Algumas instruções adicionais da família 80x86 (modo de 32 bits)

			FLAGS								
Nome	***Código***	***Ação***	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
Saltos											
JECXZ	JECXZ dest	Salta se ECX=0									
LOOP	LOOP dest	ECX ← ECX-1, salta se ECX<>0									
LOOPE	LOOPE dest	ECX ← ECX-1, salta se ECX<>0 e Z=1									
LOOPNE	LOOPNE dest	ECX ← ECX-1, salta se ECX<>0 e Z=0									
Sequências											
CMPSB	CMPSB	CMP(BYTE PTR [ESI],BYTE PTR [EDI]); ESI←ESI±1; EDI←EDI±1	X				X	X	X	X	X
CMPSW	CMPSW	CMP(WORD PTR [ESI],WORD PTR [EDI]); ESI←ESI±2; EDI←EDI±2	X				X	X	X	X	X
CMPSD	CMPSD	CMP(DWORD PTR [ESI],DWORD PTR [EDI]);ESI←ESI±4; EDI←EDI±4	X				X	X	X	X	X
LODSB	LODSB	AL ← [ESI]; ESI ←ESI ± 1									
LODSW	LODSW	AX ← [ESI]; ESI ←ESI ± 2									
LODSD	LODSD	EAX ← [ESI]; ESI ←ESI ± 4									
MOVSB	MOVSB	BYTE PTR [EDI] ←[ESI]; ESI ←ESI ± 1; EDI ←EDI ± 1									
MOVSW	MOVSW	WORD PTR [EDI] ←[ESI]; ESI ←ESI ± 2; EDI ←EDI ± 2									
MOVSD	MOVSD	DWORD PTR [EDI] ←[ESI]; ESI ←ESI ± 4; EDI ←EDI ± 4									
REP	prefixo	enquanto (ECX<>0) {ECX ← ECX-1}									
REPE	prefixo	enquanto (ECX<>0) {ECX ← ECX-1; se (Z=0) terminar}									
REPZ	prefixo	enquanto (ECX<>0) {ECX ← ECX-1; se (Z=0) terminar}									
REPNE	prefixo	enquanto (ECX<>0) {ECX ← ECX-1; se (Z=1) terminar}									
REPNZ	prefixo	enquanto (ECX<>0) {ECX ← ECX-1; se (Z=1) terminar}									
SCASB	SCASB	CMP(AL,[EDI]);EDI ←EDI ± 1	X				X	X	X	X	X
SCASW	SCASW	CMP(AX,[EDI]);EDI ←EDI ± 2	X				X	X	X	X	X
SCASD	SCASD	CMP(EAX,[EDI]);EDI ←EDI ± 4	X				X	X	X	X	X
STOSB	STOSB	[EDI] ← AL ; EDI ←EDI ± 1									
STOSW	STOSW	[EDI] ← AX ; EDI ←EDI ± 2									
STOSD	STOSD	[EDI] ← EAX; EDI ←EDI ± 4									
Operações lógicas e aritméticas											
BT(1)	BT r/m, N	Copia bit N de r/m para CF; N: r16, r32 ou imm8									
SET<cc>	SET<cc> r8/m8	r8/m8 ← valor de <cc>									
TEST	TEST op1, op2	op1 AND op2 (só afecta flags)	0				X	X	?	X	0
Transferências											
CMOV<cc>(1)	CMOV<cc> r1, r/m	r1 ← r/m, se condição for verdadeira									
CDQ	CDQ	EDX:EAX ← expansão de sinal de EAX									
LAHF	LAHF	Copia LSB de EFLAGS para AH									
LEAVE	LEAVE	Equivalente a: MOV ESP, EBP; POP EBP									
MOVZX	MOVZX r1, r/m	reg1 ← r/m, reg1 de dimensão superior a r/m, extensão com 0									
MOVSX	MOVSX r1, r/m	reg1 ← r/m, reg1 de dimensão superior a r/m, extensão de sinal									
POPAD	POPAD	Copia valores da pilha para o registos de uso geral									
POPFD	POPFD	Copia topo da pilha para EFLAGS	X	X	X	X	X	X	X	X	X
PUSHAD	PUSHAD	Copia registos de uso geral para a pilha									
PUSHFD	PUSHFD	Copia EFLAGS para pilha									
SAHF	SAHF	Copia registo AH para LSB de EFLAGS					X	X	X	X	X
XLATB	XLATB	AL ← [AL + EBX] (AL: extensão com zeros)									

(1) Apenas r16/m16 ou r32/m32 (operando não pode ser de 8 bits)
<cc> códigos usados com as instruções de salto condicional: E, Z, A, AE, B, BE, C, G, GE, L, LE, O, S e formas negadas.
Registos de uso geral: EAX, ECX, EDX, EBX, ESP, ESI, EDI.

Tipos de dados (gama de representação)

Tipo	*Limite Inferior (dec|hex)*		*Limite Superior (dec|hex)*	
BYTE	0	0h	255	0FFh
SBYTE	-128	80h	127	7Fh
WORD	0	0h	65535	0FFFFh
SWORD	-32768	8000h	32767	7FFFh
DWORD	0	0h	4294967295	0FFFFFFFFh
SDWORD	-2147483648	80000000h	2147483647	7FFFFFFFh

Unidade de vírgula flutuante (UVF)

Nome	*Ação*
Aritmética	op_destino ← op_destino ***op*** op_fonte
FADD/FADDP/FIADD	op_destino ← op_destino + op_fonte
FDIV/FDIVP/FIDIV	op_destino ← op_destino / op_fonte
FDIVR/FDIVRP/FIDIVR	op_destino ← op_fonte / op_destino
FMUL/FMULP/FIMUL	op_destino ← op_destino × op_fonte
FSUB/FSUBP/FISUB	op_destino ← op_destino − op_fonte
FSUBR/FSUBRP/FISUBR	op_destino ← op_fonte − op_destino
Outras funções	ST ← *func*(ST)
FABS	Valor absoluto.
FCHS	Troca de sinal
FCOS	Cosseno (argumento em radianos)
FSIN	Seno (arg. em radianos)
FSINCOS	Seno e cosseno (arg. em radianos); cosseno fica no topo da pilha
FSQRT	Raiz quadrada (argumento não negativo)
F2XM1	Calcula 2^x-1
FYL2X	Calcula $Y*\log_2(X)$ c/ X = ST(0), Y = ST(1); X >= 0.0
Transferências	Memória : objectos de 32, 64 ou 80 bits
FLD/FILD/FBLD	*push(mem)* [I=inteiros, B=BCD]
FST/FSTP/FIST/FISTP/FBSTP	*mem* ←ST (com conversão para o formato apropriado, I=inteiro, B=BCD) ou ST(n) ←ST
FLD1	Coloca 1.0 no topo da pilha.
FLDZ	Coloca 0.0 no topo da pilha.
FLDPI	Coloca π no topo da pilha.
FXCH / FXCH ST(i)	Troca ST(0) com ST(1) / Troca ST(0) com ST(i)

Instruções de comparação

Instrução	*Ação*
FTST	Compara ST(0) com 0.0
FCOM *mem/ST(i)* [1] FCOMP *mem/ST(i)*	Compara ST(0) e operando (afeta *flags* da UVF)
FCOMPP	Compara ST(0) e ST(1) (afeta *flags* da UVF) e remove ambos da pilha
FCOMI ST(0),ST(i) [1]FCOMIP ST(0),ST(i)	Compara ST(0) e ST(i) (afeta *flags* do CPU)

[1] remove ST(0)

Modos de endereçamento para operações aritméticas

Modo	*Formato*[1]	*Exemplo*	
Pilha	ST(1), ST	FADD	
Registo	ST, ST(n)	FADD	ST, ST(2)
	ST(n), ST	FADD	ST(2), ST
Reg- pop	ST(n), ST	FADDP	ST(3), ST
Memória	Operando	FADD	*mem_num*

[1] Primeiro operando é o operando de destino.

Mnemónicas terminadas em **P** indicam que o topo da pilha é removido *(pop)*.