, 237-240
 - 8279 initialization, 273, 274
 - adding constants to arrays of data, 86-89, 401, 402-403
 - arithmetic average, 65-67
 - backward jump, 73-74
 - Basic Input Output System (BIOS) procedure calls, 240-242
 - binary-coded decimal (BCD) packing, 69-71
 - binary-coded decimal (BCD) to binary conversion, 109-115
 - breakpoints in, 47, 61, 68-69
 - C programs with, 426-432
 - circular (ring) buffer, 513-514
 - comparing strings, 97-99
 - conditional jumps, 77
 - critical region protection, 538
 - data sampling, 103-106
 - debugging. *See* Debugging programs
 - display driver, 274-275
 - display programming, 451-458
 - divide-by-zero interrupt-service procedure, 209-212
 - division, 122-127
 - documentation of, 47
 - DOS function calls for files, 475
 - factorials, 117-121
 - far procedure additions to, 121
 - forward jump, 74
 - hand-coding, 35, 45-47, 52, 53, 68
 - heater control, 82-84
 - IF-THEN-ELSE structure, 77-82
 - IF-THEN structure, 77
 - industrial process-control system, 323-331
 - inflation factor adjustment, 86-88
 - initialization instructions for, 45, 66, 67
 - int array pointers, 401, 402-403
 - interrupt input, 217-219
 - keyboard input, 84-86, 217-219, 237-240, 262-264
 - microcomputer-based industrial process-control system, 323-331
 - microcomputer-based scale, 309-317
 - modules of, 122
 - moving strings, 95-97
 - multiplication, 53-59
 - mutual exclusion, 538
 - packed binary-coded decimal (BCD) code, 69-71, 109-115
 - printed-circuit-board-making machine, 77-82
 - printer driver, 255-259
 - printer output, 240-242
 - profit factor adjustment, 88-89, 401, 402-403
- 8086 assembly language programs (Cont.)
 - read input port, 45-47
 - real-time clock, 237-240
 - relocatable, 74
 - REPEAT-UNTIL structure, 84-89
 - scale, 309-317
 - sequence structure, 65-67, 69-71
 - simplified segment directives in, 430, 432
 - stack usage additions to, 103
 - strobed input, 84-86
 - terminal emulator, 507-508, 509
 - unconditional jumps, 73-74
 - video programming, 451-458
 - WHILE-DO structure, 82-84
- 8086-based microcomputers. *See also* SDK-86
 - block diagram of, 163-164
 - bus activities during read and write operations, 164-167
 - overview of, 163-164
 - troubleshooting, 200-204
- 8086 family of microprocessors, 27-28
- 8086 instructions
 - arithmetic, 42-43, 67-68, 590
 - bit manipulation, 43, 590-591
 - clock cycles for, 592-598
 - D bit for, 48, 49, 590, 591
 - data transfer, 42, 590
 - initialization, 45, 66, 67
 - mnemonics for, 32, 45, 46
 - MOD bit patterns for, 48-50, 590, 591, 592
 - opcodes (operation codes) for, 32, 35, 46, 48, 590-598
 - operand summary for, 592
 - processor control, 44-45, 591
 - program execution transfer, 43-44, 591
 - R/M bit patterns for, 48-50, 590, 591, 592
 - REG bits for, 48, 49, 590, 591, 592
 - second instruction byte summary for, 592
 - string, 43, 95-99, 591
 - supersets of, 28
 - templates for, 47-53
 - timing for, 91-93, 592-598
 - W bit for, 48, 590, 591
- 8086 microprocessor
 - 8087 math coprocessor cooperation with, 377-379
 - 8088 microprocessor versus, 197
 - a.c. characteristics of, 585-589
 - absolute maximum ratings for, 585
 - accumulator of, 29-30
 - arithmetic logic unit (ALU) of, 28, 29
 - assembly language for. *See* 8086 assembly language
 - block diagram of, 27-28, 579
 - booting, 164, 168
 - bus activities during read and write operations, 164-167
 - bus interface unit (BIU) of, 28-31, 581
 - bus timing for, 584
 - buses in, 29, 582

- 8086 microprocessor (Cont.)**
 clock frequencies for, 197
 control circuitry in, 28, 29
 d.c. characteristics of, 585
 decoder in, 28, 29
 description of, 27-28, 581-584
 direct addressing mode of, 34-35
 execution unit (EU) of, 28-30, 32, 581
 external interface for, 583-584
 flag register of, 28-29
 general-purpose registers of, 29-30
 high byte storage of, 35
 I/O addressing of, 582
 immediate addressing mode of, 33
 index registers of, 29, 32
 instruction pointer (IP) register of, 29, 30-31
 instructions for. *See* 8086 instructions
 interrupt response of, 207-208
 low byte storage of, 35
 math coprocessor for. *See* 8087 math coprocessor
 maximum mode of, 168, 346, 348, 581-582, 588, 589
 memory organization of, 189-191, 581
 minimum mode of, 168, 197-200, 346, 348, 581-582, 586, 587, 589
 pin configuration of, 168, 348, 579
 pin description for, 167-168, 579-580
 pointer registers of, 29, 30-31
 programmer's model of, 163
 programming introduction for, 32-35
 queue registers of, 29, 30
 READY input of, 164-167, 173, 175, 196-197
 register addressing mode of, 33-34
 reserved memory locations for, 581
 RESET response of, 164, 168, 583
 segment registers of, 29, 30-32, 34-35, 58-59, 581
 stack pointer (SP) register of, 29, 31
 stack segment (SS) register of, 29, 30-32
 system timing for, 164-167, 583, 584
 TEST input of, 375, 377-379, 584
 timing parameters for, 197-200
 virtual 8086 mode (80386), 561-562
 WAIT instruction for, 375, 377-379
 WAIT states of, 164-167, 175, 196-197
 waveforms for, 164-167, 197-200, 583, 584, 586-589
 word storage of, 35
- 8087 math coprocessor**
 8086 microprocessor cooperation with, 377-379
 architecture of, 368-369
 block diagram of, 368
 circuit connections for, 377-379
 clock cycles for instructions, 599-605
 control words for, 368-369
 data types for, 366-368
 double-precision numbers for, 366, 367-368
 fixed-point numbers for, 367
 floating-point numbers for, 367
 hypotenuse calculation with, 375-377
- 8087 math coprocessor (Cont.)**
 instructions for, 370-375, 599-605
 integers for, 366
 overview of, 365-366
 packed decimal numbers for, 366
 programming example for, 375-377
 real numbers for, 366-368
 single-precision numbers for, 366, 367
 socket for, 345
 stack of, 369-370
 status word for, 368-369
 timing for instructions, 599-605
 8086 microprocessor, 28, 197, 377-379
 8096 microcontroller family, 27, 332-333, 334
 82C08 DRAM controller, 355-357
 8237 DMA controller
 circuit connections and operation, 348-351
 initializing, 352
 pin descriptions and signals for, 352-353
 timing waveforms for, 351-352
 8250 UART
 control words for, 515, 516
 initializing, 515
 sending and receiving characters with, 517
 8251A USART, 174, 175, 184
 BISO communication with, 519-520
 block diagram of, 489, 490
 control words for, 490-493, 519, 520
 initializing, 490-493
 pin descriptions for, 489, 490
 sending and receiving characters with, 493
 status word for, 491
 system connections and signals, 489-490
 write-recovery time for, 492
 8253 programmable timer/counter, 221
 8254 programmable timer/counter
 8253 versus, 221
 block diagram of, 221
 clock frequency for, 221
 control words for, 224-226, 232
 hardware-retriggerable one-shot, 227-228
 hardware-triggered strobe, 230-231
 initializing, 223-226
 internal addresses of, 224
 interrupt on terminal count, 226-227
 nonsystem clocks with, 231
 operation of, 221-223
 read-back feature of, 221, 232
 reading the count from, 231-232
 SDK-86 addition of, 222, 223
 software-triggered strobe, 230, 231
 square-wave mode (mode 3), 229-230
 system connections for, 222, 223
 timed interrupt generator, 228-229
 timing waveforms for, 226-231
 8255A programmable parallel port
 block diagram of, 248
 Centronics parallel interface to, 252, 254-255
 control words for, 248-250, 255
 I/O interface using, 250-252
 operational modes of, 247-250
 paper tape reader interface using, 250-252
- 8255A programmable parallel port (Cont.)**
 printer driver program using, 255-259
 in SDK-86, 174, 175, 180
 status word for, 258-260
 system connections for, 247
 8259A priority interrupt controller (PIC)
 8086 interrupt-acknowledge cycles for, 215-216
 cascading, 222, 235
 fixed-priority mode of, 233-234
 in-service register (ISR) of, 233, 234
 initialization command words (ICWs) of, 235-237
 initializing, 235-240
 interrupt mask register (IMR) of, 233, 234
 interrupt request register (IRR) of, 233, 234
 operation command words (OCWs) of, 237, 238
 overview of, 233-235
 priority resolver of, 233, 234
 SDK-86 addition of, 222, 223, 235
 system connections for, 222, 223, 225
 use of, 232-233
 8272A floppy-disk controller, 469-472
 8279 dedicated display controller
 7-segment display interfacing with, 268-275
 connections for, 268-271
 control words for, 271-274
 decoded scan with, 271, 273
 encoded scan with, 271, 273
 initializing and communicating with, 270
 keypad interfacing with, 268-274
 scan time for, 270
 status word for, 274
 timing waveforms for, 269-271
 8279 specialized I/O device, 174, 178-179, 182
 8284 clock generator, 173, 174, 175, 177
 8286 control and data transceivers, 174, 179
 8421-binary-coded decimal (BCD) code, 3, 4
 8514/A high-resolution graphics board, 447
 9900 family of microprocessors, 27
 32032 microprocessor, 27
 68000 microprocessor, 27
 68020 microprocessor, 27
 80186 and 80188 microprocessors, 28, 333, 335-336, 534, 598-599
 80286 microprocessor
 80287 coprocessor for, 544
 architecture of, 543
 block diagram of, 543
 instructions for, 546-547
 interrupts for, 544, 545
 introduced, 28, 534, 543
 pin configuration of, 544
 protected mode of, 543, 545-546, 547
 real mode of, 543, 544, 546-547
 system connections for, 545
 80287 math coprocessor, 544
 80386 microprocessor
 80386DX version of, 547, 548
 80386SX version of, 547, 548
 addressing modes of, 563
 architecture of, 547
 extended industry standard architecture (EISA) bus for, 550-551

- 80386 microprocessor (*Cont.*)
 - index scaling feature of, 563
 - industry standard architecture (ISA)
 - bus for, 550
 - instructions for, 562-564
 - introduced, 28, 534, 547
 - MicroChannel Architecture (MCA)
 - bus for, 551-552
 - pin configuration of, 547, 548
 - programming, 564-568
 - protected mode of. *See* 80386 protected mode
 - real mode of, 552, 553
 - registers of, 552, 553, 562-563
 - signal groups of, 547-549
 - summary of hardware and modes, 562
 - system connections for, 549-550
 - systems using, 549-550
 - virtual 8086 mode of, 561-562
- 80386 protected mode
 - call gates, 557
 - flat system memory model, 560
 - input/output (I/O) privilege levels, 557
 - interrupt and exception handling, 557-558
 - operating systems using, 566
 - paged flat memory model, 560-561
 - paging mode, 558-560
 - segment privilege levels and protection, 556-557
 - segmentation, 554-556
 - segmented-paged memory model, 560
 - segments-only memory model, 560
 - task state segment, 558-559
 - task switching, 558-559
 - virtual memory, 554-556
- 80386DX microprocessor, 547, 548
- 80386SX microprocessor, 547, 548
- 80387 math coprocessor, 569
- 80486 built-in math coprocessor, 568-569
- 80486 microprocessor, 534, 568-570
- 80586 microprocessor, 570-571
- 80686 microprocessor, 571
- 80786 built-in math coprocessor, 571
- 80786 microprocessor, 571
- 80960 microcontroller family, 336
- 82385 cache controller, 358, 360-363
- A-bus, 29
- AAA instruction, 42, 131, 590, 592
- AAD instruction, 43, 131, 590, 592
- AAM instruction, 43, 131-132, 590, 592
- AAS instruction, 43, 132, 590, 592
- Absolute shaft encoders, 283-284
- Abstracts for programs, 47
- Access times for disks, 466, 467
- Accessed bit, 542
- Accumulator (AL register), 29-30
- Accuracy (precision) of numbers, 367
- ACK (affirmative acknowledge) character, 519
- Active filters, 291, 294-295
- Actual arguments (parameters) of functions, 418
- A/D converters. *See* Analog-to-digital converters
- ADC instruction, 42, 132, 590, 592-593
- ADD instruction, 42, 52, 70-71, 132-133, 590, 593
- Adders, 291, 293-294
- Addition
 - 8086 instructions for, 42
 - 8087 instructions for, 372-373
 - binary, 6-7
 - binary-coded decimal (BCD), 10
- Address bus, 14, 15, 23-26
- Address counter, 24
- Address decoders, 185-186, 335-336
- Address inputs, 14, 15
- Address marks on disks, 468-469
- Address pipelining, 548-549
- Address transfer instructions, 42
- Address unit (AU) (80286), 543
- Addresses
 - base, 30, 31, 32, 34, 49-50, 89-90
 - breakpoint, 47, 61, 68-69, 214, 552, 553
 - effective (EA), 34, 49-50, 89-90, 592
 - internal, 224
 - labels as, 45, 46, 58
 - logical, 541-543
 - named, 58
 - offsets (displacements) of, 31, 34, 88-90, 542
 - physical, 31-32, 34, 49, 89-90, 541-543
 - pipelined, 548-549
 - return, 100, 133
 - segment base, 30, 31, 32, 34, 49-50, 89-90
 - segment base: offset form of, 31, 35
 - virtual, 541-543
- Addressing modes
 - 80386, 563
 - defined, 33
 - direct, 34-35
 - double indexed, 49, 89-90
 - immediate, 33
 - MOD bit patterns for, 48-50
 - R/M bit patterns for, 48-50
 - real, 28
 - register, 33-34
 - single indexed, 49, 89-90
 - summary of, 49-50, 89-90
 - virtual, 28
- Advanced Micro Devices 2900 family of bit-slice processors, 27
- AF (auxiliary carry flag), 28, 29, 74, 75
- Affirmative acknowledge (ACK) character, 519
- AH register, 29-30
- AI (artificial intelligence), 572-574
- AL register (accumulator), 29-30
- Algorithms
 - converting to assembly language, 67
 - defined, 37
 - downloading program, 510
 - flowcharts for, 37-41
 - microcomputer-based scale, 308-309
 - program development, 61, 62
 - standard structures for, 39-42
 - structured programming of, 39-42
 - terminal emulator, 507
 - top-down design of, 39-42
- Align on even memory address (EVEN) directive, 159-160
- Alphanumeric codes, 4-6
- Alphanumeric displays. *See* Light-emitting diodes; Liquid-crystal displays
- ALU. *See* Arithmetic logic unit
- AM (amplitude modulation), 498, 499
- American Standard Code for Information Interchange (ASCII)
 - ASCII string, 475
- American Standard Code for Information Interchange (ASCII) (*Cont.*)
 - converting EBCDIC to, 266-267
 - in data statements, 56
 - described, 5-6
 - extended, 435-439
 - keyboard input, 84-86
 - table of codes, 5, 436
- Amplifiers. *See* Operational amplifiers
- Amplitude modulation (AM), 498, 499
- Analog-to-digital (A/D) converters
 - 8096, 333, 334
 - conversion time for, 304
 - dual-slope, 304-305, 307
 - high-speed, 304
 - microcomputer interfacing for, 306-307
 - output codes for, 306
 - parallel comparator (flash), 304, 306-307
 - specifications for, 304
 - successive approximation, 305, 307
- AND gate, 11
- AND instruction, 43, 69-70, 133, 590, 593
- AND matrixes, 12
- Answer modem, 501
- APIs (application program interfaces), 567
- Application layer (OSI model), 523
- Application program interfaces (APIs), 567
- Arithmetic average, 65-67
- Arithmetic instructions
 - 8086, 42-43, 67-68, 590
 - 8087, 372-374
- Arithmetic logic unit (ALU)
 - 74LS181, 16-17
 - 8086, 28, 29
 - defined, 16-17
 - microprocessor categorization using, 26
- Arithmetic operators in C, 406
- ARPL instruction (80286), 547
- Arrays
 - data, 86-89
 - elements of, 86-89
 - indexes of, 89
 - type of, 86
- Artificial intelligence (AI), 572-574
- ASCII. *See* American Standard Code for Information Interchange
- Assembler directives
 - align on even memory address (EVEN), 159-160
 - ASSUME, 58-59, 158
 - define byte (DB), 55, 158
 - define doubleword (DD), 55, 158
 - define quadword (DQ), 158
 - define ten bytes (DT), 158-159
 - define word (DW), 55, 159
 - end procedure (ENDP), 159
 - end program (END), 59, 159
 - end segment (ENDS), 53-54, 159
 - equate (EQU), 54-55, 159
 - external (EXTRN), 122, 160, 162
 - GLOBAL, 160
 - GROUP, 160
 - INCLUDE, 160
 - LABEL, 160
 - LENGTH operator, 161
 - NAME, 161
 - OFFSET operator, 161
 - originate (ORG), 161
 - pointer (PTR), 161
 - procedure (PROC), 161

Assembler directives (Cont.)

PUBLIC. 122, 160, 161–162
SEGMENT. 53–54, 162
SHORT operator. 162
TYPE operator. 162

Assembler macros
defined. 127
dummy variables in. 128
expanding. 127
in-line code and. 127
passing parameters to. 128
procedures versus. 127, 128–129
without parameters. 127–128

Assembler program. 33, 45, 53, 602

Assembly language. *See* 8086 assembly language

Assertion level. 11

Assignment operator in C. 406

ASSUME directive. 58–59, 158

Asynchronous communication. 488

Asynchronous inputs. 13

Attribute code. 442–443

AU (address unit) (80286). 543

Audio speaker buffer. 230

Audio-tone generators. 230

Automatic storage class in C. 420

Auxiliary carry. 10

Auxiliary carry flag (AF). 28, 29, 74, 75

Available interrupts. 208

Average of numbers. 65–67

AX register. 30

B-bus, 29

Backbones for networks. 526, 527

Backward jump. 72–74

Bandpass filters. 295

Bang-bang (on-off) control. 331

Bank-switched memory. 539–540

Base-2 numbers. *See* Binary numbers

Base-10 (decimal) numbers. 1

Base-16 (hexadecimal) numbers. 3, 9–10

Base address. 30, 31, 32, 34, 49–50, 89–90

Base pointer (BP) register. 29, 31

Baseband transmission. 523

Basic Input Output System (BIOS). 240–242, 386, 435–439, 451–453, 454, 506–518, 536–537

Baud rate. 178, 488

BCC (block check characters). 518–519

BCD. *See* Binary-coded decimal code

BCP (byte-oriented protocol). 518–520

Begin flowchart symbol. 37–38

Behavioral models. 382

BH register. 29–30

Biased exponent of numbers. 367

Bidirectional bus. 24

Binary addition. 6–7

Binary-coded decimal (BCD) code
8421. 3, 4
addition with. 10
in data statements. 56
decimal adjust operation for. 10
described. 3, 4, 10
excess-3. 4
packed. 69–71, 109–115, 366, 372
subtraction with. 10
unipolar. 306
unpacked. 69–71

Binary codes. 306

Binary counters. 14

Binary digits. *See* Bits

Binary division. 9

Binary multiplication. 8–9

Binary (base-2) numbers

2's complement of. 6–8, 43, 55, 56, 148

in data statements. 55
described. 1–3, 6–9
packed binary-coded decimal (BCD)
conversion to. 109–115

Binary subtraction. 8

Binary Synchronous Communication Protocol (BISYNC). 518–520

Binary-to-decimal conversion. 1–2

Binary words
binary digits (bits) in. *See* Bits
byte (8-bit). 1
doubleword (32-bit). 1
least significant bit (LSB) of. 1
most significant bit (MSB) of. 1, 2, 28, 29
nibble (4-bit). 1
word (16-bit). 1

Binder. 566

BIOS (Basic Input Output System). 240–242, 386, 425–439, 451–453, 454, 506–518, 536–537

Bipolar binary codes. 306

BISYNC (Binary Synchronous Communication Protocol). 518–520

Bit-aligned block transfer (BITBLT). 458

Bit manipulation instructions. 43, 590–591

Bit-mapped raster scan display. 444

Bit-oriented protocol (BOP). 520–522

Bit scan and test instructions (80386). 563

Bit-slice processors. 27

BITBLT (bit-aligned block transfer). 458

Bits (binary digits)
accessed. 542
check (encoding). 364
D. 48, 49, 590, 591
defined. 1
dibits. 499, 500
dirty. 542
flag. 518
masking. 69–70
MOD. 48–50, 590, 591, 592
parity. 4, 363–364
privilege-level. 542
quadbits. 499–500
R/M. 48–50, 590, 591, 592
REG. 48, 49, 590, 591, 592
sign. 6–8
start. 488
stop. 488
tribits. 499
W. 48, 590, 591

Bitwise operators in C. 406–407

BIU (bus interface unit). 28–31, 581

BL register. 29–30

Block check characters (BCC). 518–519

Blocked tasks. 538

Blocks of memory. 255

Boot record on disks. 473

Booting the 8086. 164, 168

BOP (bit-oriented protocol). 520–522

Bottom-up design. 39

BOUND instruction (80186/80188). 44, 336, 598

BOUND instruction (80286). 547

BP (base pointer) register. 29, 31

Breakout box. 497

Breakpoint frequency. 294

Breakpoint interrupts. 208, 214

Breakpoints. 47, 61, 68–69, 208, 214, 552, 553

Bresenham's algorithm. 457–458

Bridges (gateways) for networks. 526

Broadband bus (tree-structured) networks. 522, 523

Broadband transmission. 518–520

BSF instruction (80386). 563

BSR instruction (80386). 563

BSWAP instruction (80486). 370

BT instruction (80386). 563

BTC instruction (80386). 563

BTR instruction (80386). 563

BTS instruction (80386). 563

BU (bus unit) (80286). 543

Buffers
circular (ring). 515–517
high-power. 302–303
integrated-circuit (IC). 277–278
inverting. 11
noninverting. 11, 293
transistor. 278–280

Builder. 566

Bus interface unit (BIU). 28–31, 581

Bus master. 550

Bus timing. 8086 microprocessor. 584

Bus unit (BU) (80286). 543

Busess
8086. 29, 582
A-bus. 29
activities during read and write operations. 164–167
address. 14, 15, 23–26
B-bus. 29
bidirectional. 24
C-bus. 29
control. 23–26
data. 14, 15, 23–26
extended industry standard architecture (EISA) bus. 550–551
industry standard architecture (ISA) bus. 550
MicroChannel Architecture (MCA) bus. 551–552

BX register. 30

Bypass capacitors. 176, 185

Byte
defined. 1
high. 35
low. 35

Byte-oriented protocol (BCP). 518–520

Byte transfer instructions. 42

Byte type. 34, 55, 158

C-bus, 29

C programming language
actual arguments (parameters) of
functions in. 418
arithmetic operators in. 406
assignment operator in. 406
automatic storage class in. 420
bitwise operators in. 406–407
calling functions in. 417–419
CASE structure in. 411–412
char (character) pointers in. 404–406
char (character) variables in. 396–397
character strings in. 404–406
combined operators in. 407
curly braces in. 390
data types in. 395–396
declaring functions in. 417–419
defining functions in. 417–419
dereferencing pointers in. 400
do-while structure in. 412–413
enumerated data type in. 396

C programming language (*Cont.*)
 extern (global) storage class in. 419-420
 FILE pointers in. 475
 float (floating-point) pointers in. 403-404
 float (floating-point) variables in. 398
 FOR-DO loop in. 413-417
 FOR loop in. 413-417
 formal arguments (parameters) of functions in. 417
 function storage classes in. 419-420
 if-else structure. 410-411
 IF-THEN-ELSE structure in. 410-411
 IF-THEN structure in. 410-411
 index method of accessing array elements. 414-416
 int array (Integer array) pointers in. 400-403
 int (Integer) pointers in. 398-400
 int (Integer) variables in. 397-398
 Integrated Development Environment for. 391-395
 introduced. 395
 keyboard input library functions for. 423-424
 library functions for. 423-426
 logical operators in. 407-408
 math library functions for. 425-426
 memory models for. 428
 operator precedence in. 408-409
 output library functions for. 424
 parentheses in. 390-391
 passing array pointers to functions in. 420-421
 passing parameters by reference. 400
 passing parameters by value. 400
 pointer method of accessing array elements. 416-417
 pointers and functions with two-dimensional arrays. 421-423
 pointers to functions. 423
 preprocessor directives in. 390
 program development tools for. 391-395
 programs. *See* C programs
 prototypes of functions in. 417-418
 register storage class in. 420
 relational operators in. 407
 REPEAT-UNTIL structure in. 412-413
 static storage class in. 420
 string library functions for. 424-425
 switch structure in. 411-412
 union data structure in. 437-439
 variable declarations in. 396-398
 variable storage classes in. 419-420
 variable types in. 396-398
 WHILE-DO structure in. 412-413
 while structure in. 412-413

C programs

8086 assembly language programs with. 426-432
 adding constants to arrays of data. 389-391, 400-404, 420-421
 arithmetic average. 414-417, 421-423
 calling functions. 418-419
 char pointers. 404-406
 char variables. 396-397
 declaring functions. 418-419
 defining functions. 418-419
 disk file operations. 475-477

C programs (*Cont.*)

do-while structure. 412-413
 downloading to SDK-86. 508-518
 float pointers. 403-404
 float variables. 398
 graphics programming. 458-460
 hypotenuse calculation. 425-426
 if-else. 410-411
 index method of accessing array elements. 414-416
 int array pointers. 400-403
 int pointers. 398-400
 int variables. 397-398
 keyboard input. 437-439
 for loops. 414-417
 passing array pointers to functions. 420-421
 pointer method of accessing array elements. 416-417
 pointers and functions with two-dimensional arrays. 421-423
 profit factor adjustment. 389-391, 400-404, 420-421
 switch structure. 412-413
 video programming. 458-460
 while structure. 412-413

Cache memory
 80486. 534, 569
 80586. 571
 80686. 571
 80786. 571
 82385 cache controller for. 358, 360-363
 cache directory for. 360
 direct-mapped. 360-361
 DRAM and. 358, 360, 363
 fully associative. 362-363
 hit rate for. 358, 360
 implementation of. 358, 360
 posted-write-through for. 360
 SRAM and. 358, 360, 363
 summary of. 363
 two-way set associative. 361-362
 virtual memory versus. 541

Caches, disk. 477

CAE (computer-aided engineering). 379

CALL instruction. 43, 100-102, 106-107, 133-134, 591, 593

Call table. 325, 330

Called modem. 501-502

Calling functions in C. 417-419

Calling modem. 501-502

Calls
 direct. 101-102, 133-134
 far (intersegment). 100, 101-102, 107, 133-134
 indirect. 101, 102, 133-134
 near (intra-segment). 100, 101, 106-107, 133

Capacitive keyswitches. 260

Capacitors
 bypass. 176, 185
 filter. 176, 185

Carrier sense, multiple access with collision detection (CSMA/CD). 523, 524-525

Carry, auxiliary. 10

Carry flag (CF). 28, 29, 74-75

Cascaded devices. 235

Case design. 386

CASE structure. 39-41, 81-82, 411-412

Cathode-ray tube (CRT) displays
 - 6845 CRT controller for. 442-443
 8514/A high-resolution graphics board for. 447

Cathode-ray tube (CRT) displays (*Cont.*)

attribute code and. 442-443
 bit-mapped. 444
 character display on. 440-442
 character generator ROM for. 440-442
 color. 444-451
 color graphics adaptor (CGA) for. 447-449
 composite video color monitor. 448
 display refresh RAM for. 440, 441
 dot clock for. 440-441
 enhanced graphics adaptor (EGA) for. 447, 449
 field of. 439
 frame buffer for. 440, 441
 frame rate for. 439-440
 frequencies for. 442-443
 Hercules adaptors for. 447
 high-resolution graphics. 460-461
 horizontal sync pulse for. 440
 interlaced scanning for. 439-440
 monochrome. 440, 442-443, 444, 447
 multicolor graphics array (MCGA) for. 447
 noninterlaced scanning for. 439, 440
 overscan in. 443
 packed pixel storage for. 445, 446
 picture element (pel or pixel) of. 444
 pitch of. 444
 planar pixel storage for. 445-446
 plasma displays. 462
 programming. *See* Video programming
 RAMDAC for. 450
 raster scanning for. 439-440
 terminal. 440
 timing for. 442-443
 vertical sync pulse for. 440
 video DAC for. 449-450
 video graphics array (VGA) for. 447-449-451
 video monitor. 440
 video RAM (VRAM) for. 446-447

CBW instruction. 43, 134, 590, 593

CCD (charge-coupled device) cameras. 464

CD (compact disk). 478

CD-I (compact digital interactive). 483

CDQ instruction (80386). 563

Central processing unit (CPU). *See also* Microprocessors
 buses connected to. 23-26
 decoding of instructions by. 24-26
 defined. 19, 24
 execution of instructions by. 24-26
 fetching of instructions by. 24-26
 general-purpose. 27
 microcomputer. 23-26
 purposes of. 24
 registers in. 24

Centronics parallel interface
 connections for. 252-255
 pin descriptions for. 252-253
 printer driver program for. 254-255
 SDK-86 connections for. 252, 254-255
 timing waveforms for. 254-255

CF (carry flag). 28, 29, 74-75

CGA (color graphics adaptor). 447-449

CH register. 29-30

Char (character) pointers in C. 404-406

Char (character) variables in C. 396-397

- Character generator read-only memory (ROM), 440-442
- Character strings in C, 404-406
- Characters
- affirmative acknowledge (ACK), 519
 - block check (BCC), 518-519
 - control, 6
 - cyclic redundancy (CRC), 468, 469, 518, 519, 521
 - end-of-block (ETB), 518
 - end-of-text (ETX), 518
 - end-of-transmission (EOT), 519
 - enquiry (ENQ), 519
 - negative acknowledge (NAK), 519
 - sentinel method for sending, 255
 - start-of-header (SOH), 518
 - start-of-text (STX), 518
 - sync, 518
 - token, 523
- Charge-coupled device (CCD) cameras, 464
- Check (encoding) bits, 364
- Chip enable inputs, 15
- CIM (computer-integrated manufacturing), 386
- Circular (ring) buffer, 515-517
- CISC (complex instruction set computer) processors, 461
- CL register, 29-30
- Cladding material, 504
- CLC instruction, 44, 134, 591, 593
- CLD instruction, 44, 134, 591, 593
- CLI instruction, 44, 134, 215, 591, 593
- Clock cycles
- 8086 instructions, 592-598
 - 8087 instructions, 599-605
 - delay loops, 91-93, 103-106
- Clock waveform, 165
- Clocks
- nonsystem, with 8254, 231
 - real-time, 220-221, 237-240
 - states of, 165
- Closed-loop gain, 292, 293
- Closing files, 475-477
- Clusters on disks, 473
- CMC instruction, 44, 134, 591, 593
- CMP instruction, 43, 134-135, 590, 593-594
- CMPS/CMPSB/CMPSW instructions, 135, 591, 597-598
- CMPXCHG instruction (80486), 570
- CNC (computer numerical control) machines, 250-252
- Code conversion
- compare technique for, 262-264
 - XLAT method for, 266-267
- Code segment, 58
- Code segment (CS) register, 29, 30-32, 89-90, 581
- Codecs, 502
- Coders, 502
- Codes
- 7-segment display, 4
 - alphanumeric, 4-6
 - analog-to-digital (A/D) output, 306
 - ASCII. See American Standard Code for Information Interchange
 - attribute, 442-443
 - BCD. See Binary-coded decimal code
 - binary, 306
 - bipolar binary, 306
 - digital-to-analog (D/A) input, 302, 306
 - error detecting/correcting (ECCs), 364-365
 - Extended Binary-Coded Decimal In-
- Codes (Cont.)
- terchange Code (EBCDIC), 5, 6, 266-267
 - frequency modulation (FM), 467, 468
 - Gray, 3-4, 283-284, 499
 - Hamming, 364-365
 - linear predictive (LPC), 481
 - Manchester, 524
 - modified frequency modulation (MFM), 467-468
 - nonreturn-to-zero (NRZ), 467
 - run-length-limited (RLL), 468, 472
 - trellis, 500
 - unipolar binary, 306
- Coding sheets for programs, 45-47
- Coding templates, 47-53
- Cold-junction compensation, 298
- Collision of transmissions, 524
- Color cathode-ray tube (CRT) displays, 444-451
- Color graphics adaptor (CGA), 447-449
- Color palettes, 445
- Combined operators in C, 407
- Command words. See Control words
- Comment field, 32
- Comments in programs, 32, 45, 46
- Common-bus networks, 522, 523
- Common-mode rejection, 294
- Common-mode voltage, 294
- Compact digital interactive (CD-I), 483
- Compact disk (CD), 478
- Companders (companding codecs), 502
- Comparators, 291, 292
- Compare instructions, 8087, 374
- Compare technique for code conversion, 262-264
- Compiler program, 33
- Complete decoding, 195
- Complex instruction set computer (CISC) processors, 461
- Composite video color monitor, 448
- COMPS/COMPSB/COMPSW instructions, 43
- Computer-aided engineering (CAE), 379
- Computer-integrated manufacturing (CIM), 386
- Computer numerical control (CNC) machines, 250-252
- Computer vision, 463-465
- Conditional flags, 28-29, 74-76
- Conditional jumps, 71, 76-77, 83-84, 90-91
- Conditional transfer instructions, 44
- Connector flowchart symbol, 38
- Connector symbols, 185
- Constants
- 8087 instructions for, 375
 - named, 54-55
- Constellations (phase-amplitude graphs), 499-500
- Contact bounce and debouncing, 260, 261-265
- Contactors, 280
- Context (environment, or state) of tasks, 537-538
- Context switching, 537
- Control bus, 23-26
- Control characters, 6
- Control circuitry, 8086, 28, 29
- Control flags, 29
- Control words
- 8250, 515, 516
 - 8251A, 490-493, 519, 520
 - 8254, 224-226, 232
 - 8255A, 248-250, 255
 - 8279, 271-274
- Controllers
- dedicated, 27
 - embedded. See Embedded controllers
- Coprocessors. See Math coprocessors
- Counters
- 8253, 221
 - 8254. See 8254 programmable timer/counter
 - address, 24
 - binary, 14
 - flip-flops as, 13-14
 - location, 97-98
 - registers as, 86-89
- Counting, interrupts for, 218-219, 220
- CPU. See Central processing unit
- CRC (cyclic redundancy characters), 468, 469, 518, 519, 521
- Critical angle, 505, 506
- Critical frequency, 294
- Critical region, 517, 538-539
- CRT displays. See Cathode-ray tube displays
- CS (code segment) register, 29, 30-32, 89-90, 581
- CSMA/CD (carrier sense, multiple access with collision detection), 523, 524-525
- CTS instruction (80286), 547
- Current loops, 300-301, 488, 494
- CWD instruction, 43, 135, 590, 594
- CWDE instruction (80386), 563
- CX register, 30
- Cycles (instruction and machine), 165
- Cyclic redundancy characters (CRC), 468, 469, 518, 519, 521
- D bit, 48, 49, 590, 591
- D flip-flop, 13
- D latch, 12-13
- D/A converters. See Digital-to-analog converters
- DAA instruction, 42, 135-136, 590, 594
- Darlington transistors, 278-279
- DAS (data acquisition system), 305, 319, 322-323
- DAS instruction, 43, 136, 590, 594
- Data acquirer system (DAS), 305, 319, 322-323
- Data arrays, 86-89
- Data bases, defined, 21
- Data bus, 14, 15, 23-26
- Data communication equipment (DCE), 489, 494
- Data concentrators (multiplexers), 522
- Data link layer (OSI model), 523, 524
- Data outputs, 14-15
- Data-ready signal, 84-86
- Data segment, 58
- Data segment (DS) register, 29, 30-32, 34-35, 89-90, 581
- Data statements
- binary-coded decimal (BCD) numbers in, 56
 - binary numbers in, 55
 - decimal numbers in, 56
 - hexadecimal numbers in, 56
 - numbers used in, 55-56
- Data storage registers, 13
- Data terminal equipment (DTE), 489, 494
- Data transfer instructions
- 8086, 42, 590
 - 8087, 372
- Data-type conversion instructions (80386), 563

- Data types in C, 395-396
- DB (define byte) directive, 55, 158
- DCE (data communication equipment), 489, 494
- DD (define doubleword) directive, 55, 158
- Deadlock, 538
- Debouncing keyboards, 261-265
- Debug registers (80386), 552, 553
- Debugger program, 60, 61
- Debugging programs
 - 5-minute rule for, 68
 - breakpoints for, 68-69
 - GO command for, 68
 - with procedures, 115-116
 - single-step command for, 68
 - trace data for, 62, 170, 171-172
- DEC instruction, 42, 136, 590, 594
- Decimal adjust operation, 10
- Decimal (base-10) numbers
 - in data statements, 56
 - defined, 1
- Decimal-to-binary conversion, 2
- Decimal-to-hexadecimal conversion, 3
- Decision flowchart symbol, 38
- Decision operations, 39, 40
- Declaring functions in C, 417-419
- Decoder
 - 8086, 28, 29
 - pulse code, 502
- Decoding instructions, 24-26
- Dedicated controllers, 27
- Dedicated interrupts, 208
- Define byte (DB) directive, 55, 158
- Define doubleword (DD) directive, 55, 158
- Define quadword (DQ) directive, 158
- Define ten bytes (DT) directive, 158-159
- Define word (DW) directive, 55, 159
- Defining functions in C, 417-419
- Delay loops, 91-93, 103-106
- Delta (differential) modulation, 482
- Dereferencing pointers, 400
- Derivative feedback, 318-319
- Descramblers, 500
- Descriptor table, 542-543
- Design and development tools. *See* Electronic design automation
- Design rule checker (DRC) program, 380
- Design for test, 384, 386
- Destination index (DI) register, 29, 31
- Destination for instructions, 32-33
- DF (direction flag), 29
- DH register, 29-30
- DI (destination index) register, 29, 31
- Dibits, 499, 500
- Differential (delta) modulation, 482
- Differential operational amplifiers (op amps), 291, 294
- Differential phase-shift keying (DPSK) modulation, 499
- Differential pressure transducer, 300
- Differentiators, 291, 294
- Digital feedback, 285
- Digital filters
 - advantages of, 337
 - block diagram of, 340-341
 - development tools for, 341-342
 - finite impulse response (FIR) algorithm for, 339
 - hardware for, 339-341
 - infinite impulse response (IIR) algorithm for, 339
 - operation of, 338-339
- Digital filters (*Cont.*)
 - principle of, 336-337, 338
 - sampling signals for, 338-339, 340
 - software for, 341
 - switched capacitor, 342
- Digital-to-analog (D/A) converters
 - characteristics and specifications, 301-302
 - full-scale output voltage of, 302
 - input codes for, 302, 306
 - linearity of, 302
 - maximum error of, 302
 - microcomputer interfacing for, 302-304
 - operation of, 301
 - palette, 449-450
 - RAMDAC, 450
 - resolution of, 302
 - settling time for, 302
 - video, 449-450
- Digital video interactive (DVI), 483
- Direct addressing mode, 34-35
- Direct calls, 101-102, 133-134
- Direct input/output (I/O), 189, 193
- Direct jumps, 72-74, 143
- Direct memory access (DMA)
 - 80186 programmable DMA unit, 335, 336
 - controller for. *See* 8237 DMA controller
 - defined, 306
 - overview of, 348
 - principle of, 306
 - slave boards, 550-551
- Direction flag (DF), 29
- Directives. *See* Assembler directives
- Directory of disks, 473
- Dirty bit, 542
- Disk caches, 477
- Dispatcher, 537
- Display refresh random-access memory (RAM), 440, 441
- Displays
 - alphanumeric. *See* Light-emitting diodes; Liquid-crystal displays
 - CRT. *See* Cathode-ray tube displays
 - drivers for, 274-275
 - multiplexed, 267-268, 269
 - static, 267, 268
- Distributed processing systems, 21-23
- DIV instruction, 43, 136-137, 208, 213, 216, 590, 594
- Divide-by-zero interrupts, 208-213, 216
- Division
 - 8086 instructions for, 27, 43
 - 8087 instructions for, 373
 - binary, 9
 - programs for, 122-127
- DL register, 29-30
- DMA. *See* Direct memory access
- Do-while structure in C, 412-413
- Documentation of programs, 47
- DOS
 - 80386 programs for, 565-566
 - basic input output system (BIOS) for, 240-242, 386, 435-439, 451-453, 454, 506-518, 536-537
 - compatibility box for, 567
 - disk cache with, 477
 - file control block (FCB) in, 474
 - file handle (token) in, 474-475
 - function calls in, 474-475
 - Microsoft Windows and, 567-568
 - random-access memory (RAM) disks with, 477
- DOS (*Cont.*)
 - terminate-and-stay-resident (TSR) programs and, 535-537
- Dot-matrix light-emitting diodes (LEDs), 267, 276
- Dot-matrix printers, 479-480
- Double-density recording for disks, 467-468
- Double-handshake input/output (I/O), 246-247
- Double indexed addressing mode, 49, 89-90
- Double-precision numbers, 367-368
- Doubleword (32 bits), 1
- Doubleword type, 55, 158
- DPSK (differential phase-shift keying) modulation, 499
- DQ (define quadword) directive, 158
- DRAM. *See* Dynamic random-access memory
- DRC (design rule checker) program, 380
- Drivers, 255-259
- Drystones, 571
- DS (data segment) register, 29, 30-32, 34-35, 89-90, 581
- DT (define ten bytes) directive, 158-159
- DTE (data terminal equipment), 489, 494
- Dual-ported random-access memory (RAM), 448
- Dual-slope analog-to-digital (A/D) converters, 304-305, 307
- Dummy procedures, 115
- Dummy variables, 128
- DVI (digital video interactive), 483
- DW (define word) directive, 55, 159
- DX register, 30
- Dynamic random-access memory (DRAM)
 - 82C08 controller for, 355-357
 - block diagram of, 353, 354
 - burst-mode refresh, 355
 - cache memory and, 358, 360
 - characteristics of, 353-355
 - column-address strobe (CAS) for, 353-354
 - described, 15-16
 - dynamic-mode refresh, 355
 - error detection and correction for, 363-365
 - Hamming codes and, 364-365
 - hard errors for, 363
 - microcomputer interfacing for, 355-357
 - page mode for, 358, 359
 - parity check for, 363-364
 - precharging, 357
 - refresh controllers for, 16, 354-355
 - row-address strobe (RAS) for, 353, 354
 - soft errors for, 363
 - static column mode for, 358, 359
 - syndrome words and, 364-365
 - timing in microcomputers, 357
 - timing waveforms for, 353-354
- EA (effective address), 34, 49-50, 89-90, 592
- EBCDIC (Extended Binary-Coded Decimal Interchange Code), 5, 6, 266-267
- ECCs (error detecting/correcting codes) 364-365
- EDA. *See* Electronic design automation

- EDAC (error detecting and correcting) device, 364-365
- Editor program, 59-60
- EEPROM (electrically erasable programmable read-only memory), 15
- Effective address (EA), 34, 49-50, 89-90, 592
- EGA (enhanced graphics adaptor), 447, 449
- EISA (extended industry standard architecture) bus, 550-551
- Electrical rule checker (ERC) program, 380
- Electrically erasable programmable read-only memory (EEPROM), 15
- Electromagnetic interference (EMI), 280, 281
- Electronic design automation (EDA) behavioral models for, 382 case design, 386 computer-integrated manufacturing (CIM) and, 386 design overview and, 379 design review committee and, 379 design for test and, 384, 386 gate-level models for, 382 hardware models for, 382-383 initial design, 379-380 introduced, 379 printed-circuit-board design, 386 production and test, 386 prototyping with simulation, 380, 382-385 schematic capture, 379-380, 381 stimulus files and, 383-384 system software and, 386 time steps for simulation, 382
- Electronic mail, 523
- Elements of arrays, 86-89
- Embedded controllers 6801, 27 8048, 27, 265-266 8051 family, 27, 332, 333 8096 family, 27, 332-333, 334 80186 and 80188, 28, 333, 335-336, 534, 598-599 80960 family, 336 microprocessors versus, 332 overview of, 27
- EMI (electromagnetic interference), 280, 281
- Emulators, 59, 61-62
- Enable input, 12-13
- Encoding (check) bits, 364
- END (end program) directive, 59, 159
- End flowchart symbol, 38
- End-of-block (ETB) character, 518
- End-of-text (ETX) character, 518
- End-of-transmission (EOT) character, 519
- End procedure (ENDP) directive, 159
- End program (END) directive, 59, 159
- End segment (ENDS) directive, 53-54, 159
- ENDP (end procedure) directive, 159
- ENDS (end segment) directive, 53-54, 159
- Enhanced graphics adaptor (EGA), 447, 449
- Enhanced small device interface (ESDI) standard, 472
- Enquiry (ENQ) character, 519
- ENTER instruction 80186/80188, 44, 336, 598 80286, 546
- Entry point, 39
- Enumerated data type, 396
- Environment (context, or state) of tasks, 537-538
- EO (erasable optical) disks, 478
- EOT (end-of-transmission) character, 519
- EPROM (erasable programmable read-only memory), 15
- Equate (EQU) directive, 54-55, 159
- Erasable optical (EO) disks, 478
- Erasable programmable read-only memory (EPROM), 15
- ERC (electrical rule checker) program, 380
- Error detecting/correcting codes (ECCs), 364-365
- Error detecting and correcting (EDAC) device, 364-365
- Error trapping, 264
- ES (extra segment) register, 29, 30-32, 89-90, 581
- ESC (escape) instruction, 44, 137, 591, 594
- Escape (ESC) instruction, 44, 137, 591, 594
- ESDI (enhanced small device interface) standard, 472
- ETB (end-of-block) character, 518
- Ethernet, 524-525
- ETX (end-of-text) character, 518
- EU (execution unit) 8086, 28-30, 32, 581 80286, 543
- EVEN (align on even memory address) directive, 159-160
- Even parity, 4, 75
- Excess-3 binary-coded decimal (BCD) code, 4
- Exclusive NOR (XNOR) gate, 11, 12
- Exclusive OR (XOR) gate, 11, 12
- Executing instructions, 24-26
- Execution unit (EU) 8086, 28-30, 32, 581 80286, 543
- Executive programs, 320, 321
- Exit point, 39
- Expanded memory, 540, 541
- Expansion slots, 345
- Expert systems, 572
- Exponent of numbers, 367
- Exponential instructions (8087), 374
- Extended American Standard Code for Information Interchange (ASCII), 435-439
- Extended Binary-Coded Decimal Interchange Code (EBCDIC), 5, 6, 266-267
- Extended industry standard architecture (EISA) bus, 550-551
- Extended (XMS) memory, 541
- Extern (global) storage class in C, 419-420
- External (EXTRN) directive, 122, 160, 162
- External hardware synchronization instructions, 44-45
- Extra segment, 58
- Extra segment (ES) register, 29, 30-32, 89-90, 581
- EXTRN (external) directive, 122, 160, 162
- F2XM1 instruction (8087), 599
- FABS instruction (8087), 374, 599
- Factorials, 117-121
- FADD instruction (8087), 372-373, 599-600
- FADDP instruction (8087), 600
- Far (intersegment) calls, 100, 101-102, 107, 133-134
- Far (intersegment) jumps, 72, 142-143
- Far (intersegment) procedures, 121-127
- Farm connection scheme, 571
- FAT (file allocation table), 473
- FBLD instruction (8087), 372, 600
- FBSTP instruction (8087), 372, 600
- FCB (file control block), 474
- FCFS instruction (8087), 374, 600
- FCLEX instruction (8087), 375, 600
- FCOM instruction (8087), 374, 600
- FCOMP instruction (8087), 374, 600
- FCOMPP instruction (8087), 374, 600
- FCS (frame check sequence), 521
- FDDI (Fiber Distributed Data Interface) standard for networks, 526, 527
- FDECSTP instruction (8087), 375, 600
- FDISI instruction (8087), 375, 600
- FDIV instruction (8087), 373, 600-601
- FDIVP instruction (8087), 373, 601
- FDIVR instruction (8087), 373, 601
- FDIVRP instruction (8087), 601
- Feedback derivative, 318-319 digital, 285 integral, 318, 319 for motors, 302-303 negative, 293, 317 proportional, 318, 319
- FENI instruction (8087), 375, 601
- Fetching instructions, 24-26, 30
- FFREE instruction (8087), 375, 601
- FIADD instruction (8087), 373, 601
- Fiber Distributed Data Interface (FDDI) standard for networks, 526, 527
- Fiber-optic data communication, 503-506
- Fiber-optic local area networks (LANs), 526, 527
- FICOM instruction (8087), 374, 601
- FICOMP instruction (8087), 374, 601
- FIDIV instruction (8087), 373, 601
- FIDIVR instruction (8087), 373, 601
- Field programmable logic array (FPLA), 12
- Fields of frames, 521 for statements, 32
- FILD instruction (8087), 372, 601
- File allocation table (FAT), 473
- File control block (FCB), 474
- File handle (token), 474-475
- File servers, 527-528
- Files closing, 475-477 library, 60 link, 60, 122 list, 57, 60 object, 60, 122 opening, 475-477 path to, 473 source, 60 stimulus, 383-384
- Filter capacitors, 176, 185
- Filters active, 291, 294-295 bandpass, 295 digital. See Digital filters formant, 481-482 high-pass, 291, 294-295 low-pass, 291, 294, 295, 338

- FIMUL instruction (8087), 373, 602
 FINCSTP instruction (8087), 375, 602
 FINIT instruction (8087), 375, 602
 Finite impulse response (FIR) algorithm, 339
 FIR (finite impulse response) algorithm, 339
 Firmware, defined, 24
 FIST instruction (8087), 372, 602
 FISTP instruction (8087), 372, 602
 FISUB instruction (8087), 373, 602
 FISUBR instruction (8087), 602
 Fixed-point numbers, 367
 Fixed-port instructions, 78, 193
 Flag bits, 518
 Flag field of frames, 521
 Flag register, 28–29, 592
 Flag set/clear instructions, 44
 Flag transfer instructions, 42
 Flags
 auxiliary carry (AF), 28, 29, 74, 75
 carry (CF), 28, 29, 74–75
 conditional, 28–29, 74–76
 control, 29
 defined, 28
 direction (DF), 29
 interrupt (IF), 29, 207–209, 213–216
 overflow (OF), 28, 29, 74, 76, 214
 parity (PF), 29, 74, 75
 sign (SF), 28, 29, 74, 75
 trap (TF), 29, 207–209, 213–216
 zero (ZF), 28, 29, 74, 75
 Flash (parallel comparator) analog-to-digital (A/D) converters, 304, 306–307
 Flash electrically programmable read-only memory (flash EPROM), 15
 FLD1 instruction (8087), 375, 602
 FLD2T instruction (8087), 375
 FLD instruction (8087), 372, 602
 FLDCW instruction (8087), 375, 602
 FLDENV instruction (8087), 375, 602
 FLDL2E instruction (8087), 375, 602–603
 FLDLG2 instruction (8087), 375, 603
 FLDLN2 instruction (8087), 375
 FLDPI instruction (8087), 375, 603
 FLDZ instruction (8087), 375, 603
 Flip-flops
 as counters, 13–14
 D flip-flop, 13
 Float (floating-point) pointers in C, 403–404
 Float (floating-point) variables in C, 398
 Floating-point numbers, 367
 Floating point (float) pointers in C, 403–404
 Floating-point processors. *See* Math coprocessors
 Floating-point (float) variables in C, 398
 Floppy disks
 8272A floppy-disk controller for, 469–472
 access times for, 466
 formats for, 468–469
 formatting, 473
 hardware interfacing for, 469–472
 index holes in, 465, 466
 overview of, 465–466
 packages for, 465
 sizes for, 465
 soft-sector, 468–469
 Flow sensors, 300
 Flowcharts
 CASE structure, 40
 Flowcharts (*Cont.*)
 comparing strings, 98
 data sampling, 38, 104
 described, 37
 downloading program, 510
 factorials, 118
 IF-THEN-ELSE structure, 40, 78, 81
 IF-THEN structure, 40
 keyboard input, 261, 262
 microcomputer-based industrial process-control system, 320, 321
 microcomputer-based scale, 308–309
 REPEAT-UNTIL structure, 40, 86, 87
 sequence structure, 40
 strobed input, 86
 symbols for, 37–38
 WHILE-DO structure, 40, 83
 FM (frequency modulation) coding for disks, 467, 468
 FMUL instruction (8087), 373, 603
 FMULP instruction (8087), 373, 603
 FNCLEX instruction (8087), 375, 600
 FNDISI instruction (8087), 375, 600
 FNENI instruction (8087), 375, 601
 FNINT instruction (8087), 375, 602
 FNOP instruction (8087), 375, 603
 FNSAVE instruction (8087), 375, 603
 FNSTCW instruction (8087), 375, 604
 FNSTENV instruction (8087), 375, 604
 FNSTSW instruction (8087), 375, 604
 FOR-DO loop, 41, 90, 413–417
 FOR loop in C, 413–417
 Force transducers, 298–300, 307, 308, 309
 Formal arguments (parameters) of functions, 417
 Formant filters, 481–482
 Forward jump, 74
 Four-phase stepper motors, 281–283
 Fourier series, 337–338
 FPATAN instruction (8087), 374, 599, 603
 FPLA (field programmable logic array), 12
 FPREM instruction (8087), 374, 603
 FPTAN instruction (8087), 374, 599, 603
 Frame check sequence (FCS), 521
 Frames of messages, 520–521
 Frequency-domain description, 337, 338
 Frequency modulation (FM) coding for disks, 467, 468
 Frequency-shift keying (FSK) modulation, 498–499
 Fricatives, 481
 FRNDINT instruction (8087), 374, 603
 FRSTOR instruction (8087), 375, 603
 FSAVE instruction (8087), 375, 603
 FSCALE instruction (8087), 373, 599, 603
 FSK (frequency-shift keying) modulation, 498–499
 FSQRT instruction (8087), 373, 599, 603
 FST instruction (8087), 372, 603–604
 FSTCW instruction (8087), 375, 604
 FSTENV instruction (8087), 375, 604
 FSTP instruction (8087), 372, 604
 FSTSW instruction (8087), 375, 604
 FSUB instruction (8087), 373, 604
 FSUBP instruction (8087), 373, 604
 FSUBR instruction (8087), 373, 604
 FSUBRP instruction (8087), 373, 604–605
 FTST instruction (8087), 374, 605
 Full-duplex communication, 487, 488
 Function storage classes in C, 419–420
 Functions of programs, 47
 Fundamental frequency, 337
 Fusible matrixes, 12
 Fuzzy logic, 574
 FWAIT instruction (8087), 375, 605
 FX2M1 instruction (8087), 374
 FXAM instruction (8087), 374, 605
 FXCH instruction (8087), 372, 605
 EXTRACT instruction (8087), 374, 605
 FYL2X instruction (8087), 374–375, 599, 605
 FYL2XP1 instruction (8087), 375, 599, 605
 Gain-bandwidth product, 293
 Gate-level models, 382
 Gates. *See* Logic gates
 Gateways (bridges) for networks, 526
 General-purpose central processing unit (CPU), 27
 General-purpose registers, 29–30
 Generators
 audio-tone, 230
 ramp, 291, 294
 square-wave, 229–230
 timed interrupt, 228–229
 Gigabyte (unit), 27
 GLOBAL directive, 160
 Global (extern) storage class in C, 419–420
 GO command, 68
 Graphical user interfaces (GUIs), 567–568
 Graphics
 color, 444–447
 high-resolution, 460–461
 monochrome, 444
 Graphics processors, 460–461
 Gray code, 3–4, 283–284, 499
 GROUP directive, 160
 GUIs (graphical user interfaces), 567–568
 Half-duplex communication, 487, 488
 Hall effect keyswitches, 260–261
 Halt (HLT) instruction, 44, 137, 584, 591, 594
 Halt state, 44, 137, 584
 Hamming codes, 364–365
 Hand-coding programs, 35, 45–47, 52, 53, 68
 Hard disks
 access times for, 467
 backup storage for, 477
 cylinders of, 466–467
 enhanced small device interface (ESDI) standard for, 472
 formatting, 474
 hardware interfacing for, 472–473
 interface software for, 475–477
 interleave factor for, 472, 474
 logical drives for, 474
 overview of, 466–467
 parking zone for, 467
 partitioning, 474
 small computer systems interface (SCSI) standard for, 472–473
 ST-506 standard for, 472
 Winchester, 467
 Hard errors, 363
 Hardware, defined, 24

- Hardware interrupts, 44, 134, 156, 168, 207, 208, 213–216, 379, 583–584
- Hardware models, 382–383
- Hardware-triggered strobes, 230–231
- Hardwired matrixes, 12
- Harmonic frequencies, 337
- Harvard architecture, 340, 571
- HDLC (high-level data link control) protocol, 520–522
- Head pointer, 515–517
- Heap area of memory, 458, 460
- Hercules display adaptor, 447
- Hexadecimal (base-16) numbers
 - in data statements, 56
 - defined, 3, 9–10
- Hierarchical charts, 100
- High byte, 35
- High-level data link control (HDLC) protocol, 520–522
- High-level language interface instructions (80186/80188), 44
- High-level languages, 33
- High-pass filters, 291, 294–295
- High Performance File System (HPFS), 567
- High-power buffers, 302–303
- High-resolution graphics, 460–461
- HLT (halt) instruction, 44, 137, 584, 591, 594
- House, David, 570
- HPFS (High Performance File System), 567
- Hypercube topology, 571–572
- Hysteresis, 292

- ICs. *See* Integrated circuits
- IDIV instruction, 43, 137–138, 208, 213, 216, 590, 594
- IF (interrupt flag), 29, 207–209, 213–216
- if-else structure in C, 410–411
- IF-THEN-ELSE structure, 39, 40, 77–82, 410–411
- IF-THEN structure, 39, 40, 77, 410–411
- IGBTs (isolated-gate bipolar transistors), 279–280
- IIR (infinite impulse response) algorithm, 339
- ILDs (infrared injection laser diodes), 503–504
- Immediate addressing mode, 33
- Impact printers, 479
- Impedance, input, 293
- IMUL instruction
 - 8086, 590, 594
 - 80186/80188, 43, 138–139, 336, 598
 - 80286, 546
- IN instruction, 42, 52, 78, 139, 193, 590, 594
- In-line code, 127
- INC instruction, 42, 139, 590, 594
- INCLUDE directive, 160
- Incremental shaft encoders, 284–285
- Index field of disks, 468, 469
- Index holes in disks, 465, 466
- Index of refraction, 505, 506
- Index registers, 29, 32
- Index scaling, 563
- Indexes of arrays, 89
- Indirect calls, 101, 102, 133–134
- Indirect jumps, 72, 143
- Inductive kick, 279

- Industrial process control. *See also* Microcomputer-based industrial process control system
 - data acquisition system (DAS) for, 305, 319, 322–323
 - overview of, 317–320
 - proportional integral derivative (PID) control loops and, 319, 320, 321
 - residual error and, 319
 - servo control and, 317–318
 - set points and, 317–318
- Industry standard architecture (ISA) bus, 550
- Infinite impulse response (IIR) algorithm, 339
- Infrared injection laser diodes (ILDs), 503–504
- Infrared light-emitting diodes (LEDs), 219, 220
- Initialization
 - instructions for, 45, 66, 67, 209
 - of programmable peripheral devices, 223–224, 235–240
 - of segment registers, 58–59
- Ink-jet printers, 480–481
- Input flowchart symbol, 38
- Input impedance, 293
- Input/output (I/O)
 - 8086 addressing of, 582
 - Basic Input Output System (BIOS) for, 240–242, 386, 435–439, 451–453, 454, 506, 518, 536–537
 - direct, 189, 193
 - double-handshake, 246–247
 - drivers for, 255
 - interrupt, 216–219
 - memory-mapped, 189, 193
 - microcomputer, 23–26
 - polled, 216–217, 218
 - ports for. *See* Ports
 - read signal for, 24–26
 - simple, 245, 246
 - simple strobe, 245–246
 - single-handshake, 246
 - write signal for, 24–26
- Input ports
 - described, 23–26
 - IN instruction for, 42, 52, 78, 139, 193, 590, 594
 - program for reading, 45–47
- INS/INSB/INSW instructions (80186/80188), 43, 599
- INS instruction (80286), 546
- Instruction cycle, 165
- Instruction pipelining, 569, 571
- Instruction pointer (IP) register, 24, 29, 30–31
- Instruction unit (IU) (80286), 543
- Instructions
 - decoding, 24–26
 - executing, 24–26
 - fetching, 24–26, 30
 - overhead, 92
 - pipelined, 30, 569, 571
- Instrument prototyping, 331–332
- Instrumentation operational amplifiers (op amps), 291, 294
- Int array (integer array) pointers in C, 400–403
- INT instruction, 44, 52, 139–140, 207, 214, 216, 240–242, 591, 594
- Int (integer) pointers in C, 398–400
- Int (integer) variables in C, 397–398
- Integer array (int array) pointers in C, 400–403

- Integer (int) pointers in C, 398–400
- Integer transfers (8087), 372
- Integer (int) variables in C, 397–398
- Integral feedback, 318, 319
- Integrated circuits (ICs)
 - buffers using, 277–278
 - handling, 201–202
 - RAMDACs, 450
 - on schematic diagrams, 185
 - troubleshooting, 201–202
- Integrated Development Environment for C, 391–395
- Integrated services digital network (ISDN), 503, 504
- Integrators, 291, 294
- Interlaced scanning, 439–440
- Interleave factor, 472, 474
- Internal addresses, 224
- International Standards Organization (ISO)
 - high-level data link control (HDLC) protocol, 520–522
 - open systems interconnection (OSI) model, 523–524
- Interpreter program, 33
- Interrupt flag (IF), 29, 207–209, 213–216
- Interrupt input/output (I/O), 216–219
- Interrupt-pointer (interrupt-vector) table, 208
- Interrupt-service procedures, 116–117, 168, 207–212, 217–218, 219
- Interrupts
 - 8086 response to, 207–208
 - 80286, 544, 545
 - available, 208
 - Basic Input Output System (BIOS) procedure calls with, 240–242
 - breakpoint, 208, 214
 - CLI instruction for, 44, 134, 215, 591, 593
 - for counting, 218–219, 220
 - dedicated, 208
 - divide-by-zero, 208–213, 216
 - hardware, 44, 134, 156, 168, 207, 208, 213–216, 379, 583–584
 - INT instruction for, 44, 52, 139–140, 207, 214, 216, 240–242, 591, 594
 - INTO instruction for, 44, 140, 214, 216, 591, 594
 - IRET instruction for, 44, 140, 209, 216, 591, 594
 - maskable (INTR), 44, 134, 156, 168, 207, 215–216, 583–584
 - nonmaskable (NMI), 134, 168, 207, 208, 213–214, 216, 379, 583
 - overflow, 208, 214, 216
 - PIC for. *See* 8259A priority interrupt controller
 - priority of, 216
 - for real-time clocks, 220–221
 - reserved, 208
 - single-step, 208, 213, 216
 - software, 44, 52, 139–140, 207, 214, 216, 240–242, 591, 594
 - STI instruction for, 44, 156, 215, 591, 597
 - timed interrupt generators, 228–229
 - for timing, 219–221
- Intersegment (far) calls, 100, 101–102, 107, 133–134
- Intersegment (far) jumps, 72, 142–143
- Intersegment (far) procedures, 121–127
- INTO instruction, 44, 140, 214, 216, 591, 594

- INTR (maskable) interrupts, 44, 134, 156, 168, 207, 215-216, 583-584
- Intrasegment (near) calls, 100, 101, 106-107, 133
- Intrasegment (near) jumps, 72-74, 142-143
- Intrasegment (near) procedures, 103-106
- INVD instruction (80486), 570
- Inverters, 11
- Inverting buffers, 11
- Inverting operational amplifiers (op amps), 291, 293
- INVLPG instruction (80486), 570
- I/O. *See* Input/output
- IP (instruction pointer) register, 24, 29, 30-31
- IRET instruction, 44, 140, 209, 216, 591, 594
- ISA (industry standard architecture) bus, 550
- ISDN (integrated services digital network), 503, 504
- ISO. *See* International Standards Organization
- Isolated-gate bipolar transistors (IGBTs), 279-280
- Iteration control instructions, 44
- Iteration operations, 39-41
- IU (instruction unit) (80286), 543
- JA instruction, 44, 76, 140, 591, 594-595
- Jack (J) symbols, 185
- JAE instruction, 44, 76-77, 140, 591, 594-595
- JB instruction, 44, 76, 141, 591, 594-595
- JBE instruction, 44, 76, 141, 591, 594-595
- JC instruction, 44, 76, 141, 594-595
- JCXZ instruction, 44, 91, 141, 591, 594-595
- JE instruction, 44, 76, 141, 591, 594-595
- JG instruction, 44, 76, 141-142, 591, 594-595
- JGE instruction, 44, 76, 142, 591, 594-595
- JL instruction, 44, 76, 142, 591, 594-595
- JLE instruction, 44, 76, 142, 591, 594-595
- JMP instruction, 43, 72-74, 142-143, 591, 595
- JNA instruction, 44, 76, 141, 591, 594-595
- JNAE instruction, 44, 76, 141, 591, 594-595
- JNB instruction, 44, 76, 140, 591, 594-595
- JNBE instruction, 44, 76, 140, 591, 594-595
- JNC instruction, 44, 76, 140, 594-595
- JNE instruction, 44, 76, 143, 591, 594-595
- JNG instruction, 44, 76, 142, 591, 594-595
- JNGE instruction, 44, 76, 142, 591, 594-595
- JNL instruction, 44, 76, 142, 591, 594-595
- JNLE instruction, 44, 76, 141-142, 591, 594-595
- JNO instruction, 44, 76, 143, 591, 594-595
- JNP instruction, 44, 76, 143, 591, 594-595
- JNS instruction, 44, 76, 143, 591, 594-595
- JNZ instruction, 44, 76, 143, 591, 594-595
- JO instruction, 44, 76, 143-144, 591, 594-595
- JP instruction, 44, 76, 144, 591, 594-595
- JPE instruction, 44, 76, 144, 591, 594-595
- JPO instruction, 44, 76, 143, 591, 594-595
- JS instruction, 44, 76, 144, 591, 594-595
- Jukebox optical disk systems, 479
- Jump table, 82
- Jumps
 - backward, 72-74
 - conditional, 71, 76-77, 83-84, 90-91
 - direct, 72-74, 143
 - far (intersegment), 72, 142-143
 - forward, 74
 - indirect, 72, 143
 - near (intrasegment), 72-74, 142-143
 - short, 72-74, 76, 83-84
 - unconditional, 71, 72-74
- JZ instruction, 44, 76, 141, 591, 594-595
- Kbyte (kilobyte), 30
- Keyboard input, 84-86, 217-219, 237-240, 262-264, 423-424, 435-439
- Keyboards
 - circuit connections, 261-262
 - compare code conversion technique for, 262-264
 - debouncing, 261-265
 - dedicated microprocessor encoders for, 265-266
 - detecting keypress on, 261-265
 - EBCDIC to ASCII conversion for, 266-267
 - encoding keypress on, 261-265
 - hardware interfacing for, 264-267
 - keyswitch types for, 260-261
 - N-key rollover for, 273
 - software interfacing for, 262-264
 - two-key lockout for, 262, 273
 - two-key rollover for, 265, 273
 - XLAT code conversion technique for, 266-267
- Keypad interfacing, 268-274
- Kilobyte (Kbyte), 30
- Kosko, Bart, 574
- LABEL directive, 160
- Label field, 32
- Labels in programs, 45, 46, 58
- LAHF instruction, 42, 144, 590, 595
- LANs. *See* Local area networks
- LAR instruction (80286), 547
- Laser printers, 480
- Latches
 - D latch, 12-13
 - defined, 12
- Latency time for disks, 466
- Lathe, 250-252
- LCDs. *See* Liquid-crystal displays
- LDS instruction, 42, 144, 590, 595
- LEA instruction, 42, 144-145, 590, 595
- Least significant bit (LSB), 1
- Least significant digit (LSD), 2, 3
- LEAVE instruction
 - 80186/80188, 44, 336, 598
 - 80286, 547
- LEDs. *See* Light-emitting diodes
- LENGTH operator, 161
- LFS instruction, 42, 145, 590, 595
- LFS instruction (80386), 563-564
- LGDT instruction (80286), 547
- LGS instruction (80386), 564
- Library file, 60
- LIDT instruction (80286), 547
- Light-emitting diodes (LEDs)
 - 7-segment, 4, 267-276
 - 7-segment display code for, 4
 - 18-segment, 267, 276
 - 8279 controller for. *See* 8279 dedicated display controller described, 267
 - directly driven (static), 267, 268
 - dot-matrix, 267, 276
 - infrared, 219, 220
 - in optical couplers, 228
 - software-multiplexed, 267-268, 269
- Light sensors, 295-296
- LIM/EMS (Lotus-Intel-Microsoft Expanded Memory Standard), 540
- Linear predictive coding (LPC), 481
- Linear ramp, 294
- Linear variable differential transformers (LVDTs), 299-300
- Link file, 60, 122
- Link map, 60
- Linker program, 60
- Liquid-crystal displays (LCDs)
 - backplane drive of, 276
 - described, 267
 - dynamic scattering type, 276
 - field-effect type, 276
 - microcomputer interfacing of, 276-277
 - operation of, 276
 - reflective-type, 462
 - screen-type, 461-462
 - transmission-type, 462
- List file, 57, 60
- LLDT instruction (80286), 547
- LMSW instruction (80286), 547
- Load cells, 299, 300, 307, 308, 309
- Local area networks (LANs)
 - 10BaseT (thin Ethernet), 525
 - application example of, 526-529
 - backbones for, 526, 527
 - bridges (gateways) for, 526
 - distributed processing systems and, 21-23
 - Ethernet, 524-525
 - Fiber Distributed Data Interface (FDDI) standard for, 526, 527
 - fiber-optic, 526, 527
 - file server for, 527-528
 - overview of, 522
 - print server for, 527-528
 - protocols for, 522, 523-524
 - software overview for, 528-529
 - topologies for, 522-523
- Location counters, 97-98
- Locator program, 60
- LOCK instruction, 45, 145, 584, 591, 595
- LODS/LODSB/LODSW instructions, 43, 145, 591, 597-598

- Logarithmic instructions (8087), 374–375
- Logic, fuzzy, 574
- Logic analyzers
 - block diagram of, 169
 - clock qualifier for, 171–172
 - display formats of, 170
 - external clock for, 169–170
 - internal clock for, 169–170
 - memory access time measurement with, 172
 - operation of, 169–171
 - overview of, 168–169
 - trace data with, 62, 170, 171–172
 - triggering of, 169–171
 - troubleshooting perspective for, 202–203
 - word recognizer for, 169, 171
- Logic arrays, 12
- Logic gates
 - AND gate, 11
 - exclusive NOR (XNOR) gate, 11, 12
 - exclusive OR (XOR) gate, 11, 12
 - NAND gate, 11
 - NOR gate, 11
 - OR gate, 11
- Logical addresses, 541–543
- Logical drives, 474
- Logical instructions, 43
- Logical operators in C, 407–408
- Logical segment, 53–54, 58
- LOOP instruction, 44, 90–91, 145–146, 591, 595
- Loop networks, 522–523
- LOOPE instruction, 44, 91, 146, 591, 595
- LOOPNE instruction, 44, 91, 146, 591, 595
- LOOPNZ instruction, 44, 91, 146, 591, 595
- Loops
 - delay, 91–93, 103–106
 - FOR-DO, 41, 90
 - structure for, 40, 41, 90–91
- LOOPZ instruction, 44, 91, 146, 591, 595
- Lotus-Intel-Microsoft Expanded Memory Standard (LIM/EMS), 540
- Low byte, 35
- Low-pass filters, 291, 294, 295, 338
- LPC (linear predictive coding), 481
- LSB (least significant bit), 1
- LSD (least significant digit), 2, 3
- LSL instruction (80286), 547
- LSS instruction (80386), 564
- LTR instruction (80286), 547
- LVDTs (linear variable differential transformers), 299–300

- Machine cycle, 165
- Machine language, 32
- Machines, computer numerical control (CNC), 250–252
- Macros. *See* Assembler macros
- Magnetic disks
 - access times for, 466, 467
 - address marks on, 468–469
 - boot record of, 473
 - clusters on, 473
 - cyclic redundancy characters (CRC) on, 468, 469
 - data bit formats for, 467–468
 - date field of, 468, 469
 - directory of, 473
 - DOS function calls for, 474–475
- Magnetic disks (*Cont.*)
 - double-density recording for, 467–468
 - error detection for, 468–469
 - file allocation table (FAT) for, 473
 - floppy. *See* Floppy disks
 - frequency modulation (FM) coding for, 467, 468
 - hard. *See* Hard disks
 - ID fields of, 468, 469
 - index field of, 468, 469
 - latency time for, 466
 - modified frequency modulation (MFM) coding for, 467–468
 - nonreturn-to-zero (NRZ) coding for, 467
 - run-length-limited (RLL) coding for, 468, 472
 - seek time for, 466
 - single-density recording for, 467, 468
 - tracks of, 465, 468–469
 - types of, 465
- Magnetic tapes, 465
- Magneto-optical (MO) recording, 478
- Mail, electronic, 523
- Mainframes, 19
- Mainline programs, 320, 321
- Manchester code, 524
- Mantissa (significant) of numbers, 367
- Marking state, 488
- Maskable (INTR) interrupts, 44, 134, 156, 168, 207, 215–216, 583–584
- Masking bits, 69–70
- Master device, 235
- Math coprocessors
 - 8087. *See* 8087 math coprocessor
 - 80287, 544
 - 80387, 569
 - 80486 built-in, 534, 568–569
 - 80786 built-in, 571
 - defined, 365
- Math library functions in C, 425–426
- MAU (multistation access unit), 525, 526
- Maximum mode, 168, 346, 348, 581–582, 588, 589
- Mbyte (megabyte), 30
- MCA (MicroChannel Architecture) bus, 551–552
- MCGA (multicolor graphics array), 447
- Mechanical keyswitches, 260
- Mechanical relays, 280
- Megabyte (Mbyte), 30
- MEGAFLOPS (million floating-point operations per second), 461, 571–572
- Membrane keyswitches, 260
- Memory
 - 8086 organization of, 189–191, 581
 - access time measurements, 172
 - bank-switched, 539–540
 - blocks of, 255
 - cache. *See* Cache memory
 - DMA for. *See* Direct memory access
 - expanded, 540, 541
 - extended (XMS), 541
 - heap area of, 458, 460
 - Lotus-Intel-Microsoft Expanded Memory Standard (LIM/EMS), 540
 - microcomputer use of, 23–26
 - multiuser/multitasking operating system management of, 539–543
 - named, 109, 111
- Memory (*Cont.*)
 - nonvolatile, 14
 - overlay area of, 539
 - purposes of, 23
 - RAM. *See* Random-access memory
 - read signal for, 24–26
 - ROM. *See* Read-only memory
 - segmentation of, 30–32, 34–35
 - types of, 23
 - virtual, 541–543
 - volatile, 15
 - write signal for, 24–26
- Memory-management unit (MMU), 539, 541–543
- Memory-mapped input/output (I/O), 189, 193
- Memory models for C, 428
- Metal-oxide-semiconductor field-effect transistors (MOSFETs), 279–280
- MFLOPS (million floating-point operations per second), 461, 571–572
- MFM (modified frequency modulation) coding for disks, 467–468
- MicroChannel Architecture (MCA) bus, 551–552
- Microcode, 27
- Microcomputer-based industrial process-control system
 - 8086 assembly language program for, 323–331
 - block diagram of, 319–320
 - flowchart for, 320, 321
 - hardware for, 320, 322–323
 - overview of, 320, 321
- Microcomputer-based instrument prototyping, 331–332
- Microcomputer-based scale
 - 8086 assembly language programs for, 309–317
 - algorithm for, 308–309
 - flowchart for, 308–309
 - input circuitry for, 308, 309
 - overview of, 307–308
- Microcomputer-controlled lathe, 250–252
- Microcomputer development system, 33, 59
- Microcomputers
 - 8086-based. *See* 8086-based microcomputers; SDK-86
 - address decoders for, 185–186
 - block diagram of, 23
 - buses of, 23–26
 - CPU of, 23–26
 - evolution of, 570–571
 - I/O section of, 23, 26
 - introduced, 19, 21
 - memory section of, 23
 - motherboards for, 345, 346, 347
 - port decoders on, 188–189, 193–195
 - random-access memory (RAM) address decoding on, 187–188, 192–193
 - read-only memory (ROM) address decoding on, 186–187, 191–192
 - three-step program for, 24–26
 - troubleshooting, 200–204
- Microcontrollers. *See also* Embedded controllers
 - overview of, 27
- Microprocessors. *See also* Central processing unit
 - 4-bit, 26, 27
 - 8-bit, 26–27
 - 16-bit, 27–28

- Microprocessors (Cont.)**
 32-bit, 27
 ALU categorization of, 26
 complex instruction set computer (CISC), 461
 defined, 19
 embedded controllers versus, 332
 evolution of, 26–27, 570–571
 reduced instruction set computer (RISC), 461, 571
 Scalable Processor Architecture (SPARC), 571
- Microsoft Windows**, 567–568
- Microsleeping**, 283
- Million floating-point operations per second (MEGAFLOPS or MFLOPS)**, 461, 571–572
- Minicomputers**, 19, 20
- Minimum mode**, 168, 197–200, 346, 348, 581–582, 586, 587, 589
- Mixed-mode simulators**, 383
- Mixers**, 291, 293–294
- MMU (memory-management unit)**, 539, 541–543
- Mnemonics for instructions**, 32, 45, 46
- MO (magneto-optical) recording**, 478
- MOD bit patterns**, 48–50, 590, 591, 592
- Mode words**. *See* Control words
- Modems**
 amplitude modulation (AM) for, 498, 499
 answer, 501
 called, 501–502
 calling, 501–502
 defined, 488–489
 frequency-shift keying (FSK) modulation for, 498–499
 handshake sequence for, 501–502
 hardware overview of, 500–501
 high-speed transmission problems with, 500
 introduction to, 498
 null, 496
 originate, 501
 phase-shift keying (PSK) modulation for, 499–500
 RS-232C connections for, 494–497
 XMODEM protocol for, 519
- Modes of fibers**, 506
- Modified frequency modulation (MFM) coding for disks**, 467–468
- Modulator-demodulators**. *See* Modems
- Modules of programs**, 39, 60, 122
- Monitor program**, 47, 61
- Monochrome cathode-ray tube (CRT) displays**, 440, 442–443, 444, 447
- MOSFETs (Metal-oxide-semiconductor field-effect transistors)**, 279–280
- Most significant bit (MSB)**, 1, 2, 28, 29
- Most significant digit (MSD)**, 2, 3
- Motherboards**, 345, 346, 347
- Motors**
 absolute shaft encoders for, 284–285
 digital-to-analog (D/A) converters and, 302–303
 drivers for, 279–280
 feedback for, 302–303
 incremental shaft encoders for, 284–285
 optical shaft encoders for, 283–285
 servo control of, 317–318
 stepper, 281–283
- Mouse devices**, 462–463
- MOV instruction**, 42, 48–53, 146–147, 590, 595
- Move and expand instructions (80386)**, 564
- MOVS/MOVS/MOVSW instructions**, 43, 147, 591, 597–598
- MOVSW instruction (80386)**, 564
- MOVZX instruction (80386)**, 564
- MSB (most significant bit)**, 1, 2, 28, 29
- MSD (most significant digit)**, 2, 3
- MUL instruction**, 43, 147–148, 590, 595
- Multicolor graphics array (MCGA)**, 447
- Multilevel simulators**, 383
- Multimode fibers**, 506
- Multiple Virtual DOS Machines (MVDM)**, 567
- Multiplexed displays**, 267–268, 269
- Multiplexers (data concentrators)**, 522
- Multiplication**
 8086 instructions for, 27, 43
 8087 instructions for, 373
 binary, 8–9
 programs for, 53–59
- Multiprocessing systems**, 21–23
- Multistation access unit (MAU)**, 525, 526
- Multitasking systems**, 20–22, 35
- Multiuser/multitasking operating systems**
 accessing resources with, 538
 defined, 535
 environment preservation for, 537–538
 layers of, 539
 memory management for, 539–543
 protection in, 538–539
 scheduling for, 535–537
 tasks of, 535
- Mutual exclusion of tasks**, 538
- MVDM (Multiple Virtual DOS Machines)**, 567
- N-key rollover**, 273
- NAK (negative acknowledge) character**, 519
- NAME directive**, 161
- Named addresses**, 58
- Named constants**, 54–55
- Named memory**, 109, 111
- Named variables**, 55–58
- NAND gate**, 11
- National PACE microprocessors**, 27
- Near (Intrasegment) calls**, 100, 101, 106–107, 133
- Near (intrasegment) jumps**, 72–74, 142–143
- Near (intrasegment) procedures**, 103–106
- NEG instruction**, 43, 148, 590, 595–596
- Negative acknowledge (NAK) character**, 519
- Negative feedback**, 293, 317
- Nested procedures**, 99–100
- Netlist (wiring list) program**, 380
- Network layer (OSI model)**, 523, 524
- Networks**
 10BaseT (thin Ethernet), 525
 application example of, 526–529
 broadband bus (tree-structured), 522, 523
 common-bus, 522, 523
 Ethernet, 524–525
 integrated services digital network (ISDN), 503, 504
 LANs. *See* Local area networks
- Networks (Cont.)**
 loop, 522–523
 neural, 572–574
 ring, 522, 523
 software overview for, 528–529
 star, 522
 token-passing ring, 522, 523, 525–526, 527
 topologies for, 522–523
- Neural networks**, 572–574
- Neuron model**, 572
- Nibble (4 bits)**, 1
- NMI (nonmaskable) interrupts**, 134, 168, 207, 208, 213–214, 216, 379, 583
- No operation (NOP) instruction**, 45, 148, 596, 599
- Noninterlaced scanning**, 439, 440
- Noninterruptible power supply (NPS)**, 357
- Noninverting buffers**, 11, 293
- Noninverting operational amplifiers (op amps)**, 291, 292–293
- Nonmaskable (NMI) interrupts**, 134, 168, 207, 208, 213–214, 216, 379, 583
- Nonreentrant procedures**, 111
- Nonreturn-to-zero (NRZ) coding**, 467
- Nonvolatile memory**, 14
- NOP (no operation) instruction**, 45, 148, 596, 599
- NOR gate**, 11
- Normalizing numbers**, 367
- NOT instruction**, 43, 148, 596
- NPS (noninterruptible power supply)**, 357
- NRZ (nonreturn-to-zero) coding**, 467
- Null modems**, 496
- Number systems**. *See* Binary numbers; Decimal numbers; Hexadecimal numbers
- Object file**, 60, 122
- Odd parity**, 4, 75
- OF (overflow flag)**, 28, 29, 74, 76, 214
- Off-page connector flowchart symbol**, 38
- OFFSET operator**, 161
- Offsets (displacements) of addresses**, 31, 34, 88–90, 542
- On-off (bang-bang) control**, 331
- One-shots**, 227–228
- Onionskin diagram**, 539
- Op amps**. *See* Operational amplifiers
- Opcode field**, 32
- Opcodes (operation codes)**, 32, 45, 46, 48, 590–598
- Open-loop gain**, 292, 293
- Open systems interconnection (OSI) model**, 523–524
- Opening files**, 475–477
- Operand field**, 32
- Operands**, 32, 45, 46
- Operating systems**
 80386 protected mode, 566
 Basic Input Output System (BIOS) of, 240–242, 386, 435–439, 451–453, 454, 506–518, 536–537
- DOS**. *See* DOS
- multiuser/multitasking**. *See* Multiuser/multitasking operating systems
- OS/2**, 566–567
- Operation codes (opcodes)**, 32, 45, 46, 48, 590–598

- Operation flowchart symbol, 38
- Operational amplifiers (op amps)
 - as active filters, 291, 294–295
 - as adders (mixers), 291, 293–294
 - characteristics of, 290, 292
 - as comparators, 291, 292
 - differential, 291, 294
 - as differentiators, 291, 294
 - instrumentation, 291, 294
 - as integrators (ramp generators), 291, 294
 - inverting, 291, 293
 - neuron model using, 572
 - noninverting, 291, 292–293
 - saturation of, 294
 - voltage gain of, 290, 292
- Operator precedence in C, 408–409
- Optical couplers, 228
- Optical disks, 478–479
- Optical motor shaft encoders, 283–285
- Optical read-only memory (OROM), 478
- Optical scanners, 464
- OR gate, 11
- OR instruction, 43, 70–71, 148–149, 591, 596
- OR matrices, 12
- Originate (ORG) directive, 161
- Originate modem, 501
- OROM (optical read-only memory), 478
- OS/2 operating system, 566–567
- OSI (open systems interconnection) model, 523–524
- OUT instruction, 42, 78, 149, 193, 590, 596
- Output flowchart symbol, 38
- Output library functions in C, 424
- Output ports
 - described, 23–26
 - OUT instruction for, 42, 78, 149, 193, 590, 596
- OUTS instruction (80286), 546
- OUTSW instruction (80186/80188), 43, 336, 599
- Overdamped response, 318
- Overflow
 - numeric, 3, 76
 - stack, 113–115
- Overflow flag (OF), 28, 29, 74, 76, 214
- Overflow interrupts, 208, 214, 216
- Overhead, 92
- Overlays, 539
- Overscan, 443
- Overshoot, 317–318
- PARC microprocessors, 27
- Parity
 - binary-coded decimal (BCD)
 - defined, 69–71, 109–115, 366, 372
 - checked pixel storage, 445, 446
 - packets of data, 523
 - Paddle wheels, 300
 - Page printers, 480
 - PAL (programmable array logic), 12
 - Palette digital-to-analog (D/A) converters, 449–450
 - Palettes, 445
 - Paper tape readers, 250–252
 - Parallel comparator (flash) analog-to-digital (A/D) converters, 304, 306–307
 - Parallel data transfer. *See also* 8255A
 - programmable parallel port:
 - Centronics parallel interface
 - double-handshake input/output (I/O), 246–247
 - simple input/output (I/O), 245, 246
 - Parallel data transfer (Cont.)
 - simple strobe input/output (I/O), 245–246
 - single-handshake input/output (I/O), 246
 - Parallel processing, 571–572
 - Parameter passing
 - by reference, 400
 - by value, 400
 - in named memory, 109, 111
 - in registers, 109, 110
 - summary of, 115
 - to macros, 12^e
 - using pointers, 111–113
 - using the stack, 113–115
 - Parameters
 - actual arguments (parameters) of
 - functions, 418
 - defined, 108
 - formal arguments (parameters) of
 - functions, 417
 - Parity
 - defined, 4
 - even, 4, 75
 - odd, 4, 75
 - Parity bit, 4, 363–364
 - Parity flag (PF), 28, 29, 74, 75
 - Parking zone for hard disks, 467
 - Partitioning hard disks, 474
 - Passing parameters. *See* Parameter passing
 - Path to files, 473
 - PCL (printer control language), 480
 - PCM (pulse-code modulation), 502
 - Pel (picture element or pixel), 444
 - PF (parity flag), 28, 29, 74, 75
 - Phase-shift keying (PSK) modulation, 499–500
 - Phoneme synthesis, 482
 - Photocells, 295–296
 - Photodiodes, 295–296
 - Photoresistors, 295–296
 - Phototransistors, 219, 220, 228
 - Physical addresses, 31–32, 34, 49, 89–90, 541–543
 - Physical layer (OSI model), 523, 524
 - Physical segment, 58
 - PIC. *See* 8259A priority interrupt controller
 - Picture element (pel or pixel), 444
 - PID (proportional integral derivative) control loops, 319, 320, 321
 - Pipelined addresses, 548–549
 - Pipelined instructions, 30, 569, 571
 - Pitch
 - of displays, 444
 - of sounds, 481
 - Pixel (picture element or pel), 444
 - PLA (programmable logic array), 12
 - Planar pixel storage, 445–446
 - Plasma displays, 462
 - Plug (P) symbols, 185
 - PM (Presentation Manager), 567
 - Pointer (PTR) directive, 161
 - Pointers
 - char (character), 404–406
 - dereferencing, 400
 - float (floating-point), 403–404
 - head, 515–517
 - int (integer), 398–400
 - int array (Integer array), 400–403
 - Interrupt-pointer table, 208
 - passing parameters using, 111–113
 - registers as, 86–89
 - tail, 515–517
 - to functions, 423
 - Polled input/output (I/O), 216–217, 218
 - POP instruction, 42, 107–108, 149, 590, 596
 - POPA instruction
 - 80186/80188, 42, 336, 598
 - 80286, 546
 - POPF instruction, 42, 149, 590, 596
 - Ports
 - addressing and decoding, 174, 182, 193–195
 - decoders for, 188–189, 193–195
 - described, 23–26
 - fixed-port instructions for, 78, 193
 - IN instruction for, 42, 52, 78, 139, 193, 590, 594
 - OUT instruction for, 42, 78, 149, 193, 590, 596
 - variable-port instructions for, 78, 193
 - Postfix operations, 409
 - Powers of 2, 2
 - Precedence of operators in C, 408–409
 - Precision (accuracy) of numbers, 367
 - Preemptive priority-based scheduling, 537
 - Prefix operations, 409
 - Preprocessor directives, 390
 - Presentation layer (OSI model), 523–524
 - Presentation Manager (PM), 567
 - Pressure transducers, 299–300
 - Primary station, 521
 - Print servers, 527–528
 - Printed-circuit-board design, 386
 - Printer control language (PCL), 480
 - Printer drivers, 255–259
 - Printer output, 240–242
 - Printers. *See also* Centronics parallel interface
 - dot-matrix, 479–480
 - ink-jet, 480–481
 - laser, 480
 - page, 480
 - parallel driver program for, 255–259
 - parallel interface connections for, 252, 254–255
 - Privilege-level bits, 542
 - Procedure (PROC) directive, 161
 - Procedures
 - CALL instruction for, 43, 100–102, 106–107, 133–134, 591, 593
 - debugging programs containing, 115–116
 - defined, 99
 - dummy, 115
 - far (intersegment), 121–127
 - flowchart symbol for, 38
 - interrupt service, 116–117
 - macro versus, 127, 128–129
 - near (intra-segment), 103–106
 - nested, 99–100
 - nonreentrant, 111
 - parameters of, defined, 108
 - passing parameters to and from
 - in named memory, 109, 111
 - in registers, 109, 110
 - summary of, 115
 - using pointers, 111–113
 - using the stack, 113–115
 - program flow for, 99–100
 - recursive, 117–121
 - reentrant, 116–117, 537–538
 - RET instruction for, 43, 100, 101, 102, 107, 151, 591, 597
 - single, 99, 100
 - stacks and. *See* Stacks
 - stubs, 115

- Process control. *See* Industrial process control
- Process flowchart symbol, 38
- Processes. *See* Tasks
- Processor control instructions
8086, 44–45, 591
8087, 375
- Program development algorithm, 61, 62
- Program development tools, 59–62, 391–395
- Program execution transfer instructions, 43–44, 591
- Programmable AND matrixes, 12
- Programmable array logic (PAL), 12
- Programmable controllers, 320
- Programmable logic array (PLA), 12
- Programmable OR matrixes, 12
- Programmable read-only memory (PROM), 12, 15
- Programmer's model, 163
- Programming languages
8086 assembly language. *See* 8086 assembly language
C. *See* C programming language
high-level, 33
machine language, 32
for parallel computers, 572
- Programs. *See also* Software
8086 assembly language. *See* 8086 assembly language programs
abstracts for, 47
algorithms of. *See* Algorithms
assembler, 33, 45, 53, 60
bottom-up design of, 39
compiler, 33
debugger, 60, 61
debugging. *See* Debugging programs
design rule checker (DRC), 380
documentation of, 47
editor, 59–60
electrical rule checker (ERC), 380
error trapping for, 264
executive, 320, 321
flowcharts for. *See* Flowcharts
functions of, 47
interpreter, 33
linker, 60
locator, 60
loops in. *See* Loops
mainline, 320, 321
modules of, 39, 60, 122
monitor, 47, 61
netlist (wiring list), 380
pseudocode for. *See* Pseudocode
relocatable, 60, 74
schematic capture, 379–380, 381
simulator, 380, 382–385
structured, 39–42
stubs in, 115
subprograms, 30
system, 47
top-down design of, 39–42, 100
- PROM (programmable read-only memory), 12, 15
- Proportional feedback, 318, 319
- Proportional integral derivative (PID) control loops, 319, 320, 321
- Protected mode
80286, 543, 545–546, 547
80386. *See* 80386 protected mode
- Protocols
Binary Synchronous Communication Protocol (BISYNC), 518–520
bit-oriented (BOP), 520–522
byte-oriented (BCP), 518–520
- Protocols (Cont.)
defined, 518
high-level data link control (HDLC) protocol, 520–522
open systems interconnection (OSI) model for, 523–524
synchronous data link control (SDLC), 520
XMODEM, 519
- Prototyping
functions in C, 417–418
of microcomputer-based instruments, 331–332
simulation for, 380, 382–385
- Pseudo operations. *See* Assembler directives
- Pseudocode
CASE structure, 40, 81
comparing strings, 98
data sampling, 104
described, 39
downloading program, 510
factorials, 118
FOR-DO loop, 41, 90
IF-THEN-ELSE structure, 40, 77, 78, 80, 81
IF-THEN structure, 40, 77
interrupt input, 217
moving strings, 95–96
REPEAT-UNTIL structure, 40, 84, 86, 87, 88
sequence structure, 40
strobed input, 86
terminal emulator, 507
WHILE-DO structure, 40, 82, 83
- PSK (phase-shift keying) modulation, 499–500
- PTR (pointer) directive, 161
- PUBLIC directive, 122, 160, 161–162
- Pulse-code modulation (PCM), 502
- PUSH instruction
8086, 42, 107–108, 149, 590, 596
80186/80188, 336, 598–599
80286, 546
- PUSHA instruction
80186/80188, 42, 336, 599
80286, 546
- PUSHF instruction, 42, 149, 590, 596
- Pythagorean theorem, 375
- QAM (quaternary amplitude modulation), 499–500
- Quadrants, 499–500
- Quadword type, 158
- Quaternary amplitude modulation (QAM), 499–500
- Queue registers, 29, 30
- R (reset) inputs, 13
- R/M bit patterns, 48–50, 590, 591, 592
- RAM. *See* Random-access memory
- RAM (random-access memory) disks, 477
- RAMDACs, 450
- Ramp generators, 291, 294
- Random-access memory (RAM)
address decoding for, 187–188, 192–193
display refresh, 440, 441
- DRAM. *See* Dynamic random-access memory
- dual-ported, 448
microcomputer use of, 23
static (SRAM), 15–16, 353, 358, 360, 363
- Random-access memory (RAM) (Cont.)
video (VRAM), 446–447
volatile nature of, 15
- Random-access memory (RAM) disks, 477
- Raster scanning, 439–440
- RCL instruction
8086, 43, 70, 149–150, 590, 596
80186/80188, 599
- RCR instruction
8086, 43, 150–151, 590, 596
80186/80188, 599
- Read-only memory (ROM)
address decoding for, 186–187, 191–192
character generator, 440–442
description of, 14–15
electrically erasable programmable (EEPROM), 15
erasable programmable (EPROM), 15
flash EPROM, 15
mask-programmed, 15
microcomputer use of, 23
nonvolatile nature of, 14
optical (OROM), 478
programmable (PROM), 12, 15
- Read-write memory. *See* Random-access memory
- READY input, 164–167, 173, 175, 196–197
- Real mode
8086, 28
80286, 543, 544, 546–547
80386, 552, 553
- Real numbers, 366–368
- Real-time clocks, 220–221, 237–240
- Real transfers (8087), 372
- Recursive procedures, 117–121
- Red-green-blue (RGB) monitor, 444–445
- Redirected data, 471
- Reduced instruction set computer (RISC) processors, 461, 571
- Reentrant procedures, 116–117, 537–538
- Refresh controllers, 16, 354–355
- REG bits, 48, 49, 590, 591, 592
- Register addressing mode, 33–34
- Register storage class in C, 420
- Register-to-register architecture, 332, 333
- Registers
80386, 552, 553, 562–563
accumulator (AL), 29–30
AH, 29–30
AX, 30
base pointer (BP), 29, 31
BH, 29–30
BL, 29–30
BX, 30
CH, 29–30
CL, 29–30
code segment (CS), 29, 30–32, 89–90, 581
as counters, 86–89
CX, 30
data segment (DS), 29, 30–32, 34–35, 89–90, 581
data storage, 13
debug (80386), 552, 553
destination index (DI), 29, 31
DH, 29–30
DL, 29–30
DX, 30
extra segment (ES), 29, 30–32, 89–90, 581

Registers (Cont.)

- flag, 28, 29, 592
- general-purpose, 29-30
- index, 29, 32
- instruction pointer (IP), 24, 29, 30-31
- passing parameters in, 109, 110
- as pointers, 86-89
- queue, 29, 30
- segment, 29, 30-32, 34-35, 58-59, 581
- shift, 13, 464
- source index (SI), 29, 31
- stack pointer (SP), 29, 31
- stack segment (SS), 29, 30-32, 89-90, 581

Relational operators in C, 407

Relay drivers, 279-280

Relays

- mechanical, 280
- solid-state, 280-281

Relocatable programs, 60, 74

REP (repeat) instruction, 43, 96-97, 151, 591, 596

REPE instruction, 43, 151, 596

Repeat (REP) instruction, 43, 96-97, 151, 591, 596

REPEAT-UNTIL structure, 40, 41, 84-89, 91, 412-413

Repetition operations, 39-41

REPNE instruction, 43, 151, 596

REPNZ instruction, 43, 151, 596

REPZ instruction, 43, 151, 596

Reserved interrupts, 208

Reset (R) inputs, 13

RESET response of 8086, 164, 168, 583

Residual error, 319

Resistance temperature detectors (RTDs), 298

Resistor packs, 185

RET instruction, 43, 100, 101, 102, 107, 151, 591, 597

Return address, 100, 133

Reversed division instructions, 8087, 373

Reversed subtraction instructions, 8087, 373

RGB (red-green-blue) monitor, 444-445

Ring (circular) buffer, 515-517

Ring networks, 522, 523

RISC (reduced instruction set computer) processors, 461, 571

RLL (run-length-limited) coding for disks, 468, 472

Robots and robotics, 332, 464-465

ROL instruction

8086, 43, 70, 151-152, 590, 597

80186/80188, 599

ROM. See Read-only Memory

ROR instruction

8086, 43, 152, 590, 597

80186/80188, 599

ROTATE instruction

80186/80188, 336

80286, 546

Rotate instructions, 8086, 43

RS-232C standard, 494-497

RS-422A standard, 497-498

RS-423A standard, 497

RS-449 standard, 498

RTDs (resistance temperature detectors), 298

Run-length-limited (RLL) coding for disks, 468, 472

S (set) inputs, 13

SAHF instruction, 42, 152, 590, 597

SAL instruction

8086, 43, 153, 590, 597

80186/80188, 599

Sampling theorem, 340

SAR instruction

8086, 43, 153-154, 590, 597

80186/80188, 599

Saturation of amplifiers, 294

SBB instruction, 42, 154, 597

Scalable Processor Architecture (SPARC), 571

Scale. See Microcomputer-based scale

SCAS/SCASB/SCASW instructions, 43, 155, 591, 597-598

Scheduler, 537

Scheduling

- preemptive priority-based, 537
- terminate-and-stay-resident (TSR) programs and, 535-537
- time-slice, 537

Schematic diagrams

capture programs for, 379-380, 381

connector symbols on, 185

ICs on, 185

input signal lines on, 377, 378

jack (J) symbols on, 185

output signal lines on, 377, 378

plug (P) symbols on, 185

resistor packs on, 185

SDK-86, 176-184, 185

zone coordinates for, 185, 377

Scientific notation, 367

Scramblers, 500

Scrubbing process, 365

SCSI (small computer systems inter-

face) standard, 472-473

SDK-86. See also Microcomputer-based industrial process control system; Microcomputer-based scale

7-segment LCD interfacing with, 276-277

7-segment LED display interfacing with, 268-275

74LS138 address decoder added to, 222, 223

74LS164 wait-state generator, 174, 175, 177, 196-197

74LS244 drivers, 174, 179

74LS393 baud rate generator, 174, 175, 178, 184

74S373 address latches, 174, 175, 178, 185

2142 SRAM, 174, 175, 181, 192-193

2316/2716 PROM, 174, 175, 176, 191-192

3625 I/O decoder, 174, 182, 193-195

3625 off-board decoder, 174, 180, 195-196

3625 PROM decoder, 174, 175, 176, 191-192

3625 RAM decoder, 174, 175, 181, 192-193

8251A USART, 174, 175, 184

8254 programmable timer/counter added to, 222, 223

8255A programmable parallel ports, 174, 175, 180

8259A priority interrupt controller (PIC) added to, 222, 223, 235

8279 specialized I/O device, 174, 178-179, 182

SDK-86 (Cont.)

- 8284 clock generator, 173, 174, 175, 177
 - 8286 control and data transceivers, 174, 179
 - block diagram of, 174
 - bypass capacitors in, 176, 185
 - clock frequency of, 173
 - description of, 173-175, 178-179, 185
 - display driver for, 274-275
 - downloading programs to, 494, 496-497, 508-518
 - filter capacitors in, 176, 185
 - GO command, 68
 - input/output (I/O) addressing and decoding on, 174, 182, 193-195
 - instrument prototyping with, 331-332
 - keypad interfacing with, 266-274
 - off-board decoder, 174, 180, 195-196
 - parallel printer connection to, 252, 254-255
 - parallel printer driver program for, 255-259
 - port addressing and decoding on, 174, 182, 193-195
 - printer driver program for, 255-259
 - random-access memory (RAM) address decoding on, 174, 175, 181, 192-193
 - read-only memory (ROM) address decoding on, 174, 175, 176, 191-192
 - RS-232C interface for, 494, 496-497, 508-518
 - schematic diagrams of, 176-184, 185
 - single-step command, 68
 - wait-state generator, 174, 175, 177, 196-197
- ## SDK-386, 549, 560, 566
- ## SDLC (synchronous data link control) protocol, 520
- ## Second-generation microprocessors, 17
- ## Secondary stations, 521
- ## Seek time for disks, 466
- ## Segment base address, 30, 31, 32, 34, 49-50, 89-90
- ## Segment base/offset form of addresses, 31, 35
- ## SEGMENT directive, 53-54, 162
- ## Segment load instructions (80386), 563-564
- ## Segment override prefix, 49, 51-52, 591, 592
- ## Segment registers, 29, 30-32, 34-35, 58-59, 581
- ## Segment selector, 542
- ## Segmentation of memory, 30-32, 34-35
- ## Segments
- code, 58
 - data, 58
 - extra, 58
 - initializing segment registers, 58-59
 - logical, 53-54, 58
 - physical, 58
 - stack, 49, 58
- ## Selection flowchart symbol, 38
- ## Selection operations, 39, 40
- ## Semaphores, 538-539
- ## Semiconductor temperature sensors, 296-297

- Sensors. *See also* Transducers
 - defined, 290
 - flow, 300
 - light, 295-296
 - temperature, 296-298, 320, 322
- Sentinel method, 255
- Sequence structure, 39, 40, 65-67, 69-71
- Serial data communication. *See also* 8251A USART
 - asynchronous, 488
 - baud rate for, 488
 - full-duplex, 487, 488
 - half-duplex, 487, 488
 - marking state for, 488
 - modems for. *See* Modems
 - RS-232C standard for, 494-497
 - RS-422A standard for, 497-498
 - RS-423A standard for, 497
 - RS-449 standard for, 498
 - simplex, 487-488
 - start bit for, 488
 - stop bit for, 488
 - synchronous, 488
- Servo control, 317-318
- Session layer (OSI model), 523, 524
- Set (S) inputs, 13
- Set memory flag word instruction (80386), 564
- Set points, 317-318
- Settling time, 317-318
- SETxx instruction (80386), 564
- SF (sign flag), 28, 29, 74, 75
- SGDT instruction (80286), 547
- Shaft encoders
 - absolute, 283-284
 - defined, 283
 - incremental, 284-285
- Shift between words instructions (80386), 564
- SHIFT instruction
 - 80186/80188, 336
 - 80286, 546
- Shift instructions, 8086, 43
- Shift registers, 13, 464
- SHL instruction
 - 8086, 43, 153, 590, 597
 - 80186/80188, 599
- SHLD instruction (80386), 564
- Short jumps, 72-74, 76, 83-84
- SHORT operator, 162
- SHR instruction
 - 8086, 43, 155-156, 590, 597
 - 80186/80188, 599
- SHRD instruction (80386), 564
- SI (source index) register, 29, 31
- Sign bit, 6-8
- Sign flag (SF), 28, 29, 74, 75
- Signal assertion level, 11
- Signed numbers, 6-8
- Significand (mantissa) of numbers, 367
- Simple input/output (I/O), 245, 246
- Simple strobe input/output (I/O), 245-246
- Simplex communication, 487-488
- Simulator programs, 380, 382-385
- Single-board computers, 19, 21
- Single-density recording for disks, 467, 468
- Single-handshake input/output (I/O), 246
- Single indexed addressing mode, 49, 89-90
- Single-mode fibers, 506
- Single-precision numbers, 367
- Single-step command, 68
- Single-step interrupts, 208, 213, 216
- Slave device, 235
- SLDT instruction (80286), 547
- Slice, 27
- Small computer systems interface (SCSI) standard, 472-473
- SMSW instruction (80286), 547
- Snubber circuits, 281
- Soft errors, 363
- Software. *See also* Programs
 - defined, 24
 - upward-compatible, 28
- Software interrupts, 44, 52, 139-140, 207, 214, 216, 240-242, 591, 594
- Software-triggered strobes, 230, 231
- SOH (start-of-header) character, 518
- Solar cells, 296
- Solenoid drivers, 279-280
- Solid-state relays, 280-281
- Source file, 60
- Source index (SI) register, 29, 31
- Source for instructions, 32-33
- SP (stack pointer) register, 29, 31
- SPARC (Scalable Processor Architecture), 571
- Speech recognition, 481, 482-483
- Speech synthesis, 481-482
- Square-wave generators, 229-230
- Square waves, 337-338
- SRAM (static random-access memory), 15-16, 353, 358, 360, 363
- SS (stack segment) register, 29, 30-32, 89-90, 581
- SSB instruction, 590
- ST-506 standard, 472
- Stack diagrams, 102-103, 106-107, 108, 113-115, 118-121
- Stack overflow, 113-115
- Stack pointer (SP) register, 29, 31
- Stack segment, 49, 58
- Stack segment (SS) register, 29, 30-32, 89-90, 581
- Stacks
 - 8087, 369-370
 - data sampling program using, 103-106
 - defined, 30
 - operation of, 102-103, 106-107
 - passing parameters using, 113-115
 - POP instruction for, 42, 107-108, 149, 590, 596
 - PUSH instruction for, 42, 107-108, 149, 590, 596
 - top of stack, 31
 - uses for, 102-103
- Standard structures, 39-42
- Star networks, 522
- Start bit, 488
- Start flowchart symbol, 37-38
- Start-of-header (SOH) character, 518
- Start-of-text (STX) character, 518
- State (context, or environment) of tasks, 537-538
- States
 - of clocks, 165
 - undefined, 223
- Static displays, 267, 268
- Static random-access memory (SRAM), 15-16, 353, 358, 360, 363
- Static storage class in C, 420
- STC instruction, 44, 156, 591, 597
- STD instruction, 44, 156, 591, 597
- Stepper motors, 281-283
- STI instruction, 44, 156, 215, 591, 597
- Stimulus files, 383-384
- Stop bit, 488
- Stop flowchart symbol, 38
- STOS/STOSB/STOSW 'nstructions, 43, 156, 591, 597-598
- STR instruction (80286), 547
- Strain gages, 298-299
- Streaming tape systems, 477
- String instructions, 43, 95-99, 591
- String library functions in C, 424-425
- Strings
 - comparing, 97-99
 - defined, 43, 95
 - moving, 95-97
- Strobe input/output (I/O), 245-246
- Strobes
 - described, 84-86
 - hardware-triggered, 230-231
 - software-triggered, 230, 231
- Structured programming, 39-42
- Stubs, 115
- STX (start-of-text) character, 518
- SUB instruction, 42, 154-155, 590, 598
- Subprograms. *See* Procedures
- Subroutine flowchart symbol, 38
- Subroutines. *See* Procedures
- Subtraction
 - 8086 instructions for, 42-43
 - 8087 instructions for, 373
 - binary, 8
 - binary-coded decimal (BCD), 10
- Successive approximation analog-to-digital (A/D) converters, 305, 307
- Summing point, 294
- Supercomputers, 19, 20, 571-572
- Supersets of instructions, 28
- Supervisor, 537
- Switch structure in C, 411-412
- Switched capacitor digital filters, 342
- Switched phone lines, 488
- Symbol table, 57, 60, 211
- Sync characters, 518
- Synchronization instructions, 44-45
- Synchronous communication, 488
- Synchronous data link control (SDLC) protocol, 520
- Syndrome word, 364-365
- Syntax of assembly language, 47
- System commands, 47
- System degradation for time-sliced systems, 537
- System expansion slots, 345
- System program, 47
- Tachometers, 317-318
- Tail pointer, 515-517
- Tape readers, 250-252
- Tasks
 - blocked, 538
 - defined, 534
 - environment (context, or state) of, 537-538
 - mutual exclusion of, 538
- TDM (time-division multiplexing), 502-503
- Temperature sensors, 296-298, 320, 322
- Templates for instructions, 47-53
- Ten-byte type, 158-159
- Terabyte (unit), 27
- Terminal emulator program, 507-508, 509
- Terminals, 440
- Terminate-and-stay-resident (TSR) programs, 535-537

- TEST instruction, 43, 156–157, 590, 598
- Texas Instruments Graphics Architecture (TIGA) standard, 461
- TF (trap flag), 29, 207–209, 213–216
- Thermal printers, 479–480
- Thermal sensitive resistors (thermistors), 298
- Thermocouples, 297–298
- Thin Ethernet (10BaseT) networks, 525
- Thrashing process, 361
- Three-state outputs, 14–15, 24
- TIGA (Texas Instruments Graphics Architecture) standard, 461
- Time-division multiplexing (TDM), 502–503
- Time-domain description, 337, 338
- Time-multiplexed systems, 20–22, 35
- Time-slice scheduling, 537
- Time-sliced systems, 20–22, 35, 320
- Timed interrupt generators, 228–229
- Timesharing systems, 20–22, 35
- Timing
- 8086 bus, 584
 - 8086 instructions, 91–93, 592–598
 - 8087 instructions, 599–605
 - delay loops for, 91–93, 103–106
 - effective address (EA), 592
 - interrupts for, 219–221
 - segment override prefix, 592
 - Timing parameters, 8086, 197–200
- Timing waveforms
- 8086 maximum mode, 588, 589
 - 8086 minimum mode, 197–200, 586, 587, 589
 - 8086 system timing, 164–167, 583, 584
 - 8237 DMA controller, 351–352
 - 8254 programmable timer/counter, 226–231
 - 8255 handshake data input from a tape, 251–252
 - 8279, 269–271
- Centronics parallel interface, 254–255
- clock, 165
- data acquisition system (DAS), 322–323
- double-handshake input/output (I/O), 246–247
- dynamic random-access memory (DRAM), 353–354
- simple input/output (I/O), 245, 246
- simple strobe input/output (I/O), 245–246
- single-handshake input/output (I/O), 246
- Token, defined, 523
- Token (file handle), 474–475
- Token-passing ring networks, 522, 523, 525–526, 527
- Top-down design, 39–42, 100
- Top of stack, 31
- Topologies
- hypercube, 571–572
 - network, 522–523
- Trace data, 62, 170, 171–172
- Trackballs, 462–463
- Tracks of disks, 465, 468–469
- Transceivers, 524
- Transcendental instructions (8087), 374–375
- Transducers. *See also* Sensors
- defined, 298
 - differential pressure, 300
 - Transducers (*Cont.*)
 - force, 298–300, 307, 308, 309
 - pressure, 299–300
 - Transistor buffers, 278–280
 - Transistors
 - Darlington, 278–279
 - isolated-gate bipolar (IGBTs), 279–280
 - metal-oxide-semiconductor field-effect (MOSFETs), 279–280
 - Transport layer (OSI model), 523, 524
 - Trap flag (TF), 29, 207–209, 213–216
 - Tree-structured (broadband bus) networks, 522, 523
 - Trees (structures), 117
 - Trellis code, 500
 - Triacs, 280–281
 - Tribits, 4–9
 - Trigonometric instructions (8087), 374
 - Troubleshooting microcomputers, 200–204
 - TSR (terminate-and-stay-resident) programs, 535–537
 - Two-key lockout, 262, 273
 - Two-key rollover, 265, 273
 - Type byte, 34, 55, 158
 - Type conversion instructions (80386), 563
 - Type doubleword, 55, 158
 - Type error, 34
 - TYPE operator, 162
 - Type quadword, 158
 - Type ten bytes, 158–159
 - Type word, 34, 55, 159
- UART (universal asynchronous receiver-transmitter), 488. *See also* 8250 UART
- Unconditional jumps, 71, 72–74
- Unconditional transfer instructions, 43
- Undefined states, 223
- Underdamped response, 317–318
- Underflow, 8
- Union data structure in C, 437–439
- Unipolar binary-coded decimal (BCD) codes, 306
- Unipolar binary codes, 306
- Unity-gain bandwidth, 293
- Universal asynchronous receiver-transmitter (UART), 488. *See also* 8250 UART
- Universal Synchronous/Asynchronous Receiver Transmitter (USART), 174, 175, 184, 488. *See also* 8251A USART
- Unpacked binary-coded decimal (BCD) code, 69–71
- Unvoiced sounds, 481
- Upward-compatible software, 28
- USART (Universal Synchronous/Asynchronous Receiver Transmitter), 174, 175, 184, 488. *See also* 8251A USART
- Variable-port instructions, 78, 193
- Variable storage classes in C, 419–420
- Variables
- char (character), 396–397
 - dummy, 128
 - float (floating-point), 398
 - int (Integer), 397–398
 - named, 55–58
 - types in C, 396–398
- VERR instruction (80286), 547
- VERW instruction (80286), 547
- VGA (video graphics array), 447, 449–451
- Video cameras, 463–464
- Video digital-to-anaalog (D/A) converters, 449–450
- Video graphics array (VGA), 447, 449–451
- Video monitors, 440
- Video programming
- high-level, 458–460
 - introduced, 451
 - low-level, 451–458
- Video random-access memory (VRAM), 446–447
- Vidicons, 463–464
- Virtual 8086 mode (80386), 561–562
- Virtual address mode, 28
- Virtual addresses, 541–543
- Virtual ground, 293–294
- Virtual memory, 541–543
- Vocal tract model, 481
- Voiced sounds, 481
- Volatile memory, 15
- Voltage gain, 290, 292
- Von Neumann architecture, 339
- VRAM (video random-access memory), 446–447
- W bit, 48, 590, 591
- WAIT instruction, 44, 157, 584, 591, 598, 599–605
- WAIT states, 164–167, 175, 196–197
- WBINVD instruction (80486), 570
- WHILE-DO structure, 40, 41, 82–84, 412–413
- While structure in C, 412–413
- Winchester hard disks, 467
- Windows program, 567–568
- Wiring list (netlist) program, 380
- Word transfer instructions, 42
- Word type, 34, 55, 159
- Words
- 8086 storage of, 35
 - binary. *See* Binary words
 - command. *See* Control words
 - control. *See* Control words
 - defined, 1
 - mode. *See* Control words
 - syndrome, 364–365
- Workstations, 571
- Write once/read many (WORM) disks, 478
- XADD instruction (80486), 570
- XCHG instruction, 42, 157, 590, 598
- XLAT instruction, 42, 157, 266–267, 590, 598
- XLATB instruction, 157, 598
- XMODEM protocol, 519
- XMS (extended) memory, 541
- XNOR (exclusive NOR) gate, 11, 12
- XOR (exclusive OR) gate, 11, 12
- XOR instruction, 43, 157–158, 591, 598
- Z80 microprocessor, 27
- Zadeh, Lofti A., 574
- Zero flag (ZF), 28, 29, 74, 75
- Zero-point switching, 281
- ZF (zero flag), 28, 29, 74, 75
- Zone coordinates, 185, 377