Appendix A

80x86 Instructions

A.1 Non-floating Point Instructions

This section lists and describes the actions and formats of the non-floating point instructions of the Intel 80x86 CPU family.

The formats use the following abbreviations:

R	general register
R8	8-bit register
R16	16-bit register
R32	32-bit register
SR	segment register
M	memory
M8	byte
M16	word
M32	double word
I	immediate value

These can be combined for the multiple operand instructions. For example, the format *R, R* means that the instruction takes two register operands. Many of the two operand instructions allow the same operands. The abbreviation *O2* is used to represent these operands: *R,R R,M R,I M,R M,I.* If a 8-bit register or memory can be used for an operand, the abbreviation, *R/M8* is used.

The table also shows how various bits of the FLAGS register are affected by each instruction. If the column is blank, the corresponding bit is not affected at all. If the bit is always changed to a particular value, a 1 or 0 is shown in the column. If the bit is changed to a value that depends on the operands of the instruction, a *C* is placed in the column. Finally, if the bit is modified in some undefined way a *?* appears in the column. Because the

only instructions that change the direction flag are `CLD` and `STD`, it is not listed under the FLAGS columns.

Name	Description	Formats	Flags					
			O	**S**	**Z**	**A**	**P**	**C**
`ADC`	Add with Carry	O2	C	C	C	C	C	C
`ADD`	Add Integers	O2	C	C	C	C	C	C
`AND`	Bitwise AND	O2	0	C	C	?	C	0
`BSWAP`	Byte Swap	R32						
`CALL`	Call Routine	R M I						
`CBW`	Convert Byte to Word							
`CDQ`	Convert Dword to Qword							
`CLC`	Clear Carry							0
`CLD`	Clear Direction Flag							
`CMC`	Complement Carry							C
`CMP`	Compare Integers	O2	C	C	C	C	C	C
`CMPSB`	Compare Bytes		C	C	C	C	C	C
`CMPSW`	Compare Words		C	C	C	C	C	C
`CMPSD`	Compare Dwords		C	C	C	C	C	C
`CWD`	Convert Word to Dword into DX:AX							
`CWDE`	Convert Word to Dword into EAX							
`DEC`	Decrement Integer	R M	C	C	C	C	C	
`DIV`	Unsigned Divide	R M	?	?	?	?	?	?
`ENTER`	Make stack frame	I,0						
`IDIV`	Signed Divide	R M	?	?	?	?	?	?
`IMUL`	Signed Multiply	R M R16,R/M16 R32,R/M32 R16,I R32,I R16,R/M16,I R32,R/M32,I	C	?	?	?	?	C
`INC`	Increment Integer	R M	C	C	C	C	C	
`INT`	Generate Interrupt	I						
`JA`	Jump Above	I						
`JAE`	Jump Above or Equal	I						
`JB`	Jump Below	I						
`JBE`	Jump Below or Equal	I						
`JC`	Jump Carry	I						

Name	Description	Formats	Flags					
			O	S	Z	A	P	C
JCXZ	Jump if CX = 0	I						
JE	Jump Equal	I						
JG	Jump Greater	I						
JGE	Jump Greater or Equal	I						
JL	Jump Less	I						
JLE	Jump Less or Equal	I						
JMP	Unconditional Jump	R M I						
JNA	Jump Not Above	I						
JNAE	Jump Not Above or Equal	I						
JNB	Jump Not Below	I						
JNBE	Jump Not Below or Equal	I						
JNC	Jump No Carry	I						
JNE	Jump Not Equal	I						
JNG	Jump Not Greater	I						
JNGE	Jump Not Greater or Equal	I						
JNL	Jump Not Less	I						
JNLE	Jump Not Less or Equal	I						
JNO	Jump No Overflow	I						
JNS	Jump No Sign	I						
JNZ	Jump Not Zero	I						
JO	Jump Overflow	I						
JPE	Jump Parity Even	I						
JPO	Jump Parity Odd	I						
JS	Jump Sign	I						
JZ	Jump Zero	I						
LAHF	Load FLAGS into AH							
LEA	Load Effective Address	R32,M						
LEAVE	Leave Stack Frame							
LODSB	Load Byte							
LODSW	Load Word							
LODSD	Load Dword							
LOOP	Loop	I						
LOOPE/LOOPZ	Loop If Equal	I						
LOOPNE/LOOPNZ	Loop If Not Equal	I						

Name	Description	Formats	Flags					
			O	S	Z	A	P	C
MOV	Move Data	O2						
		SR,R/M16						
		R/M16,SR						
MOVSB	Move Byte							
MOVSW	Move Word							
MOVSD	Move Dword							
MOVSX	Move Signed	R16,R/M8						
		R32,R/M8						
		R32,R/M16						
MOVZX	Move Unsigned	R16,R/M8						
		R32,R/M8						
		R32,R/M16						
MUL	Unsigned Multiply	R M	C	?	?	?	?	C
NEG	Negate	R M	C	C	C	C	C	C
NOP	No Operation							
NOT	1's Complement	R M						
OR	Bitwise OR	O2	0	C	C	?	C	0
POP	Pop From Stack	R/M16						
		R/M32						
POPA	Pop All							
POPF	Pop FLAGS		C	C	C	C	C	C
PUSH	Push to Stack	R/M16						
		R/M32 I						
PUSHA	Push All							
PUSHF	Push FLAGS							
RCL	Rotate Left with Carry	R/M,I	C					C
		R/M,CL						
RCR	Rotate Right with Carry	R/M,I	C					C
		R/M,CL						
REP	Repeat							
REPE/REPZ	Repeat If Equal							
REPNE/REPNZ	Repeat If Not Equal							
RET	Return							
ROL	Rotate Left	R/M,I	C					C
		R/M,CL						
ROR	Rotate Right	R/M,I	C					C
		R/M,CL						
SAHF	Copies AH into FLAGS			C	C	C	C	C

Name	Description	Formats	Flags					
			O	S	Z	A	P	C
SAL	Shifts to Left	R/M,I R/M, CL						C
SBB	Subtract with Borrow	O2	C	C	C	C	C	C
SCASB	Scan for Byte		C	C	C	C	C	C
SCASW	Scan for Word		C	C	C	C	C	C
SCASD	Scan for Dword		C	C	C	C	C	C
SETA	Set Above	R/M8						
SETAE	Set Above or Equal	R/M8						
SETB	Set Below	R/M8						
SETBE	Set Below or Equal	R/M8						
SETC	Set Carry	R/M8						
SETE	Set Equal	R/M8						
SETG	Set Greater	R/M8						
SETGE	Set Greater or Equal	R/M8						
SETL	Set Less	R/M8						
SETLE	Set Less or Equal	R/M8						
SETNA	Set Not Above	R/M8						
SETNAE	Set Not Above or Equal	R/M8						
SETNB	Set Not Below	R/M8						
SETNBE	Set Not Below or Equal	R/M8						
SETNC	Set No Carry	R/M8						
SETNE	Set Not Equal	R/M8						
SETNG	Set Not Greater	R/M8						
SETNGE	Set Not Greater or Equal	R/M8						
SETNL	Set Not Less	R/M8						
SETNLE	Set Not LEss or Equal	R/M8						
SETNO	Set No Overflow	R/M8						
SETNS	Set No Sign	R/M8						
SETNZ	Set Not Zero	R/M8						
SETO	Set Overflow	R/M8						
SETPE	Set Parity Even	R/M8						
SETPO	Set Parity Odd	R/M8						
SETS	Set Sign	R/M8						
SETZ	Set Zero	R/M8						
SAR	Arithmetic Shift to Right	R/M,I R/M, CL						C

Name	Description	Formats	Flags					
			O	S	Z	A	P	C
SHR	Logical Shift to Right	R/M,I						C
		R/M, CL						
SHL	Logical Shift to Left	R/M,I						C
		R/M, CL						
STC	Set Carry							1
STD	Set Direction Flag							
STOSB	Store Btye							
STOSW	Store Word							
STOSD	Store Dword							
SUB	Subtract	O2	C	C	C	C	C	C
TEST	Logical Compare	R/M,R	0	C	C	?	C	0
		R/M,I						
XCHG	Exchange	R/M,R						
		R,R/M						
XOR	Bitwise XOR	O2	0	C	C	?	C	0