| | |
|---|---|
| | 0EH = return logical device map<br>Entry: BL = drive number<br>Exit: AL = number of last device<br>0FH = change logical device map<br>Entry: BL = drive number<br>Exit: AL = number of last device |
| **45H** | DUPLICATE FILE HANDLE |
| Entry | AH = 45H<br>BX = current file handle |
| Exit | AX = error code if carry set<br>AX = duplicate file handle |
| **46H** | FORCE DUPLICATE FILE HANDLE |
| Entry | AH = 46H<br>BX = current file handle<br>CX = new file handle |
| Exit | AX = error code if carry set |
| Notes | This function works like function 45H except that function 45H allows DOS to select the new handle, while this function allows the user to select the new handle. |
| **47H** | READ CURRENT DIRECTORY |
| Entry | AH = 47H<br>DL = drive number<br>DS:SI = address of a 64-byte buffer for directory name |
| Exit | DS:SI addresses current directory name if carry cleared |
| Notes | Drive A = 00, drive B = 01, and so forth |
| **48H** | ALLOCATE MEMORY BLOCK |
| Entry | AH = 48H<br>BX = number of paragraphs to allocate<br>CX = new file handle |
| Exit | BX = largest block available if carry cleared |
| **49H** | RELEASE ALLOCATED MEMORY BLOCK |
| Entry | AH = 49H<br>ES = segment address of block to be released<br>CX = new file handle |
| Exit | Carry indicates an error if set |

| **4AH** | MODIFY ALLOCATED MEMORY BLOCK |
|---|---|
| Entry | AH = 4AH<br>BX = new block size in paragraphs<br>ES = segment address of block to be modified |
| Exit | BX = largest block available if carry cleared |

| **4BH** | LOAD OR EXECUTE A PROGRAM |
|---|---|
| Entry | AH = 4BH<br>AL = function code<br>ES:BX = address of parameter block<br>DS:DX = address ASCII-Z string command |
| Exit | Carry indicates an error if set |
| Notes | The function codes are AL = 00H to load and execute a program, AL = 01H to load a program but not execute it, AL = 03H to load a program overlay, and AL = 05H to enter the EXEC state. Figure B–6 (p. 610) shows the parameter block used with this function. |

| **4CH** | TERMINATE A PROCESS |
|---|---|
| Entry | AH = 4CH<br>AL = error code |
| Exit | Returns control to DOS |
| Notes | This function returns control to DOS with the error code saved so it can be obtained using DOS ERROR LEVEL batch processing system. We normally use this function with an error code of 00H to return to DOS. |

| **4DH** | READ RETURN CODE |
|---|---|
| Entry | AH = 4DH |
| Exit | AX = return error code |
| Notes | This function is used to obtain the return status code created by executing a program with DOS function 4BH. The return codes are AX = 0000H for a normal-no error-termination, AX = 0001H for a control-break termination, AX = 0002H for a critical device error, and AX = 0003H for a termination by an INT 31H. |

| 4EH | FIND FIRST MATCHING FILE |
|---|---|
| Entry | AH = 4EH<br>CX = file attributes<br>DS:DX = address ASCII-Z string file name |
| Exit | Carry is set for file not found |
| Notes | This function searches the current or named directory for the first matching file. Upon exit, the DTA contains the file information. See Figure B–7 (p. 610) for the disk transfer area (DTA). |

| 4FH | FIND NEXT MATCHING FILE |
|---|---|
| Entry | AH = 4FH |
| Exit | Carry is set for file not found |
| Notes | This function is used after the first file is found with function 4EH. |

| 50H | SET PROGRAM SEGMENT PREFIX (PSP) ADDRESS |
|---|---|
| Entry | AH = 50H<br>BX = offset address of the new PSP |
| Notes | Extreme care must be used with this function because no error recovery is possible. |

| 51H | GET PSP ADDRESS |
|---|---|
| Entry | AH = 51H |
| Exit | BX = current PSP segment address |

| 54H | READ DISK VERIFY STATUS |
|---|---|
| Entry | AH = 54H |
| Exit | AL = 00H if verify off<br>AL = 01H if verify on |

| 56H | RENAME FILE |
|---|---|
| Entry | AH = 56H<br>ES:DI = address of ASCII-Z string containing new file name<br>DS:DX = address of ASCII-Z string containing file to be renamed |
| Exit | Carry is set for error condition |

| 57H | READ FILE'S DATE AND TIME STAMP |
|---|---|
| Entry | AH = 57H<br>AL = function code<br>BX = file handle<br>CX = new time<br>DX = new date |
| Exit | Carry is set for error condition<br>CX = time if carry cleared<br>DX = date if carry cleared |
| Notes | AL = 00H to read date and time or 01H to write date and time. |
| **59H** | **GET EXTENDED ERROR INFORMATION** |
| Entry | AH = 59H<br>BX = 0000H for DOS version 3.X |
| Exit | AX = extended error code<br>BH = error class<br>BL = recommended action<br>CH = locus |
| Notes | Following are the extended error codes found in AX:<br>0001H = invalid function number<br>0002H = file not found<br>0003H = path not found<br>0004H = no file handles available<br>0005H = access denied<br>0006H = file handle invalid<br>0007H = memory control block failure<br>0008H = insufficient memory<br>0009H = memory block address invalid<br>000AH = environment failure<br>000BH = format invalid<br>000CH = access code invalid<br>000DH = data invalid<br>000EH = unknown unit<br>000FH = disk drive invalid<br>0010H = attempted to remove current directory<br>0011H = not same device<br>0012H = no more files<br>0013H = disk write-protected<br>0014H = unknown unit<br>0015H = drive not ready<br>0016H = unknown command<br>0017H = data error (CRC check error)<br>0018H = bad request structure length<br>0019H = seek error<br>001AH = unknown media type<br>001BH = sector not found<br>001CH = printer out of paper<br>001DH = write fault<br>001EH = read fault |

001FH = general failure
0020H = sharing violation
0021H = lock violation
0022H = disk change invalid
0023H = FCB unavailable
0024H = sharing buffer exceeded
0025H = code page mismatch
0026H = handle end of file operation not completed
0027H = disk full
0028H–0031H reserved
0032H = unsupported network request
0033H = remote machine not listed
0034H = duplicate name on network
0035H = network name not found
0036H = network busy
0037H = device no longer exists on network
0038H = netBIOS command limit exceeded
0039H = error in network adapter hardware
003AH = incorrect response from network
003BH = unexpected network error
003CH = remote adapter is incompatible
003DH = print queue is full
003EH = not enough room for print file
003FH = print file was deleted
0040H = network name deleted
0041H = network access denied
0042H = incorrect network device type
0043H = network name not found
0044H = network name exceeded limit
0045H = netBIOS session limit exceeded
0046H = temporary pause
0047H = network request not accepted
0048H = print or disk redirection pause
0049H–004FH reserved
0050H = file already exists
0051H = duplicate FCB
0052H = cannot make directory
0053H = failure in INT 24H (critical error)
0054H = too many re-directions
0055H = duplicate redirection
0056H = invalid password
0057H = invalid parameter
0058H = network write failure
0059H = function not supported by network
005AH = required system component not installed
0065H = device not selected

Following are the error class codes as found in BH:

01H = no resources available
02H = temporary error
03H = authorization error
04H = internal software error
05H = hardware error
06H = system failure
07H = application software error
08H = item not found
09H = invalid format
0AH = item blocked
0BH = media error

| | |
|---|---|
| | 0CH = item already exists<br>0DH = unknown error<br>Following is the recommended action as found in BL:<br>01H = retry operation<br>02H = delay and retry operation<br>03H = user retry<br>04H = abort processing<br>05H = immediate exit<br>06H = ignore error<br>07H = retry with user intervention<br>Following is a list of locus in CH:<br>01H = unknown source<br>02H = block device error<br>03H = network area<br>04H = serial device error<br>05H = memory error |
| **5AH** | CREATE UNIQUE FILE NAME |
| Entry | AH = 5AH<br>CX = attribute code<br>DS:DX = address of the ASCII-Z string directory path |
| Exit | Carry is set for error condition<br>AX = file handle if carry cleared<br>DS:DX = address of the appended directory name |
| Notes | The ASCII-Z file directory path must end with a backslash (\). On exit, the directory name is appended with a unique file name. |
| **5BH** | CREATE A DOS FILE |
| Entry | AH = 5BH<br>CX = attribute code<br>DS:DX = address of the ASCII-Z string contain the file name |
| Exit | Carry is set for error condition<br>AX = file handle if carry cleared |
| Notes | The function works only in DOS version 3.X or higher. It is almost identical to function 3CH, except that function 3CH erases the file, if it already exists, while function 5BH reports that the file exists without erasing it. |
| **5CH** | LOCK/UNLOCK FILE CONTENTS |
| Entry | AH = 5CH<br>BX = file handle<br>CX:DX = offset address of locked/unlocked area<br>SI:DI = number of bytes to lock or unlock beginning at offset |
| Exit | Carry is set for error condition |

| | |
|---|---|
| **5DH** | SET EXTENDED ERROR INFORMATION |
| Entry | AH = 5DH<br>AL = 0AH<br>DS:DX = address of the extended error data structure |
| Notes | This function is used by DOS version 3.1 or higher to store extended error information. |
| **5EH** | NETWORK/PRINTER |
| Entry | AH = 5EH<br>AL = 00H (get network name)<br>DS:DX = address of the ASCII-Z string containing network name |
| Exit | Carry is set for error condition<br>CL = netBIOS number if carry cleared |
| Entry | AH = 5EH<br>AL = 02H (define network printer)<br>BX = redirection list<br>CX = length of setup string<br>DS:DX = address of printer setup buffer |
| Exit | Carry is set for error condition |
| Entry | AH = 5EH<br>AL = 03H (read network printer setup string)<br>BX = redirection list<br>DS:DX = address of printer setup buffer |
| Exit | Carry is set for error condition<br>CX = length of setup string if carry cleared<br>ES:DI = address of printer setup buffer |
| **62H** | GET PSP ADDRESS |
| Entry | AH = 62H |
| Exit | BX = segment address of the current program |
| Notes | The function works only in DOS version 3.0 or higher. |
| **65H** | GET EXTENDED COUNTRY INFORMATION |
| Entry | AH = 65H<br>AL = function code<br>ES:DI = address of buffer to receive information |
| Exit | Carry is set for error condition<br>CX = length of country information |
| Notes | The function works only in DOS version 3.3 or higher. |

| **66H** | GET/SET CODE PAGE |
|---|---|
| Entry | AH = 66H<br>AL = function code<br>BX = code page number |
| Exit | Carry is set for error condition<br>BX = active code page number<br>DX = default code page number |
| Notes | A function code in AL of 01H gets the code page number, and a code of 02H sets the code page number. |

| **67H** | SET HANDLE COUNT |
|---|---|
| Entry | AH = 67H<br>BX = number of handles desired |
| Exit | Carry is set for error condition |
| Notes | This function is available for DOS version 3.3 or higher. |

| **68H** | COMMIT FILE |
|---|---|
| Entry | AH = 68H<br>BX = handle number |
| Exit | Carry is set for error condition; otherwise, the date and time stamp is written to directory. |
| Notes | This function is available for DOS version 3.3 or higher. |

| **6CH** | EXTENDED OPEN FILE |
|---|---|
| Entry | AH = 6CH<br>AL = 00H<br>BX = open mode<br>CX = attributes<br>DX = open flag<br>DS:SI = address of ASCII-Z string file name |
| Exit | AX = error code if carry is set<br>AX = handle if carry is cleared<br>CX = 0001H file existed and was opened<br>CX = 0002H file did not exist and was created |
| Notes | This function is available for DOS version 4.0 or higher |

| Offset | Contents |
|---|---|
| 00H | Drive |
| 01H | 8-character filename |
| 09H | 3-character file extension |
| 0CH | Current block number |
| 0EH | Record size |
| 10H | File size |
| 14H | Creation date |
| 16H | Reserved space |
| 20H | Current record number |
| 21H | Relative record number |

**FIGURE B–2** Contents of the file control block (FCB).

| Offset | Contents |
|---|---|
| 00H | Drive |
| 01H | 8-character filename |
| 09H | 3-character extension |
| 0CH | Current block number |
| 0EH | Record size |
| 10H | File size |
| 14H | Creation date |
| 16H | Second file name |

**FIGURE B–3** Contents of the modified file control block (FCB).

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? | ? |

Bit 0 = 0 if not two-sided
= 1 if two-sided

Bit 1 = 0 if not eight sectors per track
= 1 if eight sectors per track

Bit 2 = 0 if nonremovable
= 1 if removable

**FIGURE B–4** Contents of the media-descriptor byte.

| Offset | Contents |
|---|---|
| 00H | INT 20H |
| 02H | Top of memory |
| 04H | Reserved |
| 05H | Opcode |
| 06H | Number of bytes in segment |
| 0AH | Terminate address (offset) |
| 0CH | Terminate address (segment) |
| 0EH | Control-break address (offset) |
| 10H | Control-break address (segment) |
| 12H | Critical error address (offset) |
| 14H | Critical error address (segment) |
| 16H | Reserved |
| 2CH | Environment address (segment) |
| 2EH | Reserved |
| 50H | DOS call |
| 52H | Reserved |
| 5CH | File control block 1 |
| 6CH | File control block 2 |
| 80H | Command line length |
| 81H | Command line |

**FIGURE B–5** Contents of the program segment prefix (PSP).

(a)

| Offset | Contents |
|---|---|
| 00H | Environment address (segment) |
| 02H | Command line address (offset) |
| 04H | Command line address (segment) |
| 06H | File control block 1 address (offset) |
| 08H | File control block 1 address (segment) |
| 0AH | File control block 2 address (segment) |
| 0CH | File control block 2 address (offset) |

(b)

| Offset | Contents |
|---|---|
| 00H | Overlay destination segment address |
| 02H | Relocation factor |

**FIGURE B–6** The parameter blocks used with function 4BH (EXEC). (a) For function code 00H. (b) For function code 03H.

| Offset | Contents |
|---|---|
| 15H | Attributes |
| 16H | Creation time |
| 18H | Creation date |
| 1AH | Low word file size |
| 1CH | High word file size |
| 1EH | Search file name |

**FIGURE B–7** Data transfer area (DTA) used to find a file.

## BIOS FUNCTION CALLS

In addition to DOS function call INT 21H, some other BIOS function calls are useful in controlling the I/O environment of the computer. Unlike INT 21H, which exists in the DOS program, the BIOS function calls are found stored in the system and video BIOS ROMs. These BIOS ROM functions directly control the I/O devices, with or without DOS loaded into a system.

### INT 10H

The INT 10H BIOS interrupt is often called the *video services interrupt* because it directly controls the video display in a system. The INT 10H instruction uses register AH to select the video service provided by this interrupt. The video BIOS ROM is located on the video board and varies from one video card to another.

**TABLE B–4** Video display modes.

| *Mode* | *Type* | *Columns* | *Rows* | *Resolution* | *Standard* | *Colors* |
|---|---|---|---|---|---|---|
| 00H | Text | 40 | 25 | 320 x 200 | CGA | 2 |
| 00H | Text | 40 | 25 | 320 x 250 | EGA | 2 |
| 00H | Text | 40 | 25 | 360 x 400 | VGA | 2 |
| 01H | Text | 40 | 25 | 320 x 200 | CGA | 16 |
| 01H | Text | 40 | 25 | 320 x 350 | EGA | 16 |
| 01H | Text | 40 | 25 | 360 x 640 | VGA | 16 |
| 02H | Text | 80 | 25 | 640 x 200 | CGA | 2 |
| 02H | Text | 80 | 25 | 640 x 350 | EGA | 2 |
| 02H | Text | 80 | 25 | 720 x 400 | VGA | 2 |
| 03H | Text | 80 | 25 | 640 x 200 | CGA | 16 |
| 03H | Text | 80 | 25 | 640 x 350 | EGA | 16 |
| 03H | Text | 80 | 25 | 720 x 400 | VGA | 16 |
| 04H | Graphics | 80 | 25 | 320 x 200 | CGA | 4 |
| 05H | Graphics | 80 | 25 | 320 x 350 | CGA | 2 |
| 06H | Graphics | 80 | 25 | 640 x 200 | CGA | 2 |
| 07H | Text | 80 | 25 | 720 x 350 | EGA | 4 |
| 07H | Text | 80 | 25 | 720 x 400 | VGA | 4 |
| 0DH | Graphics | 80 | 25 | 320 x 200 | CGA | 16 |
| 0EH | Graphics | 80 | 25 | 640 x 200 | CGA | 16 |
| 0FH | Graphics | 80 | 25 | 640 x 350 | EGA | 4 |
| 10H | Graphics | 80 | 25 | 640 x 350 | EGA | 16 |
| 11H | Graphics | 80 | 30 | 640 x 480 | VGA | 2 |
| 12H | Graphics | 80 | 30 | 640 x 480 | VGA | 16 |
| 13H | Graphics | 40 | 25 | 320 x 200 | VGA | 256 |

***Video Mode Selection.*** The mode of operation for the video display is accomplished by placing a 00H into AH, followed by one of many mode numbers in AL. Table B-4 lists the modes of operation found in video display systems using standard video modes. The VGA can use any mode listed, whereas the other displays are more restrictive in use. Additional higher resolution modes are explained later in this section.

Example B–7 lists a short sequence of instructions that place the video display into mode 03H operation. This mode is available on CGA, EGA, and VGA displays. This mode allows the display to draw test data with 16 colors at various resolutions, dependent upon the display adapter.

**EXAMPLE B–7**

```
0000 B4 00          MOV   AH,0          ;select mode
0002 B0 03          MOV   AL,3          ;mode is 03H
0004 CD 10          INT   10H
```

***Cursor Control and Other Standard Features.*** Table B–5 shows the function codes (placed in AH) used to control the cursor on the video display. These cursor control functions will work on any video display, from the CGA display to the latest super VGA display. It also lists the functions used to display data and change to a different character set.

**TABLE B–5** Video BIOS (INT 10H) functions (pp. 612–614).

| | |
|---|---|
| **00H** | SELECT VIDEO MODE |
| Entry | AH = 00H<br>AL = mode number |
| Exit | Mode changed and screen cleared |
| **01H** | SELECT CURSOR TYPE |
| Entry | AH = 01H<br>CH = starting line number<br>CL = ending line number |
| Exit | Cursor size changed |
| **02H** | SELECT CURSOR POSITION |
| Entry | AH = 02H<br>BH = page number (usually 0)<br>DH = row number (beginning with 0)<br>DL = column number (beginning with 0) |
| Exit | Changes cursor to new position |
| **03H** | READ CURSOR POSITION |
| Entry | AH = 03H<br>BH = page number |
| Exit | CH = starting line (cursor size)<br>CL = ending line (cursor size)<br>DH = current row<br>DL = current column |
| **04H** | READ LIGHT PEN |
| Entry | AH = 04H (not supported in VGA) |
| Exit | AH = 0, light pen triggered<br>BX = pixel column<br>CX = pixel row<br>DH = character row<br>DL = character column |

| 05H | SELECT DISPLAY PAGE |
|---|---|
| Entry | AH = 05H<br>AL = page number |
| Exit | Page number selected. Following are the valid page numbers:<br>Mode 0 and 1 support pages 0–7<br>Mode 2 and 3 support pages 0–7<br>Mode 4, 5, and 6 support page 0<br>Mode 7 and D support pages 0–7<br>Mode E supports pages 0–3<br>Mode F and 10 support pages 0–1<br>Mode 11, 12, and 13 support page 0 |
| Notes | Most modern displays use page 0 for most operations. |

| 06H | SCROLL PAGE UP |
|---|---|
| Entry | AH = 06H<br>AL = number of lines to scroll (0 clears window)<br>BH = character attribute for new lines<br>CH = top row of scroll window<br>CL = left column of scroll window<br>DH = bottom row of scroll window<br>DL = right column of scroll window |
| Exit | Scrolls window from the bottom toward the top of the screen. Blank lines fill the bottom using the character attribute in BH. |

| 07H | SCROLL PAGE DOWN |
|---|---|
| Entry | AH = 07H<br>AL = number of lines to scroll (0 clears window)<br>BH = character attribute for new lines<br>CH = top row of scroll window<br>CL = left column of scroll window<br>DH = bottom row of scroll window<br>DL = right column of scroll window |
| Exit | Scrolls window from the top toward the bottom of the screen. Blank lines fill from the top using the character attribute in BH. |

| 08H | READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION |
|---|---|
| Entry | AH = 08H<br>BH = page number |
| Exit | AL = ASCII character code<br>AH = character attribute |
| Notes | This function does not advance the cursor. |

| | |
|---|---|
| **09H** | WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION |
| Entry | AH = 09H<br>AL = ASCII character code<br>BH = page number<br>BL = character attribute<br>CX = number of characters to write |
| Notes | This function does not advance the cursor. |
| **0AH** | WRITE CHARACTER AT CURRENT CURSOR POSITION |
| Entry | AH = 0AH<br>AL = ASCII character code<br>BH = page number<br>CX = number of characters to write |
| Notes | This function does not advance the cursor. |
| **0FH** | READ VIDEO MODE |
| Entry | AH = 0FH |
| Exit | AL = current video mode<br>AH = number of character columns<br>BH = page number |
| **10H** | SET VGA PALETTE REGISTER |
| Entry | AH = 10H<br>AL = 10H<br>BX = color number (0–255)<br>CH = green (0–63)<br>CL = blue (0–63)<br>DH = red (0–63) |
| Exit | Palette register color is changed. Note: The first 16 colors (0–15) are used in the 16-color, VGA text mode and other modes. |
| **10H** | READ VGA PALETTE REGISTER |
| Entry | AH = 10H<br>AL = 15H<br>BX = color number (0–255) |
| Exit | CH = green<br>CL = blue<br>DH = red |

| 11H | GET ROM CHARACTER SET |
|---|---|
| Entry | AH = 11H<br>AL = 30H<br>BH = 2 = ROM 8 x 14 character set<br>BH = 3 = ROM 8 x 8 character set<br>BH = 4 = ROM 8 x 8 extended character set<br>BH = 5 = ROM 9 x 14 character set<br>BH = 6 = ROM 8 x 16 character set<br>BH = 7 = ROM 9 x 16 character set |
| Exit | CX = bytes per character<br>DL = rows per character<br>ES:BP = address of character set |

If an SVGA (super VGA), EVGA (extended VGA), or XVGA (also extended VGA) adapter is available, the super VGA mode is set by using INT 10H function call AX = 4F02H, with BX = to the VGA mode for these advanced display adapters. This conforms to the VESA standard for VGA adapters. Table B-6 shows the modes selected by register BX for this INT 10H function call. Most video cards are equipped with a driver called VVESA.COM or VVESA.SYS, which conforms the card to the VESA standard functions

**TABLE B–6** Extended VGA functions.

| *BX* | *Extended Mode* |
|---|---|
| 100H | 640 x 400 with 256 colors |
| 101H | 640 x 480 with 256 colors |
| 102H | 800 x 600 with 16 colors |
| 103H | 800 x 600 with 256 colors |
| 104H | 1024 x 768 with 16 colors |
| 105H | 1024 x 768 with 256 colors |
| 106H | 1280 x 1024 with 16 colors |
| 107H | 1280 x 1024 with 256 colors |
| 108H | 80 x 60 in text mode |
| 109H | 132 x 25 in text mode |
| 10AH | 132 x 43 in text mode |
| 10BH | 132 x 50 in text mode |
| 10CH | 132 x 60 in text mode |

## INT 11H

This function is used to determine the type of equipment installed in the system. To use this call, the AX register is loaded with an FFFFH, and then the INT 11H instruction is executed. In return, an INT 11H provides information, as listed in Figure B–8.

## INT 12H

The memory size is returned by the INT 12H instruction. After executing the INT 12H instruction, the AX register contains the number of 1K-byte blocks of memory (conventional memory in the first 1M bytes of address space) installed in the computer.

## INT 13H

This call controls the diskettes ($5^{1}/4$" or $3^{1}/2$") and also fixed or hard disk drives attached to the system. Table B–7 lists the functions available to this interrupt via register AH. The direct control of a floppy disk or hard disk can lead to problems. Therefore we provide only a listing of the functions, without details of their usage. Before using these functions, refer to the BIOS literature available from the company that produced your version of the BIOS ROM. Never use these functions for normal disk operations.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P0 |  | G | S2 | S1 | S0 | D2 | D1 |  |  |  |  |  |  |  |

P1, P0 = number of parallel ports
G = 1 if game I/O attached
S2, S1, S0 = number of serial ports
D2, D1 = number of disk drives

**FIGURE B–8** The contents of AX as it indicates the equipment attached to the computer.

**TABLE B–8** COM port interrupt INT 14H.

| *AH* | *Function* |
|---|---|
| 00H | Initialize communications port |
| 01H | Send character |
| 02H | Receive character |
| 03H | Get COM port status |
| 04H | Extended initialize communications port |
| 05H | Extended communications port control |

**TABLE B–7** Disk I/O function via INT 13H.

| *AH* | *Function* |
|---|---|
| 00H | Reset the system disk |
| 01H | Read disk status to AL |
| 02H | Read sector |
| 03H | Write sector |
| 04H | Verify sector |
| 05H | Format track |
| 06H | Format bad track |
| 07H | Format drive |
| 08H | Get drive parameters |
| 09H | Initialize fixed disk characteristics |
| 0AH | Read long sector |
| 0BH | Write long sector |
| 0CH | Seek |
| 0DH | Reset fixed disk system |
| 0EH | Read sector buffer |
| 0FH | Write sector buffer |
| 10H | Get drive status |
| 11H | Re-calibrate drive |
| 12H | Controller RAM diagnostics |
| 13H | Controller drive diagnostics |
| 14H | Controller internal diagnostics |
| 15H | Get disk type |
| 16H | Get disk changed status |
| 17H | Set disk type |
| 18H | Set media type |
| 19H | Park heads |
| 1AH | Format ESDI drive |

## INT 14H

Interrupt 14H controls the serial COM (communications) ports attached to the computer. The computer system contains two COM ports, COM1 and COM2, unless you have a newer AT-style machine, in which the number of communications ports are extended to COM3 and COM4. Communications ports are normally controlled with software packages that allow data transfer through a modem and the telephone lines. The INT 14H instruction controls these ports, as illustrated in Table B–8.

## INT 15H

The INT 15H instruction controls many of the various I/O devices interfaced to the computer. It also allows access to protected mode operation and the extended memory system on an 80286–Pentium 4, but it is not recommended. Table B–9 lists the functions supported by INT 15H.

## INT 16H

The INT 16H instruction is used as a keyboard interrupt. This interrupt is accessed by DOS interrupt INT 21H, but it can be accessed directly. Table B–10 shows the functions performed by INT 16H.

## INT 17H

The INT 17H instruction accesses the parallel printer port, usually labeled LPT1 in most systems. Table B–11 lists the three functions available for the INT 17H instruction.

## DOS Low Memory Assignments

Table B–12 shows the low memory assignments (00000H–005FFH) for the DOS-based microprocessor system. This area of memory contains the interrupt vectors, BIOS data area, and the DOS/BIOS data.

**TABLE B–9** The I/O subsystem interrupt INT 15H.

| *AH* | *Function* |
|---|---|
| 00H | Cassette motor on |
| 01H | Cassette motor off |
| 02H | Read cassette |
| 03H | Write cassette |
| 0FH | Format ESDI periodic interrupt |
| 21H | Keyboard intercept |
| 80H | Device open |
| 81H | Device closed |
| 82H | Process termination |
| 83H | Event wait |
| 84H | Read joystick |
| 85H | System request key |
| 86H | Delay |
| 87H | Move extended block of memory |
| 88H | Get extended memory size |
| 89H | Enter protected mode |
| 90H | Device wait |
| 91H | Device power on self test (POST) |
| C0H | Get system environment |
| C1H | Get address of extended BIOS data area |
| C2H | Mouse pointer |
| C3H | Set watchdog timer |
| C4H | Programmable option select |

**TABLE B–10** Keyboard interrupt INT 16H.

| *AH* | *Function* |
|---|---|
| 00H | Read keyboard character |
| 01H | Get keyboard status |
| 02H | Get keyboard flags |
| 03H | Set repeat rate |
| 04H | Set keyboard click |
| 05H | Push character and scan code |

**TABLE B–11** Parallel printer interrupt INT 17H.

| *AH* | *Function* |
|---|---|
| 00H | Print character |
| 01H | Initialize printer |
| 02H | Get printer status |

**TABLE B–12** DOS low memory assignments.

| *Location* | *Purpose* |
|---|---|
| 00000H–002FFH | System interrupt vectors |
| 00300H–003FFH | System interrupt vectors, power on, and bootstrap area |
| 00400H–00407H | COM1–COM4 I/O port base addresses |
| 00408H–0040FH | LPT1–LPT4 I/O port base addresses |
| 00410H–00411H | Equipment flag word, returned in AX by INT 11H (refer to Figure B–8) |
| 00412H | Reserved |
| 00413H–00414H | Memory size in K byte (0–640K) |
| 00415H–00416H | Reserved |
| 00417H | Keyboard control byte<br>*Bit* *Purpose*<br>7 Insert locked<br>6 Caps locked<br>5 Numbers locked<br>4 Scroll locked<br>3 Alternate key pressed<br>2 Control key pressed<br>1 Left shift key pressed<br>0 Right shift key pressed |
| 00418H | Keyboard control byte<br>*Bit* *Purpose*<br>7 Insert key pressed<br>6 Caps lock key pressed<br>5 Numbers lock key pressed<br>4 Scroll lock key pressed<br>3 Pause key pressed<br>2 System request key pressed<br>1 Left alternate key pressed<br>0 Right control key pressed |
| 00419H | Alternate keyboard entry |
| 0041AH–0041BH | Keyboard buffer header pointer |
| 0041CH–0041DH | Keyboard buffer tail pointer |
| 0041EH–0043DH | 32-byte keyboard buffer area |
| 0043EH–00448H | Disk drive control area |
| 00449H–00466H | Video control area |
| 00467H–0046BH | Reserved |
| 0046CH–0046FH | Timer counter |
| 00470H | Timer overflow |
| 00471H | Break key state |
| 00472H–00473H | Reset flag |
| 00474H–00477H | Hard disk drive data area |
| 00478H–0047BH | LPT1–LPT4 time-out area |
| 0047CH–0047FH | COM1–COM4 time-out area |
| 0047CH–0047FH | COM1–COM4 time-out area |

*(continued on the next page)*

**TABLE B–12** *(continued)*

| *Location* | *Purpose* |
|---|---|
| 00480H–00481H | Keyboard buffer start offset pointer |
| 00482H–00483H | Keyboard buffer end offset pointer |
| 00484H–0048AH | Video control data area |
| 0048BH–00495H | Hard disk control area |
| 00496H | Keyboard mode, state, and type flag |
| 00497H | Keyboard LED flags |
| 00498H–00499H | Offset address of user wait complete flag |
| 0049AH–0049BH | Segment address of user wait complete flag |
| 0049CH–0049FH | User wait count |
| 004A0H | Wait active flag |
| 004A1H–004A7H | Reserved |
| 004A8H–004ABH | Pointer to video parameters |
| 004ACH–004EFH | Reserved |
| 004F0H–004FFH | Applications program communications area |
| 00500H | Print screen status |
| 00501H–00503H | Reserved |
| 00504H | Single drive mode status |
| 00505H–0050FH | Reserved |
| 00510H–00521H | Used by ROM BASIC |
| 00522H–0052FH | Used by DOS for disk initialization |
| 00530H–00533H | Used by the MODE command |
| 00534H–005FFH | Reserved |

# APPENDIX C

# Instruction Set Summary

The instruction set summary contains a complete alphabetical listing of the entire 8086–Pentium 4 instruction set and are not repeated in this Appendix. The SIMD instructions are listed after the main instruction set summary.

Each instruction entry lists the mnemonic opcode plus a brief description of the purpose of the instruction. Also listed is the binary machine language coding of each instruction and any other data required to form the instruction, such as the displacement or immediate data. Listed to the right of each binary machine language version of the instruction are the flag bits and any change that might occur for the instruction. The flags are described in the following manner: a blank indicates no effect or change, a ? indicates a change with an unpredictable outcome, a * indicates a change with a predictable outcome, a 1 indicates the flag is set, and a 0 indicates that the flag is cleared. If the flag bits ODITSZAPC are not illustrated with an instruction, the instruction does not modify any of these flags.

Before the instruction listing begins, some information about the bit settings in binary machine language versions of the instructions is presented. Table C–1 lists the modifier bits, coded as oo in the instruction listings.

Table C–2 lists the memory-addressing modes available using a register field coding of mmm. This table applies to all versions of the microprocessor, as long as the operating mode is 16 bits.

Table C–3 lists the register selections provided by the rrr field in an instruction. This table includes the register selections for 8-, 16-, and 32-bit registers.

Table C–4 lists the segment register bit assignment (rrr) found with the MOV, PUSH, and POP instructions.

**TABLE C–1** The modifier bits, coded as oo in the instruction listing.

| *oo* | *Function* |
|---|---|
| 00 | If mmm = 110, then a displacement follows the opcode; otherwise, no displacement is used |
| 01 | An 8-bit signed displacement follows the opcode |
| 10 | A 16-bit signed displacement follows the opcode (unless it is a 32-bit displacement) |
| 11 | mmm specifies a register instead of an addressing mode |

**TABLE C–2** The 16-bit register/memory (mmm) field description.

| *mmm* | *Function* |
|---|---|
| 000 | DS:[BX+SI] |
| 001 | DS:[BX+DI] |
| 010 | SS:[BP+SI] |
| 011 | SS:[BP+DI] |
| 100 | DS:[SI] |
| 101 | DS:[DI] |
| 110 | SS:[BP] |
| 111 | DS:[BX] |

**TABLE C–3** The register field (rrr) assignment.

| *rrr* | *W=0* | *W=1 (16-bit)* | *W=1 (32-bit)* |
|---|---|---|---|
| 000 | AL | AX | EAX |
| 001 | CL | CX | ECX |
| 010 | DL | DX | EDX |
| 011 | BL | BX | EBX |
| 100 | AH | SP | ESP |
| 101 | CH | BP | EBP |
| 110 | DH | SI | ESI |
| 111 | BH | DI | EDI |

**TABLE C–4** Register field assignments (rrr) for the segment registers.

| *rrr* | *Segment Register* |
|---|---|
| 000 | ES |
| 001 | CS |
| 010 | SS |
| 011 | DS |
| 100 | FS |
| 101 | GS |

When the 80386–Pentium 4 are used, some of the definitions provided in Tables C–1 through C–3 change. See Tables C–5 and C–6 for these changes as they apply to the 80386–Pentium 4 microprocessors.

In order to use the scaled index addressing modes listed in Table C–6, code oo and mmm in the second byte of the opcode. The scaled index byte is usually the third byte and contains three fields. The leftmost two bits determine the scaling factor (00 = X1, 01 = X2, 10 = X4, or 11 = X8). The next three bits toward the right contain the scaled index register number (this is obtained from Table C–5). The rightmost three bits are from the rrr field listed in Table C–6. For example, the MOV AL,[EBX+2*ECX] instruction has a scaled index byte of 01 001 011, where 01 = X2, the 001 = ECX, and 011 = EBX.

Some instructions are prefixed to change the default segment or to override the instruction mode. Table C–7 lists the segment and instruction mode override prefixes with append at the beginning of an instruction if they are used to form the instruction. For example, the MOV AL,ES:[BX] instruction used the extra segment because of the override prefix ES:.

In the 8086 and 8088 microprocessors, the effective address calculation required additional clocks that are added to the times in the instruction set summary. These additional times are listed in Table C–8. No such times are added to the 80286–Pentium 4. Note that the instruction set summary does not include clock times for the Pentium Pro through the Pentium 4. Intel has not released these times and has decided that the RDTSC instruction can be used to have the microprocessor count the number of clocks required for a given application. Even though the timings do not appear for these new microprocessors, they are very similar to the Pentium, which can be used as a guide.

**TABLE C–5** Index register specified with rrr for the advanced addressing mode found in the 80386–Pentium 4 microprocessors.

| *rrr* | *Index Register* |
|---|---|
| 000 | DS:[EAX] |
| 001 | DS:[ECX] |
| 010 | DS:[EDX] |
| 011 | DS:[EBX] |
| 100 | No index (see Table C–6) |
| 101 | SS:[EBP] |
| 110 | DS:[ESI] |
| 111 | DS:[EDI] |

**TABLE C–6** Possible combinations of oo, mmm, and rrr for the 80386–Pentium 4 microprocessors using 32-bit addressing.

| *oo* | *mmm* | *rrr (base in scaled index byte)* | *Addressing Mode* |
|---|---|---|---|
| 00 | 000 | — | DS:[EAX] |
| 00 | 001 | — | DS:[ECX] |
| 00 | 010 | — | DS:[EDX] |
| 00 | 011 | — | DS:[EBX] |
| 00 | 100 | 000 | DS:[EAX+scaled index] |
| 00 | 100 | 001 | DS:[ECX+scaled index] |
| 00 | 100 | 010 | DS:[EDX+scaled index] |
| 00 | 100 | 011 | DS:[EBX+scaled index] |
| 00 | 100 | 100 | SS:[ESP+scaled index] |
| 00 | 100 | 101 | DS:[disp32+scaled index] |
| 00 | 100 | 110 | DS:[ESI+scaled index] |
| 00 | 100 | 111 | DS:[EDI+scaled index] |
| 00 | 101 | — | DS:disp32 |
| 00 | 110 | — | DS:[ESI] |
| 00 | 111 | — | DS:[EDI] |
| 01 | 000 | — | DS:[EAX+disp8] |
| 01 | 001 | — | DS:[ECX+disp8] |
| 01 | 010 | — | DS:[EDX+disp8] |
| 01 | 011 | — | DS:[EBX+disp8] |
| 01 | 100 | 000 | DS:[EAX+scaled index+disp8] |
| 01 | 100 | 001 | DS:[ECX+scaled index+disp8] |
| 01 | 100 | 010 | DS:[EDX+scaled index+disp8] |
| 01 | 100 | 011 | DS:[EBX+scaled index+disp8] |
| 01 | 100 | 100 | SS:[ESP+scaled index+disp8] |
| 01 | 100 | 101 | SS:[EBP+scaled index+disp8] |
| 01 | 100 | 110 | DS:[ESI+scaled index+disp8] |
| 01 | 100 | 111 | DS:[EDI+scaled index+disp8] |
| 01 | 101 | — | SS:[EBP+disp8] |
| 01 | 110 | — | DS:[ESI+disp8] |
| 01 | 111 | — | DS:[EDI+disp8] |
| 10 | 000 | — | DS:[EAX+disp32] |
| 10 | 001 | — | DS:[ECX+disp32] |
| 10 | 010 | — | DS:[EDX+disp32] |
| 10 | 011 | — | DS:[EBX+disp32] |
| 10 | 100 | 000 | DS:[EAX+scaled index+disp32] |
| 10 | 100 | 001 | DS:[ECX+scaled index+disp32] |
| 10 | 100 | 010 | DS:[EDX+scaled index+disp32] |
| 10 | 100 | 011 | DS:[EBX+scaled index+disp32] |
| 10 | 100 | 100 | SS:[ESP+scaled index+disp32] |
| 10 | 100 | 101 | SS:[EBP+scaled index+disp32] |
| 10 | 100 | 110 | DS:[ESI+scaled index+disp32] |
| 10 | 100 | 111 | DS:[EDI+scaled index+disp32] |
| 10 | 101 | — | SS:[EBP+disp32] |
| 10 | 110 | — | DS:[ESI+disp32] |
| 10 | 111 | — | DS:[EDI+disp32] |

*Notes:* disp8 = 8-bit displacement and disp32 = 32-bit displacement.

TABLE C–7 Override prefixes.

| Prefix Byte | Purpose |
|---|---|
| 26H | ES: segment override prefix |
| 2EH | CS: segment override prefix |
| 36H | SS: segment override prefix |
| 3EH | DS: segment override prefix |
| 64H | FS: segment override prefix |
| 65H | GS: segment override prefix |
| 66H | Operand size instruction mode override |
| 67H | Register size instruction mode override |

TABLE C–8 Effective address calculations for the 8086 and 8088 microprocessors.

| Type | Clocks | Example Instruction |
|---|---|---|
| Base or index | 5 | MOV CL,[DI] |
| Displacement | 3 | MOV AL,DATA1 |
| Base plus index | 7 | MOV AL,[BP+SI] |
| Displacement plus base or index | 9 | MOV DH,[DI+20H] |
| Base plus index plus displacement | 11 | MOV CL,[BX+DI+2] |
| Segment override | ea + 2 | MOV AL,ED:[DI] |

## INSTRUCTION SET SUMMARY (pp. 624–707)

**AAA** ASCII adjust AL after addition

00110111

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? |  |  |  | ? | ? | * | ? | * |

| Example | Microprocessor | Clocks |
|---|---|---|
| **AAA** | 8086 | 8 |
|  | 8088 | 8 |
|  | 80286 | 3 |
|  | 80386 | 4 |
|  | 80486 | 3 |
|  | Pentium | 3 |

**AAD** ASCII adjust before division

11010101 00001010

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? |  |  |  | * | * | ? | * | ? |

| Example | Microprocessor | Clocks |
|---|---|---|
| **AAD** | 8086 | 60 |
|  | 8088 | 60 |
|  | 80286 | 14 |
|  | 80386 | 19 |
|  | 80486 | 14 |
|  | Pentium | 10 |

**AAM** ASCII adjust AX after multiplication

11010100 00001010

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? |  |  |  | * | * | ? | * | ? |

| Example | Microprocessor | Clocks |
|---|---|---|
| **AAM** | 8086 | 83 |
|  | 8088 | 83 |
|  | 80286 | 16 |
|  | 80386 | 17 |
|  | 80486 | 15 |
|  | Pentium | 18 |

*Note:* The instructions in bold are the ones which are 8086 instructions. This is apart from the right side shaded block

| AAS | ASCII adjust AL after subtraction | [illegible] | [illegible] |
|---|---|---|---|

00111111

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? | | | | ? | ? | * | ? | * |

| Example | Microprocessor | Clocks |
|---|---|---|
| **AAS** | 8086 | 8 |
| | 8088 | 8 |
| | 80286 | 3 |
| | 80386 | 4 |
| | 80486 | 3 |
| | Pentium | 3 |

| ADC | Addition with carry | [illegible] | [illegible] |
|---|---|---|---|

000100dw oorrrmmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| * | | | | * | * | * | * | * |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **ADC reg,reg** | **ADC AX,BX**<br>**ADC AL,BL**<br>ADC EAX,EBX<br>ADC CX,SI<br>ADC ESI,EDI | [illegible] | [illegible] |
| | | [illegible] | 3 |
| | | 80286 | 3 |
| | | 80386 | 3 |
| | | 80486 | 1 |
| | | Pentium | 1 or 3 |
| **ADC mem,reg** | **ADC DATAY,AL**<br>**ADC LIST,SI**<br>**ADC DATA2[DI],CL**<br>ADC [EAX],BL<br>ADC [EBX+2*ECX],EDX | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 7 |
| | | 80386 | 7 |
| | | 80486 | 3 |
| | | Pentium | 1 or 3 |

| | | | |
|---|---|---|---|
| **ADC reg,mem** | **ADC BL,DATA1**<br>**ADC SI,LIST1**<br>**ADC CL,DATA2[SI]**<br>ADC CX,[ESI]<br>ADC ESI,[2*ECX] | [illegible] | [illegible] |
| | | 80286 | 7 |
| | | 80386 | 6 |
| | | 80486 | 2 |
| | | Pentium | 1 or 2 |

100000sw oo010mmm disp data

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **ADC reg,imm** | **ADC CX,3**<br>**ADC DI,1AH**<br>**ADC DL,34H**<br>ADC EAX,12345<br>ADC CX,1234H | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 3 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 or 3 |
| **ADC mem,imm** | **ADC DATA4,33**<br>**ADC LIST,'A'**<br>**ADC DATA3[DI],2**<br>ADC BYTE PTR[EBX],3<br>ADC WORD PTR[DI],669H | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 7 |
| | | 80386 | 7 |
| | | 80486 | 3 |
| | | Pentium | 1 or 3 |
| **ADC acc,imm** | **ADC AX,3**<br>**ADC AL,1AH**<br>**ADC AH,34H**<br>ADC EAX,2<br>ADC AL,'Z' | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 3 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 |

000000dw oorrrmmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| * |  |  |  | * | * | * | * | * |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **ADD reg,reg** | **ADD AX,BX**<br>**ADD AL,BL**<br>**ADD EAX,EBX**<br>ADD CX,SI<br>ADD ESI,EDI |  |  |
|  |  | 80286 | 2 |
|  |  | 80386 | 2 |
|  |  | 80486 | 1 |
|  |  | Pentium | 1 or 3 |
| **ADD mem,reg** | **ADD DATAY,AL**<br>**ADD LIST,SI**<br>**ADD DATA6[DI],CL**<br>ADD [EAX],CL<br>ADD [EDX+4*ECX],EBX |  |  |
|  |  | 80286 | 7 |
|  |  | 80386 | 7 |
|  |  | 80486 | 3 |
|  |  | Pentium | 1 or 3 |
| **ADD reg,mem** | **ADD BL,DATA2**<br>**ADD SI,LIST3**<br>**ADD CL,DATA2[DI]**<br>ADD CX,[EDI]<br>ADD ESI,[ECX+200H] |  |  |
|  |  | 80286 | 7 |
|  |  | 80386 | 6 |
|  |  | 80486 | 2 |
|  |  | Pentium | 1 or 2 |

100000sw oo000mmm disp data

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **ADD reg,imm** | **ADD CX,3**<br>**ADD DI,1AH**<br>**ADD DL,34H**<br>ADD EDX,1345H<br>ADD CX,1834H |  |  |
|  |  | 80286 | 3 |
|  |  | 80386 | 2 |
|  |  | 80486 | 1 |
|  |  | Pentium | 1 or 3 |

| | | | |
|---|---|---|---|
| **ADD mem,imm** | **ADD DATA4,33**<br>**ADD LIST,'A'**<br>ADD DATA3[DI],2<br>ADD BYTE PTR[EBX],3<br>ADD WORD PTR[DI],669H | [redacted] | [redacted] |
| | | [redacted] | [redacted] |
| | | 80286 | 7 |
| | | 80386 | 7 |
| | | 80486 | 3 |
| | | Pentium | 1 or 3 |
| **ADD acc,imm** | **ADD AX,3**<br>**ADD AL,1AH**<br>**ADD AH,34H**<br>ADD EAX,2<br>ADD AL,'Z' | [redacted] | [redacted] |
| | | [redacted] | [redacted] |
| | | 80286 | 3 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 |

[redacted]

001000dw oorrrmmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | * | * | ? | * | 0 |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **AND reg,reg** | **AND CX,BX**<br>**AND DL,BL**<br>AND ECX,EBX<br>AND BP,SI<br>AND EDX,EDI | [redacted] | [redacted] |
| | | [redacted] | [redacted] |
| | | 80286 | 2 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 or 3 |
| **AND mem,reg** | **AND BIT,AL**<br>**AND LIST,DI**<br>AND DATAZ[BX],CL<br>AND [EAX],BL<br>AND [ESI+4*ECX],EDX | [redacted] | [redacted] |
| | | [redacted] | [redacted] |
| | | 80286 | 7 |
| | | 80386 | 7 |
| | | 80486 | 3 |
| | | Pentium | 1 or 3 |

| | | | |
|---|---|---|---|
| **AND reg,mem** | **AND BL,DATAW**<br>AND SI,LIST<br>AND CL,DATAQ[SI]<br>AND CX,[EAX]<br>AND ESI,[ECX+43H] | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 7 |
| | | 80386 | 6 |
| | | 80486 | 2 |
| | | Pentium | 1 or 2 |

100000sw oo100mmm disp data

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **AND reg,imm** | **AND BP,1**<br>**AND DI,10H**<br>AND DL,34H<br>AND EBP,1345H<br>AND SP,1834H | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 3 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 or 3 |
| **AND mem,imm** | **AND DATA4,33**<br>**AND LIST,'A'**<br>AND DATA3[DI],2<br>AND BYTE PTR[EBX],3<br>AND DWORD PTR[DI],66H | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 7 |
| | | 80386 | 7 |
| | | 80486 | 3 |
| | | Pentium | 1 or 3 |
| **AND acc,imm** | **AND AX,3**<br>**AND AL,1AH**<br>AND AH,34H<br>AND EAX,2<br>AND AL,'r' | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 3 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 |

**ARPL** Adjust requested privilege level

01100011 oorrrmmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  | * |  |  |  |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| ARPL reg,reg | ARPL AX,BX<br>ARPL BX,SI<br>ARPL AX,DX<br>ARPL BX,AX<br>ARPL SI,DI | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | 10 |
|  |  | 80386 | 20 |
|  |  | 80486 | 9 |
|  |  | Pentium | 7 |
| ARPL mem,reg | ARPL DATAY,AX<br>ARPL LIST,DI<br>ARPL DATA3[DI],CX<br>ARPL [EBX],AX<br>ARPL [EDX+4*ECX],BP | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | 11 |
|  |  | 80386 | 21 |
|  |  | 80486 | 9 |
|  |  | Pentium | 7 |

**BOUND** Check array against boundary

01100010 oorrrmmm disp

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BOUND reg,mem<br>8088 | BOUND AX,BETS<br>BOUND BP,LISTG<br>BOUND CX,DATAX<br>BOUND BX,[DI]<br>BOUND SI,[BX+2] | 8086 | — |
|  |  | 80286 | 13 |
|  |  | 80386 | 10 |
|  |  | 80486 | 7 |
|  |  | Pentium | 8 |

BSF Bit scan forward

00001111 10111100 oorrrmmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? |  |  |  | ? | * | ? | ? | ? |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BSF reg,reg | BSF AX,BX<br>BSF BX,SI<br>BSF EAX,EDX<br>BSF EBX,EAX<br>BSF SI,DI | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | — |
|  |  | 80386 | 10 + 3n |
|  |  | 80486 | 6–42 |
|  |  | Pentium | 6–42 |
| BSF reg,mem | BSF AX,DATAY<br>BSF SI,LIST<br>BSF CX,DATA3[DI]<br>BSF EAX,[EBX]<br>BSF EBP,[EDX+4*ECX] | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | — |
|  |  | 80386 | 10 + 3n |
|  |  | 80486 | 7–43 |
|  |  | Pentium | 6–43 |

BSR Bit scan reverse

00001111 10111101 oorrrmmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? |  |  |  | ? | * | ? | ? | ? |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BSR reg,reg | BSR AX,BX<br>BSR BX,SI<br>BSR EAX,EDX<br>BSR EBX,EAX<br>BSR SI,DI | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | — |
|  |  | 80386 | 10 + 3n |
|  |  | 80486 | 6–103 |
|  |  | Pentium | 7–71 |

| | | | |
|---|---|---|---|
| BSR reg,mem | BSR AX,DATAY<br>BSR SI,LIST<br>BSR CX,DATA3[DI]<br>BSR EAX,[EBX]<br>BSR EBP,[EDX+4*ECX] | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 10 + 3n |
| | | 80486 | 7–104 |
| | | Pentium | 7–72 |

BSWAP Byte swap

00001111 11001rrr

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BSWAP reg32 | BSWAP EAX<br>BSWAP EBX<br>BSWAP EDX<br>BSWAP ECX<br>BSWAP ESI | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | — |
| | | 80486 | 1 |
| | | Pentium | 1 |

BT Bit test

00001111 10111010 oo100mmm disp data

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | * |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BT reg,imm8 | BT AX,2<br>BT CX,4<br>BT BP,10H<br>BT CX,8<br>BT BX,2 | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 3 |
| | | 80486 | 3 |
| | | Pentium | 4 |

| | | | |
|---|---|---|---|
| BT mem,imm8 | BT DATA1,2<br>BT LIST,2<br>BT DATA2[DI],3<br>BT [EAX],1<br>BT FROG,6 | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 6 |
| | | 80486 | 3 |
| | | Pentium | 4 |

00001111 10100011 disp

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BT reg,reg | BT AX,CX<br>BT CX,DX<br>BT BP,AX<br>BT SI,CX<br>BT EAX,EBX | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 3 |
| | | 80486 | 3 |
| | | Pentium | 4 or 9 |
| BT mem,reg | BT DATA4,AX<br>BT LIST,BX<br>BT DATA3[DI],CX<br>BT [EBX],DX<br>BT [DI],DI | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 12 |
| | | 80486 | 8 |
| | | Pentium | 4 or 9 |

BTC Bit test and complement

00001111 10111010 oo111mmm disp data

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  | * |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BTC reg,imm8 | BTC AX,2<br>BTC CX,4<br>BTC BP,10H<br>BTC CX,8<br>BTC BX,2 | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | — |
|  |  | 80386 | 6 |
|  |  | 80486 | 6 |
|  |  | Pentium | 7 or 8 |
| BTC mem,imm8 | BTC DATA1,2<br>BTC LIST,2<br>BTC DATA2[DI],3<br>BTC [EAX],1<br>BTC FROG,6 | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | — |
|  |  | 80386 | 7 or 8 |
|  |  | 80486 | 8 |
|  |  | Pentium | 8 |

00001111 10111011 disp

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BTC reg,reg | BTC AX,CX<br>BTC CX,DX<br>BTC BP,AX<br>BTC SI,CX<br>BTC EAX,EBX | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | — |
|  |  | 80386 | 6 |
|  |  | 80486 | 6 |
|  |  | Pentium | 7 or 13 |
| BTC mem,reg | BTC DATA4,AX<br>BTC LIST,BX<br>BTC DATA3[DI],CX<br>BTC [EBX],DX<br>BTC [DI],DI | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | — |
|  |  | 80386 | 13 |
|  |  | 80486 | 13 |
|  |  | Pentium | 7 or 13 |

| | | | |
|---|---|---|---|
| BTR | Bit test and reset | | |
| 00001111 10111010 oo110mmm disp data | | O D I T | S Z A P C<br>        * |
| Format | Examples | Microprocessor | Clocks |
| BTR reg,imm8 | BTR AX,2<br>BTR CX,4<br>BTR BP,10H<br>BTR CX,8<br>BTR BX,2 | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 6 |
| | | 80486 | 6 |
| | | Pentium | 7 or 8 |
| BTR mem,imm8 | BTR DATA1,2<br>BTR LIST,2<br>BTR DATA2[DI],3<br>BTR [EAX],1<br>BTR FROG,6 | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 8 |
| | | 80486 | 8 |
| | | Pentium | 7 or 8 |
| 00001111 10110011 disp | | | |
| Format | Examples | Microprocessor | Clocks |
| BTR reg,reg | BTR AX,CX<br>BTR CX,DX<br>BTR BP,AX<br>BTR SI,CX<br>BTR EAX,EBX | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 6 |
| | | 80486 | 6 |
| | | Pentium | 7 or 13 |
| BTR mem,reg | BTR DATA4,AX<br>BTR LIST,BX<br>BTR DATA3[DI],CX<br>BTR [EBX],DX<br>BTR [DI],DI<br>BTC [DI],DI | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 13 |
| | | 80486 | 13 |
| | | Pentium | 7 or 13 |

BTS Bit test and set

00001111 10111010 oo101mmm disp data

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | * |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BTS reg,imm8 | BTS AX,2<br>BTS CX,4<br>BTS BP,10H<br>BTS CX,8<br>BTS BX,2 | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 6 |
| | | 80486 | 6 |
| | | Pentium | 7 or 8 |
| BTS mem,imm8 | BTS DATA1,2<br>BTS LIST,2<br>BTS DATA2[DI],3<br>BTS [EAX],1<br>BTS FROG,6 | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 8 |
| | | 80486 | 8 |
| | | Pentium | 7 or 8 |

00001111 10101011 disp

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| BTS reg,reg | BTS AX,CX<br>BTS CX,DX<br>BTS BP,AX<br>BTS SI,CX<br>BTS EAX,EBX | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 6 |
| | | 80486 | 6 |
| | | Pentium | 7 or 13 |
| BTS mem,reg | BTS DATA4,AX<br>BTS LIST,BX<br>BTS DATA3[DI],CX<br>BTS [EBX],DX<br>BTS [DI],DI | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | 13 |
| | | 80486 | 13 |
| | | Pentium | 7 or 13 |

| | | | |
|---|---|---|---|
| 11101000 disp | | | |
| Format | Examples | Microprocessor | Clocks |
| **CALL label**<br>**(near)** | CALL FOR_FUN<br>CALL HOME<br>CALL ET<br>CALL WAITING<br>CALL SOMEONE | | |
| | | 80286 | 7 |
| | | 80386 | 3 |
| | | 80486 | 3 |
| | | Pentium | 1 |
| 10011010 disp | | | |
| Format | Examples | Microprocessor | Clocks |
| **CALL label**<br>**(far)** | **CALL FAR PTR DATES**<br>CALL WHAT<br>CALL WHERE<br>CALL FARCE<br>CALL WHOM | | |
| | | 80286 | 13 |
| | | 80386 | 17 |
| | | 80486 | 18 |
| | | Pentium | 4 |
| 11111111 oo010mmm | | | |
| Format | Examples | Microprocessor | Clocks |
| **CALL reg**<br>**(near)** | **CALL AX**<br>**CALL BX**<br>CALL CX<br>CALL DI<br>CALL SI | | |
| | | 80286 | 7 |
| | | 80386 | 7 |
| | | 80486 | 5 |
| | | Pentium | 2 |

| | | | |
|---|---|---|---|
| **CALL mem**<br>**(near)** | **CALL ADDRESS**<br>**CALL NEAR PTR [DI]**<br>CALL DATA1<br>CALL FROG<br>CALL ME_NOW | [redacted] | [redacted] |
| | | [redacted] | [redacted] |
| | | 80286 | 11 |
| | | 80386 | 10 |
| | | 80486 | 5 |
| | | Pentium | 2 |

11111111 oo011mmm

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **CALL mem**<br>**(far)** | CALL FAR_LIST[SI]<br>CALL FROM_HERE<br>CALL TO_THERE<br>CALL SIXX<br>CALL OCT | [redacted] | [redacted] |
| | | [redacted] | [redacted] |
| | | 80286 | 7 |
| | | 80386 | 7 |
| | | 80486 | 5 |
| | | Pentium | 2 |

[redacted]

10011000

| Example | Microprocessor | Clocks |
|---|---|---|
| **CBW** | [redacted] | [redacted] |
| | [redacted] | [redacted] |
| | 80286 | 2 |
| | 80386 | 3 |
| | 80486 | 3 |
| | Pentium | 3 |

CDQ Convert doubleword to quadword
(EAX | EDX:EAX)

11010100 00001010

| Example | Microprocessor | Clocks |
|---|---|---|
| CDQ | 8086 | — |
| | 8088 | — |
| | 80286 | — |
| | 80386 | 2 |
| | 80486 | 2 |
| | Pentium | 2 |

[redacted]

11111000

O D I T S Z A P C
0

| Example | Microprocessor | Clocks |
|---|---|---|
| CLC | [redacted] | [redacted] |
| | [redacted] | [redacted] |
| | 80286 | 2 |
| | 80386 | 2 |
| | 80486 | 2 |
| | Pentium | 2 |

[redacted]

11111100

O D I T S Z A P C
0

| Example | Microprocessor | Clocks |
|---|---|---|
| CLD | [redacted] | [redacted] |
| | [redacted] | [redacted] |
| | 80286 | 2 |
| | 80386 | 2 |
| | 80486 | 2 |
| | Pentium | 2 |

[illegible]

11111010

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| | | 0 | | | | | | |

| Example | Microprocessor | Clocks |
|---|---|---|
| **CLI** | 8086 | 2 |
| | 8088 | 2 |
| | 80286 | 3 |
| | 80386 | 3 |
| | 80486 | 5 |
| | Pentium | 7 |

CLTS Clear task switched flag (CR0)

00001111 00000110

| Example | Microprocessor | Clocks |
|---|---|---|
| CLTS | 8086 | — |
| | 8088 | — |
| | 80286 | 2 |
| | 80386 | 5 |
| | 80486 | 7 |
| | Pentium | 10 |

[illegible]

10011000

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | * |

| Example | Microprocessor | Clocks |
|---|---|---|
| **CMC** | 8086 | 2 |
| | 8088 | 2 |
| | 80286 | 2 |
| | 80386 | 2 |
| | 80486 | 2 |
| | Pentium | 2 |

| CMOVcondition Conditional move | | | |
|---|---|---|---|
| 00001111 0100cccc oorrrmmm | | | |
| Format | Examples | Microprocessor | Clocks |
| CMOVcc reg,mem | CMOVNZ AX,FROG<br>CMOVC EAX,[EDI]<br>CMOVNC BX,DATA1<br>CMOVP EBX,WAITING<br>CMOVNE DI,[SI] | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | — |
| | | 80486 | — |
| | | Pentium | — |

| Condition Codes | Mnemonic | Flag | Description |
|---|---|---|---|
| 0000 | CMOVO | O = 1 | Move if overflow |
| 0001 | CMOVNO | O = 0 | Move if no overflow |
| 0010 | CMOVB | C = 1 | Move if below |
| 0011 | CMOVAE | C = 0 | Move if above or equal |
| 0100 | CMOVE | Z = 1 | Move if equal/zero |
| 0101 | CMOVNE | Z = 0 | Move if not equal/zero |
| 0110 | CMOVBE | C = 1 + Z = 1 | Move if below or equal |
| 0111 | CMOVA | C = 0 • Z = 0 | Move if above |
| 1000 | CMOVS | S = 1 | Move if sign |
| 1001 | CMOVNS | S = 0 | Move if no sign |
| 1010 | CMOVP | P = 1 | Move if parity |
| 1011 | CMOVNP | P = 0 | Move if no parity |
| 1100 | CMOVL | S • O | Move if less than |
| 1101 | CMOVGE | S = 0 | Move if greater than or equal |
| 1110 | CMOVLE | Z = 1 + S •σ O | Move if less than or equal |
| 1111 | CMOVG | Z = 0 + S = O | Move if greater than |

| CMP Compare | | | |
|---|---|---|---|
| 001110dw oorrrmmm disp | | O D I T S Z A P C<br>* * * * * * | |
| Format | Examples | Microprocessor | Clocks |
| **CMP reg,reg** | **CMP AX,BX**<br>**CMP AL,BL**<br>CMP EAX,EBX<br>CMP CX,SI<br>CMP ESI,EDI | 8086 | 3 |
| | | 8088 | 3 |
| | | 80286 | 2 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 or 2 |

| | | | |
|---|---|---|---|
| **CMP mem,reg** | **CMP DATAY,AL**<br>**CMP LIST,SI**<br>CMP DATA6[DI],CL<br>CMP [EAX],CL<br>CMP [EDX+4*ECX],EBX | | |
| | | 80286 | 7 |
| | | 80386 | 5 |
| | | 80486 | 2 |
| | | Pentium | 1 or 2 |
| **CMP reg,mem** | **CMP BL,DATA2**<br>**CMP SI,LIST3**<br>CMP CL,DATA2[DI]<br>CMP CX,[EDI]<br>CMP ESI,[ECX+200H] | | |
| | | 80286 | 6 |
| | | 80386 | 6 |
| | | 80486 | 2 |
| | | Pentium | 1 or 2 |
| 100000sw oo111mmm disp data | | | |
| Format | Examples | Microprocessor | Clocks |
| **CMP reg,imm** | **CMP CX,3**<br>**CMP DI,1AH**<br>**CMP DL,34H**<br>CMP EDX,1345H<br>CMP CX,1834H | | |
| | | 80286 | 3 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 or 2 |
| **CMP mem,imm** | **CMP DATAS,3**<br>**CMP BYTE PTR[EDI],1AH**<br>CMP DADDY,34H<br>CMP LIST,'A'<br>CMP TOAD,1834H | | |
| | | 80286 | 6 |
| | | 80386 | 5 |
| | | 80486 | 2 |
| | | Pentium | 1 or 2 |

0001111w data

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **CMP acc,imm** | **CMP AX,3**<br>**CMP AL,1AH**<br>CMP AH,34H<br>CMP EAX,1345H<br>CMP AL,'Y' | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 3 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 |

[illegible]

1010011w

O D I T S Z A P C
*       * * * * *

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **CMPSB**<br>**CMPSW**<br>CMPSD | **CMPSB**<br>**CMPSW**<br>CMPSD<br>CMPSB DATA1,DATA2<br>REPE CMPSB<br>REPNE CMPSW | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 8 |
| | | 80386 | 10 |
| | | 80486 | 8 |
| | | Pentium | 5 |

CMPXCHG Compare and exchange

00001111 1011000w 11rrrrrr

O D I T S Z A P C
*       * * * * *

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| CMPXCHG reg,reg | CMPXCHG EAX,EBX<br>CMPXCHG ECX,EDX | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | — |
| | | 80486 | 6 |
| | | Pentium | 6 |

0001111w data

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| CMPXCHG mem,reg | CMPXCHG DATAD,EAX<br>CMPXCHG DATA2,EDI | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | — |
| | | 80486 | 7 |
| | | Pentium | 6 |

**CMPXCHG8B** Compare and exchange 8 bytes

00001111 11000111 oorrrmmm

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| | | | | | * | | | |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| CMPXCHG8B mem64 | CMPXCHG8B DATA3 | 8086 | — |
| | | 8088 | — |
| | | 80286 | — |
| | | 80386 | — |
| | | 80486 | — |
| | | Pentium | 10 |

**CPUID** CPU identification code

00001111 10100010

| Example | Microprocessor | Clocks |
|---|---|---|
| CPUID | 8086 | — |
| | 8088 | — |
| | 80286 | — |
| | 80386 | — |
| | 80486 | — |
| | Pentium | 14 |

[illegible]

10011000

| Example | Microprocessor | Clocks |
|---|---|---|
| **CWD** | [illegible] | [illegible] |
| | [illegible] | [illegible] |
| | 80286 | 2 |
| | 80386 | 2 |
| | 80486 | 3 |
| | Pentium | 2 |

CWDE Convert word to extended doubleword
(AX | EAX)

10011000

| Example | Microprocessor | Clocks |
|---|---|---|
| CWDE | 8086 | — |
| | 8088 | — |
| | 80286 | — |
| | 80386 | 3 |
| | 80486 | 3 |
| | Pentium | 3 |

[illegible]

00100111

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? | | | | * | * | * | * | * |

| Example | Microprocessor | Clocks |
|---|---|---|
| **DAA** | [illegible] | [illegible] |
| | [illegible] | [illegible] |
| | 80286 | 3 |
| | 80386 | 4 |
| | 80486 | 2 |
| | Pentium | 3 |

[illegible]

| 00101111 | O D I T S Z A P C<br>? * * * * * | |
|---|---|---|
| Example | Microprocessor | Clocks |
| **DAS** | 8086 | 4 |
| | 8088 | 4 |
| | 80286 | 3 |
| | 80386 | 4 |
| | 80486 | 2 |
| | Pentium | 3 |

[illegible]

| 1111111w oo001mmm disp | | O D I T S Z A P C<br>* * * * * | |
|---|---|---|---|
| Format | Examples | Microprocessor | Clocks |
| **DEC reg8** | **DEC BL**<br>**DEC BH**<br>**DEC CL**<br>**DEC DH**<br>**DEC AH** | 8086 | 3 |
| | | 8088 | 3 |
| | | 80286 | 2 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 or 3 |
| **DEC mem** | **DEC DATAY**<br>**DEC LIST**<br>DEC DATA6[DI]<br>DEC BYTE PTR [BX]<br>DEC WORD PTR [EBX] | [illegible] | [illegible] |
| | | 8088 | [illegible] |
| | | 80286 | 7 |
| | | 80386 | 6 |
| | | 80486 | 3 |
| | | Pentium | 1 or 3 |

| 01001rrr<br>Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **DEC reg16**<br>**DEC reg32** | **DEC CX**<br>**DEC DI**<br>DEC EDX<br>DEC ECX<br>DEC BP | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 2 |
| | | 80386 | 2 |
| | | 80486 | 1 |
| | | Pentium | 1 |

**DIV** [illegible] **Division**

1111011w oo110mmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? | | | | ? | ? | ? | ? | ? |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **DIV reg** | **DIV BL**<br>**DIV BH**<br>DIV ECX<br>**DIV DH**<br>**DIV CX** | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 22 |
| | | 80386 | 38 |
| | | 80486 | 40 |
| | | Pentium | 17–41 |
| **DIV mem** | **DIV DATAY**<br>**DIV LIST**<br>DIV DATA6[DI]<br>DIV BYTE PTR [BX]<br>DIV WORD PTR [EBX] | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 25 |
| | | 80386 | 41 |
| | | 80486 | 40 |
| | | Pentium | 17–41 |

| ENTER | Create a stack frame | | |
|---|---|---|---|
| 11001000 data<br>Format | Examples | Microprocessor | Clocks |
| ENTER imm,0 | ENTER 4,0<br>ENTER 8,0<br>ENTER 100,0<br>ENTER 200,0<br>ENTER 1024,0 | 8086 | — |
| | | 8088 | — |
| | | 80286 | 11 |
| | | 80386 | 10 |
| | | 80486 | 14 |
| | | Pentium | 11 |
| ENTER imm,1 | ENTER 4,1<br>ENTER 10,1 | 8086 | — |
| | | 8088 | — |
| | | 80286 | 12 |
| | | 80386 | 15 |
| | | 80486 | 17 |
| | | Pentium | 15 |
| ENTER imm,imm | ENTER 3,6<br>ENTER 100,3 | 8086 | — |
| | | 8088 | — |
| | | 80286 | 12 |
| | | 80386 | 15 |
| | | 80486 | 17 |
| | | Pentium | 15 + 2n |
| ESC | Escape (obsolete–see coprocessor) | | |

[illegible]

11110100

| Example | Microprocessor | Clocks |
|---|---|---|
| **HLT** | [illegible] | [illegible] |
| | [illegible] | [illegible] |
| | 80286 | 2 |
| | 80386 | 5 |
| | 80486 | 4 |
| | Pentium | varies |

[illegible] division

1111011w oo111mmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| ? | | | | ? | ? | ? | ? | ? |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **IDIV reg** | **IDIV BL**<br>**IDIV BH**<br>IDIV ECX<br>**IDIV DH**<br>**IDIV CX** | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 25 |
| | | 80386 | 43 |
| | | 80486 | 43 |
| | | Pentium | 22–46 |
| **IDIV mem** | **IDIV DATAY**<br>**IDIV LIST**<br>**IDIV DATA6[DI]**<br>**IDIV BYTE PTR [BX]**<br>IDIV WORD PTR [EBX] | [illegible] | [illegible] |
| | | [illegible] | [illegible] |
| | | 80286 | 28 |
| | | 80386 | 46 |
| | | 80486 | 44 |
| | | Pentium | 22–46 |

[redacted]

1111011w oo101mmm disp

| O | D | I | T | S | Z | A | P | C |
|---|---|---|---|---|---|---|---|---|
| * |  |  |  | ? | ? | ? | ? | * |

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| **IMUL reg** | **IMUL BL**<br>**IMUL CX**<br>IMUL ECX<br>**IMUL DH**<br>**IMUL AL** | [redacted] | [redacted] |
|  |  | 80286 | 21 |
|  |  | 80386 | 38 |
|  |  | 80486 | 42 |
|  |  | Pentium | 10–11 |
| **IMUL mem** | **IMUL DATAY**<br>**IMUL LIST**<br>**IMUL DATA6[DI]**<br>**IMUL BYTE PTR [BX]**<br>IMUL WORD PTR [EBX] | [redacted] | [redacted] |
|  |  | 80286 | 24 |
|  |  | 80386 | 41 |
|  |  | 80486 | 42 |
|  |  | Pentium | 10–11 |

011010s1 oorrmmm disp data

| Format | Examples | Microprocessor | Clocks |
|---|---|---|---|
| IMUL reg,imm | IMUL CX,16<br>IMUL DI,100<br>IMUL EDX,20 | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | 21 |
|  |  | 80386 | 38 |
|  |  | 80486 | 42 |
|  |  | Pentium | 10 |
| IMUL reg,reg,imm | IMUL DX,AX,2<br>IMUL CX,DX,3<br>IMUL BX,AX,33 | 8086 | — |
|  |  | 8088 | — |
|  |  | 80286 | 21 |
|  |  | 80386 | 38 |
|  |  | 80486 | 42 |
|  |  | Pentium | 10 |