80x86 Instructions

This is a list of instructions for use in the unit DP234. Note that there are many other instructions.

An explanation of instruction arguments is at the end of this document.

ADC - Add With Carry	ADD - Add	AND - Logical And
CALL - Subroutine Call	CBW - Convert Byte to Word	CLC - Clear Carry Flag
CLD - Clear Direction Flag	CLI - Clear Interrupt Flag	CMC - Complement Carry Flag
CMP - Compare	CMPS - Compare String	CWD - Convert Word to Double
DEC - Decrement	DIV - Divide	HALT - Halt CPU
IDIV - Signed Division	IMUL - Signed Multiply	IN - Input from port
INC - Increment	INS - Input String	INT- Interrupt
INTO - Interrupt on Overflow	IRET - Interrupt Return	Jxx - Conditional Jump
JCXZ - Jump CX Zero	JMP - unconditional Jump	LDS - Load Pointer (DS)
LEA - Load Effective Address	LES - Load Pointer (ES)	LODS - Load String
LOOP - Loop	LOOPE - Loop on Equal	LOOPNZ - Loop Non Zero
MOV - Move	MOVS - Move string	MUL - Multiply
NEG - Negation	NOP - No-operation	NOT - Logical NOT
OR - Logical OR	OUT - Ooutput to port	OUTS - Output String
POP - Pop from stack	POPA - Pop all registers	POPF - Pop Flags
PUSH - Push onto stack	PUSHA - Push all flags	PUSHF - Push Flags
RCL - Rotate Left (carry)	RCR - Rotate Carry (right)	REP - Repeat instruction
REPE - Repeat Equal	REPNE - Repeat Not Equal	RET - Subroutine return
ROL - Rotate Left	ROR - Rotate Right	SAL - Shift Left
SAR - Shift Right	SBB - Subtract with carry	SCAS - Scan String
SHL - Shift Left	STC - Set Carry flag	STD - Set Direction flag
STI - Set Interrupt flag	STOS - Store String	SUB - Subtract
TEST - Test bits	XCHG - exchange values	XOR - Logical Exclusive OR

ADC - Add With Carry

Usage: ADC dest,src

Modifies flags: AF CF OF SF PF ZF

Sums two binary operands placing the result in the destination. If CF is set, a 1 is added to the destination.

Arguments: reg,reg mem,reg reg,mem reg,immed mem,immed accum,immed

ADD - Arithmetic Addition

Usage: ADD dest,src

Modifies flags: AF CF OF PF SF ZF

Adds "src" to "dest" and replacing the original contents of "dest". Both operands are binary.

Arguments: reg,reg mem,reg reg,mem reg,immed mem,immed accum,immed

AND - Logical And

Usage: AND dest,src

Modifies flags: CF OF PF SF ZF (AF undefined)

Performs a logical AND of the two operands replacing the destination with the result.

Arguments: reg,reg mem,reg reg,mem reg,immed mem,immed accum,immed

CALL - Procedure Call

Usage: CALL destination

Modifies flags: None

Pushes Instruction Pointer (and Code Segment for far calls) onto stack and loads Instruction Pointer with the address of proc-name. Code continues with execution at CS:IP.

CBW - Convert Byte to Word

Usage: CBW

Modifies flags: None

Converts byte in AL to word Value in AX by extending sign of AL throughout register AH.

CLC - Clear Carry

Usage: CLC

Modifies flags: CF

Clears the Carry Flag.

CLD - Clear Direction Flag

Usage: CLD

Modifies flags: DF

Clears the Direction Flag causing string instructions to increment the SI and DI index registers.

CLI - Clear Interrupt Flag (disable)

Usage: CLI

Modifies flags: IF

Disables the maskable hardware interrupts by clearing the Interrupt flag. NMI's and software interrupts are not inhibited.

CMC - Complement Carry Flag

Usage: CMC

Modifies flags: CF

Toggles (inverts) the Carry Flag

CMP - Compare

Usage: CMP dest,src

Modifies flags: AF CF OF PF SF ZF

Subtracts source from destination and updates the flags but does not save result. Flags can subsequently be checked for conditions (e.g. with Jxx instructions).

Arguments: reg,reg mem,reg reg,mem reg,immed mem,immed accum,immed

CMPS - Compare String (Byte, Word)

Usage: CMPS dest,src CMPSB CMPSW

Modifies flags: AF CF OF PF SF ZF

Subtracts destination value from source without saving results. Updates flags based on the subtraction and the index registers SI and DI are incremented or decremented depending on the state of the Direction Flag. CMPSB inc/decrements the index registers by 1, CMPSW inc/decrements by 2, while CMPSD increments or decrements by 4. The REP prefixes can be used to process entire data items.

CWD - Convert Word to Doubleword

Usage: CWD

Modifies flags: None

Extends sign of word in register AX throughout register DX forming a doubleword quantity in DX:AX.

DEC - Decrement

Usage: DEC dest

Modifies flags: AF OF PF SF ZF

Unsigned binary subtraction of one from the destination.

DIV - Divide

Usage: DIV src

Modifies flags: (AF,CF,OF,PF,SF,ZF undefined)

Unsigned binary division of accumulator by source. If the source divisor is a byte value then AX is divided by "src" and the quotient is placed in AL and the remainder in AH. If source operand is a word value, then DX:AX is divided by "src" and the quotient is stored in AX and the remainder in DX.

Arguments: reg8 reg16 mem8 mem16

HLT - Halt CPU

Usage: HLT

Modifies flags: None

Halts CPU until RESET line is activated, NMI or maskable interrupt received. The CPU becomes dormant but retains the current CS:IP for later restart.

IDIV - Signed Integer Division

Usage: IDIV src

Modifies flags: (AF,CF,OF,PF,SF,ZF undefined)

Signed binary division of accumulator by source. If source is a byte value, AX is divided by "src" and the quotient is stored in AL and the remainder in AH. If source is a word value, DX:AX is divided by "src", and the quotient is stored in AL and the remainder in DX.

Arguments: reg8 reg16 mem8 mem16

IMUL - Signed Multiply

Usage: IMUL src

Modifies flags: CF OF (AF,PF,SF,ZF undefined)

Signed multiplication of accumulator by "src" with result placed in the accumulator. If the source operand is a byte value, it is multiplied by AL and the result stored in AX. If the source operand is a word value it is multiplied by AX and the result is stored in DX:AX.

Arguments: reg8 reg16 mem8 mem16

IN - Input Byte or Word From Port

Usage: IN accum,port

Modifies flags: None

A byte or word is read from "port" and placed in AL or AX respectively. If the port number is in the range of 0-255 it can be specified as an immediate, otherwise the port number must be specified in DX. Valid port ranges on the PC are 0-1024, though values through 65535 may be specified and recognized by third party vendors and PS/2's.

INC - Increment

Usage: INC dest

Modifies flags: AF OF PF SF ZF

Adds one to destination unsigned binary operand.

INS - Input String from Port (80188+)

Usage: INS dest,port INSB INSW

Modifies flags: None

Loads data from port to the destination ES:DI (even if a destination operand is supplied). DI is adjusted by the size of the operand and increased if the Direction Flag is cleared and decreased if the Direction Flag is set. For INSB, INSW no operands are allowed and the size is determined by the mnemonic.

INT - Interrupt

Usage: INT num

Modifies flags: TF IF

Initiates a software interrupt by pushing the flags, clearing the Trap and Interrupt Flags, pushing CS followed by IP and loading CS:IP with the value found in the interrupt vector table. Execution then begins at the location addressed by the new CS:IP

INTO - Interrupt on Overflow

Usage: INTO

Modifies flags: IF TF

If the Overflow Flag is set this instruction generates an INT 4 which causes the code addressed by 0000:0010 to be executed.

IRET - Interrupt Return

Usage: IRET

Modifies flags: AF CF DF IF PF SF TF ZF

Returns control to point of interruption by popping IP, CS and then the Flags from the stack and

continues execution at this location. CPU exception interrupts will return to the instruction that cause the exception because the CS:IP placed on the stack during the interrupt is the address of the offending instruction.

Jxx - Jump Instructions Table

Mnemonic Meaning Jump Condition

JA Jump if Above CF=0 and ZF=0

JAE Jump if Above or Equal CF=0

JB Jump if Below CF=1

JBE Jump if Below or Equal CF=1 or ZF=1

JC Jump if Carry CF=1

JCXZ Jump if CX Zero CX=0

JE Jump if Equal ZF=1

JG Jump if Greater (signed) ZF=0 and SF=OF

JGE Jump if Greater or Equal (signed) SF=OF

JL Jump if Less (signed) SF != OF

JLE Jump if Less or Equal (signed) ZF=1 or SF != OF

JMP Unconditional Jump unconditional

JNA Jump if Not Above CF=1 or ZF=1

JNAE Jump if Not Above or Equal CF=1

JNB Jump if Not Below CF=0

JNBE Jump if Not Below or Equal CF=0 and ZF=0

JNC Jump if Not Carry CF=0

JNE Jump if Not Equal ZF=0

JNG Jump if Not Greater (signed) ZF=1 or SF != OF

JNGE Jump if Not Greater or Equal (signed) SF != OF

JNL Jump if Not Less (signed) SF=OF

JNLE Jump if Not Less or Equal (signed) ZF=0 and SF=OF

JNO Jump if Not Overflow (signed) OF=0

JNP Jump if No Parity PF=0

JNS Jump if Not Signed (signed) SF=0

JNZ Jump if Not Zero ZF=0

JO Jump if Overflow (signed) OF=1

JP Jump if Parity PF=1

JPE Jump if Parity Even PF=1

JPO Jump if Parity Odd PF=0

JS Jump if Signed (signed) SF=1

JZ Jump if Zero ZF=1

- It's a good programming practice to organize code so the expected case is executed without a jump since the actual jump takes longer to execute than falling through the test.

 JCXZ - Jump if Register CX is Zero

Usage: JCXZ label

Modifies flags: None

Causes execution to branch to "label" if register CX is zero. Uses unsigned comparision.

 JMP - Unconditional Jump

Usage: JMP label

Modifies flags: None

Unconditionally transfers control to "label". Jumps by default are within -32768 to 32767 bytes from the instruction following the jump. NEAR and SHORT jumps cause the IP to be updated while FAR jumps cause CS and IP to be updated.

 LDS - Load Pointer Using DS

Usage: LDS dest,src

Modifies flags: None

Loads 32-bit pointer from memory source to destination register and DS. The offset is placed in the destination register and the segment is placed in DS. To use this instruction the word at the lower memory address must contain the offset and the word at the higher address must contain the segment. This simplifies the loading of far pointers from the stack and the interrupt vector table.

 LEA - Load Effective Address

Usage: LEA dest,src

Modifies flags: None

Transfers offset address of "src" to the destination register.

LES - Load Pointer Using ES

Usage: LES dest,src Modifies flags: None

Loads 32-bit pointer from memory source to destination register and ES. The offset is placed in the destination register and the segment is placed in ES. To use this instruction the word at the lower memory address must contain the offset and the word at the higher address must contain the segment. This simplifies the loading of far pointers from the stack and the interrupt vector table.

LODS - Load String (Byte, Word)

Usage: LODS src LODSB LODSW

Modifies flags: None

Transfers string element addressed by DS:SI (even if an operand is supplied) to the accumulator. SI is incremented based on the size of the operand or based on the instruction used. If the Direction Flag is set SI is decremented, if the Direction Flag is clear SI is incremented. Use with REP prefixes.

LOOP - Decrement CX and Loop if CX Not Zero

Usage: LOOP label

Modifies flags: None

Decrements CX by 1 and transfers control to "label" if CX is not Zero. The "label" operand must be within -128 or 127 bytes of the instruction following the loop instruction

LOOPE/LOOPZ - Loop While Equal / Loop While Zero

Usage: LOOPE label LOOPZ label Modifies flags: None

Decrements CX by 1 (without modifying the flags) and transfers control to "label" if CX != 0 and the Zero Flag is set. The "label" operand must be within -128 or 127 bytes of the instruction following the loop instruction.

LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal

Usage: LOOPNZ label LOOPNE label

Modifies flags: None

Decrements CX by 1 (without modifying the flags) and transfers control to "label" if CX != 0 and the Zero Flag is clear. The "label" operand must be within -128 or 127 bytes of the instruction following the loop instruction.

MOV - Move Byte or Word

Usage: MOV dest,src

Modifies flags: None

Copies byte or word from the source operand to the destination operand. If the destination is SS interrupts are disabled except on early buggy 808x CPUs. Some CPUs disable interrupts if the destination is any of the segment registers

Arguments: reg,reg mem,reg reg,mem mem,immed reg,immed mem,accum accum,mem segreg,reg16 segreg,mem16 reg16,segreg mem16,segreg

MOVS - Move String (Byte or Word)

Usage: MOVS dest,src MOVSB MOVSW

Modifies flags: None

Copies data from addressed by DS:SI (even if operands are given) to the location ES:DI destination and updates SI and DI based on the size of the operand or instruction used. SI and DI are incremented when the Direction Flag is cleared and decremented when the Direction Flag is Set. Use with REP prefixes.

MUL - Unsigned Multiply

Usage: MUL src

Modifies flags: CF OF (AF,PF,SF,ZF undefined)

Unsigned multiply of the accumulator by the source. If "src" is a byte value, then AL is used as the other multiplicand and the result is placed in AX. If "src" is a word value, then AX is multiplied by "src" and DX:AX receives the result.

Arguments: reg8 reg16 mem8 mem16

NEG - Two's Complement Negation

Usage: NEG dest

Modifies flags: AF CF OF PF SF ZF

Subtracts the destination from 0 and saves the 2s complement of "dest" back into "dest".

NOP - No Operation (90h)

Usage: NOP

Modifies flags: None

This is a do nothing instruction. It results in occupation of both space and time and is most useful for patching code segments. (This is the original XCHG AL,AL instruction)

NOT - One's Compliment Negation (Logical NOT)

Usage: NOT dest

Modifies flags: None

Inverts the bits of the "dest" operand forming the 1s complement.

OR - Inclusive Logical OR

Usage: OR dest,src

Modifies flags: CF OF PF SF ZF (AF undefined)

Logical inclusive OR of the two operands returning the result in the destination. Any bit set in either operand will be set in the destination.

Arguments: reg,reg mem,reg reg,mem reg,immed mem8,immed8 mem16,immed16 accum,immed

OUT - Output Data to Port

Usage: OUT port,accum

Modifies flags: None

Transfers byte in AL,word in AX or dword in EAX to the specified hardware port address. If the port number is in the range of 0-255 it can be specified as an immediate. If greater than 255 then the port number must be specified in DX. Since the PC only decodes 10 bits of the port address, values over 1023 can only be decoded by third party vendor equipment and also map to the port range 0-1023.

Arguments: immed8,accum immed8,accum DX,accum DX,accum

OUTS - Output String to Port (80188+)

Usage: OUTS port,src OUTSB OUTSW

Modifies flags: None

Transfers a byte, word or doubleword from "src" to the hardware port specified in DX. For instructions with no operands the "src" is located at DS:SI and SI is incremented or decremented by the size of the operand or the size dictated by the instruction format. When the Direction Flag is set SI is decremented, when clear, SI is incremented. If the port number is in the range of 0-255 it can be specified as an immediate. If greater than 255 then the port number must be specified in DX. Since the PC only decodes 10 bits of the port address, values over 1023 can only be decoded by third party vendor equipment and also map to the port range 0-1023.

POP - Pop Word off Stack

Usage: POP dest Modifies flags: None

Transfers word at the current stack top (SS:SP) to the destination then increments SP by two to point to the new stack top. CS is not a valid destination.

POPA - Pop All Registers off Stack (80188+)

Usage: POPA

Modifies flags: None

Pops the top 8 words off the stack into the 8 general purpose 16 bit registers. Registers are popped in the following order: DI, SI, BP, SP, DX, CX and AX. The SP value popped from the stack is actually discarded.

POPF - Pop Flags off Stack

Usage: POPF

Modifies flags: all flags

Pops word from stack into the Flags Register and then increments SP by 2.

PUSH - Push Word onto Stack

Usage: PUSH src PUSH immed (80188+ only)

Modifies flags: None

Decrements SP by the size of the operand (two or four, byte values are sign extended) and transfers one word from source to the stack top (SS:SP).

PUSHA - Push All Registers onto Stack (80188+)

Usage: PUSHA

Modifies flags: None

Pushes all general purpose registers onto the stack in the following order: AX, CX, DX, BX, SP, BP, SI, DI. The value of SP is the value before the actual push of SP.

PUSHF - Push Flags onto Stack

Usage: PUSHF

Modifies flags: None

Transfers the Flags Register onto the stack. PUSHF saves a 16 bit value.

RCL - Rotate Through Carry Left

Usage: RCL dest,count

Modifies flags: CF OF

Rotates the bits in the destination to the left "count" times with all data pushed out the left side re-entering on the right. The Carry Flag holds the last bit rotated out.

Arguments: reg,1 mem,1 reg,CL mem,CL reg,immed8 mem,immed8

RCR - Rotate Through Carry Right

Usage: RCR dest,count

Modifies flags: CF OF

Rotates the bits in the destination to the right "count" times with all data pushed out the right side re-entering on the left. The Carry Flag holds the last bit rotated out.

Arguments: reg,1 mem,1 reg,CL mem,CL reg,immed8 mem,immed8

REP - Repeat String Operation

Usage: REP

Modifies flags: None

Repeats execution of string instructions while CX != 0. After each string operation, CX is decremented and the Zero Flag is tested. The combination of a repeat prefix and a segment override on CPU's before the 386 may result in errors if an interrupt occurs before CX=0.

REPE/REPZ - Repeat Equal / Repeat Zero

Usage: REPE / REPZ

Modifies flags: None

Repeats execution of string instructions while CX != 0 and the Zero Flag is set. CX is decremented and the Zero Flag tested after each string operation. The combination of a repeat prefix and a segment override on processors other than the 386 may result in errors if an interrupt occurs before CX=0.

REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero

Usage: REPNE REPNZ

Modifies flags: None

Repeats execution of string instructions while CX != 0 and the Zero Flag is clear. CX is decremented and the Zero Flag tested after each string operation. The combination of a repeat prefix and a segment override on processors other than the 386 may result in errors if an interrupt occurs before CX=0.

RET/RETF - Return From Procedure

Usage: RET / RETF

Modifies flags: None

Transfers control from a procedure back to the instruction address saved on the stack. Far returns (RETF) pop the IP followed by the CS, while near returns pop only the IP register.

ROL - Rotate Left

Usage: ROL dest,count

Modifies flags: CF OF

Rotates the bits in the destination to the left "count" times with all data pushed out the left side re-entering on the right. The Carry Flag will contain the value of the last bit rotated out.

Arguments: reg,1 mem,1 reg,CL mem,CL reg,immed8 mem,immed8

ROR - Rotate Right

Usage: ROR dest,count

Modifies flags: CF OF

Rotates the bits in the destination to the right "count" times with all data pushed out the right side re-entering on the left. The Carry Flag will contain the value of the last bit rotated out.

Arguments: reg,1 mem,1 reg,CL mem,CL reg,immed8 mem,immed8

SAL/SHL - Shift Arithmetic Left / Shift Logical Left

Usage: SAL dest,count SHL dest,count

Modifies flags: CF OF PF SF ZF (AF undefined)

Shifts the destination left by "count" bits with zeroes shifted in on right. The Carry Flag contains the last bit shifted out.

Arguments: reg,1 mem,1 reg,CL mem,CL reg,immed8 mem,immed8

SAR - Shift Arithmetic Right

Usage: SAR dest,count

Modifies flags: CF OF PF SF ZF (AF undefined)

Shifts the destination right by "count" bits with the current sign bit replicated in the leftmost bit. The Carry Flag contains the last bit shifted out.

Arguments: reg,1 mem,1 reg,CL mem,CL reg,immed8 mem,immed8

SBB - Subtract with Borrow/Carry

Usage: SBB dest,src Modifies flags: AF CF OF PF SF ZF

Subtracts the source from the destination, and subtracts 1 extra if the Carry Flag is set. Results are returned in "dest".

Arguments: reg,reg mem,reg reg,mem reg,immed mem,immed accum,immed

SCAS - Scan String (Byte, Word)

Usage: SCAS string SCASB SCASW

Modifies flags: AF CF OF PF SF ZF

Compares value at ES:DI (even if operand is specified) from the accumulator and sets the flags similar to a subtraction. DI is incremented/decremented based on the instruction format (or operand size) and the state of the Direction Flag. Use with REP prefixes.

SHL - Shift Logical Left

See: SAL SHR - Shift Logical Right

Usage: SHR dest,count

Modifies flags: CF OF PF SF ZF (AF undefined)

Shifts the destination right by "count" bits with zeroes shifted in on the left. The Carry Flag contains the last bit shifted out.

Arguments: reg,1 mem,1 reg,CL mem,CL reg,immed8 mem,immed8

STC - Set Carry

Usage: STC Modifies flags: CF

Sets the Carry Flag to 1.

STD - Set Direction Flag

Usage: STD Modifies flags: DF

Sets the Direction Flag to 1 causing string instructions to auto-decrement SI and DI instead of auto-increment.

STI - Set Interrupt Flag (Enable Interrupts)

Usage: STI Modifies flags: IF

Sets the Interrupt Flag to 1, which enables recognition of all hardware interrupts. If an interrupt is generated by a hardware device, an End of Interrupt (EOI) must also be issued to enable other hardware interrupts of the same or lower priority.

STOS - Store String (Byte, Word)

Usage: STOS dest STOSB STOSW Modifies flags: None

Stores value in accumulator to location at ES:DI (even if operand is given). DI is incremented/decremented based on the size of the operand (or instruction format) and the state of the Direction Flag. Use with REP prefixes.

SUB - Subtract

Usage: SUB dest,src

Modifies flags: AF CF OF PF SF ZF

The source is subtracted from the destination and the result is stored in the destination.

Arguments: reg,reg mem,reg reg,mem reg,immed mem,immed accum,immed

TEST - Test For Bit Pattern

Usage: TEST dest,src

Modifies flags: CF OF PF SF ZF (AF undefined)

Performs a logical AND of the two operands updating the flags register without saving the result.

Arguments: reg,reg reg,mem mem,reg reg,immed mem,immed accum,immed

XCHG - Exchange

Usage: XCHG dest,src

Modifies flags: None

Exchanges contents of source and destination.

Arguments: reg,reg mem,reg reg,mem accum,reg reg,accum

XOR - Exclusive OR

Usage: XOR dest,src

Modifies flags: CF OF PF SF ZF (AF undefined)

Performs a bitwise exclusive OR of the operands and returns the result in the destination.

Arguments: reg,reg mem,reg reg,mem reg,immed mem,immed accum,immed

---

Arguments to instructions

Abreviation	Meaning	Example
1	the value 1	SAL AX,**1**
accum	register AX	MOV **AX**,23
CL	the register CL	SAL AX,**CL**
immed	a value	MOV AX,**2345**
immed8	an 8 bit value	MOV AX,**23**
mem	byte or word address	MOV AX,**lab1**
mem8	byte address	MOV AX,**lab1**
mem16	word address	MOV AX,**lab1**
reg	8 or 16 bit register	**AX**, **DH**,...
reg8	8 bit register	**AH**, **AL**, ...

reg16	16 bit register	**AX**,**BX**, ...
segreg	segment registers	**CS**, **DS**, **SS**, **ES**

This page is maintained by Barry Wilks.