

附录

附录 A 常用 80x86 指令速查表

指令按助记符字母顺序排列，缩写、符号约定如下：

(1) 指令中，*dst, src* 表示目的操作数和源操作数。仅一个操作数时，个别处也表示为 *opr*。

(2) *imm* 表示立即数，8/16/32 位立即数记作：*imm8/imm16/imm32*。

(3) *reg* 表示通用寄存器，8/16/32 位通用寄存器记作：*reg8/reg16/reg32*。

(4) *mem* 表示内存操作数，8/16/32 等内存操作数记作：*mem8/mem16/mem32* 等。

(5) *seg* 表示段寄存器，CS, DS, SS, ES, FS, GS。

(6) *acc* 表示累加器，8/16/32 累加器对应 AL/AX/EAX。

(7) OF, SF, ZF, AF, PF, CF 分别表示为 O, S, Z, A, P, C，相应位置为：字母，根据结果状态设置；？，状态不确定；-，状态不变；1，置 1；0，清 0；例如：0SZ?P-表示：OF 清 0，AF 不确定，CF 不变，其它根据结果设置。若该栏空白，则表示无关。

(8) 寄存器符号诸如(E)CX, (E)SI, (E)DI, (E)SP, (E)BP 和(E)IP 等，表示在 16 地址模式下使用 16 位寄存器(如 CX)，或在 32 地址模式下使用 32 位寄存器(如 ECX)。

(9) 周期数表示指令执行所需的 CPU 时钟周期个数，即执行时间为：周期数/主频(秒)。

(10) 诸如(386+)是表示该指令只能用于 80386 及以后微处理器上。

指 令	功 能	指 令 形 式	周期数	影响标志位
AAA	非压缩BCD加法调整，AH+进位	AAA	3	?SZ?PC
AAD	AH×10+AL⇒AL，之后AH清0	AAD	10	OSZAPC
AAM	AL÷10的商⇒AH，余数⇒AL	AAM	18	OSZAPC
AAS	非压缩BCD减法调整，AH-借位	AAS	3	?SZ?PC
ADC <i>dst, src</i>	带进位加法： $dst+src+CF \Rightarrow dst$	ADC <i>reg, reg</i>	1	OSZAPC
		ADC <i>reg, mem</i>	2	
		ADC <i>reg, imm</i>	1	
		ADC <i>acc, imm</i>	1	
		ADC <i>mem, reg</i>	3	
		ADC <i>mem, imm</i>	3	
ADD <i>dst, src</i>	加法： $dst+src \Rightarrow dst$	ADD <i>reg, reg</i>	1	OSZAPC
		ADD <i>reg, mem</i>	2	
		ADD <i>reg, imm</i>	1	
		ADD <i>acc, imm</i>	1	
		ADD <i>mem, reg</i>	3	
		ADD <i>mem, imm</i>	3	

续表

指 令	功 能	指 令 形 式	周期数	影响标志位
AND <i>dst, src</i>	逻辑与: $dst \wedge src \Rightarrow dst$	AND <i>reg, reg</i>	1	0 S Z ? P 0
		AND <i>reg, mem</i>	2	
		AND <i>reg, imm</i>	1	
		AND <i>acc, imm</i>	1	
		AND <i>mem, reg</i>	3	
		AND <i>mem, imm</i>	3	
ARPL <i>dst, src</i>	调整选择器的RPL域	ARPL <i>reg/mem16, reg16</i>	7	-- Z ---
BOUND <i>reg, mem</i>	越界检查: (80188+) 若 <i>reg</i> 值超出 <i>mem</i> , 则产生INT 5	BOUND <i>reg16, mem32</i>	INT+32	
		BOUND <i>reg32, mem64</i>		
BSF <i>reg, src</i>	从低到高扫描 <i>src</i> , 16/32位 (386+) 若 <i>src</i> =0, ZF清0, 否则置1, 位置 \Rightarrow <i>reg</i>	BSF <i>reg, reg</i>	6~35	?? Z ???
		BSF <i>reg, mem</i>	6~43	
BSR <i>reg, src</i>	从高到低扫描 <i>src</i> , 16/32位 (386+) 若 <i>src</i> =0, ZF清0, 否则置1, 位置 \Rightarrow <i>reg</i>	BSR <i>reg, reg</i>	6~35	?? Z ???
		BSR <i>reg, mem</i>	6~43	
BSWAP <i>reg32</i>	反转 <i>reg32</i> 字节顺序 (486+)	BSWAP <i>reg32</i>	1	
BT <i>dst, src</i>	位测试 (386+) 由 <i>dst</i> 指定的位 \Rightarrow CF (16/32位)	BT <i>reg, reg</i>	4	????? C
		BT <i>reg, imm</i>	4	
		BT <i>mem, reg</i>	9	
		BT <i>mem, imm</i>	4	
BTC <i>dst, src</i>	位测试并变反 (386+) <i>dst</i> 的指定位 \Rightarrow CF, 然后该位变反, (16/32位)	BTC <i>reg, reg</i>	7	????? C
		BTC <i>reg, imm</i>	7	
		BTC <i>mem, reg</i>	13	
		BTC <i>mem, imm</i>	8	
BTR <i>dst, src</i>	位测试并清0 (386+) <i>dst</i> 的指定位 \Rightarrow CF, 然后该位清0, (16/32位)	BTR <i>reg, reg</i>	7	????? C
		BTR <i>reg, imm</i>	7	
		BTR <i>mem, reg</i>	13	
		BTR <i>mem, imm</i>	8	
BTS <i>dst, src</i>	位测试并置1 (386+) <i>dst</i> 的指定位 \Rightarrow CF, 然后该位置1, (16/32位)	BTS <i>reg, reg</i>	7	????? C
		BTS <i>reg, imm</i>	7	
		BTS <i>mem, reg</i>	13	
		BTS <i>mem, imm</i>	8	
CALL <i>dst</i>	子程序调用 近调用: 返回的偏移地址进栈, 然后转至 <i>dst</i> 处执行; 远调用: 返回的段和偏移地址进栈, 然后转至 <i>dst</i> 处执行	CALL <i>label</i> (near)	1	
		CALL <i>reg</i> (near)	2	
		CALL <i>mem</i> (near)	2	
		CALL <i>label</i> (far)	4	
		CALL <i>mem</i> (far)	5	
CBW	AL符号扩展成AX	CBW	3	
CDQ	EAX符号扩展成EDX:EAX	CDQ	2	
CLC	CF清0	CLC	2	----- 0
CLD	DF清0	CLD	2	
CLI	IF清0, 即关中断	CLI	7	
CLTS	清除CR0中任务切换标志 (386+)	CLTS	10	
CMC	CF取反, 即 $\neg CF \Rightarrow CF$	CMC	2	----- C
CMOV _{cc} <i>reg, src</i>	条件成立 $src \Rightarrow reg$, 16/32位 (586+) <i>cc</i> : 参见J _{cc} 指令。	CMOV _{cc} <i>reg, reg</i>	4~9	
		CMOV _{cc} <i>reg, mem</i>		

续表

指令	功能	指令形式	周期数	影响标志位
CMP <i>dst, src</i>	比较: <i>dst-src</i> , 据此设置标志位	CMP <i>reg, reg</i>	1	O S Z A P C
		CMP <i>reg, mem</i>	2	
		CMP <i>reg, imm</i>	1	
		CMP <i>acc, imm</i>	1	
		CMP <i>mem, reg</i>	2	
		CMP <i>mem, imm</i>	2	
CMPS _x	串比较:[(E)SI]-ES:[(E)DI], 然后(E)SI, (E)DI增或减 $\Delta(1/2/4)$ <i>x</i> : B, W, D 对应字节(1)、字(2)、双字(4)。 DF=0 增,否则减	CMPSB	5	O S Z A P C
		CMPSW		
		CMPSD		
CMPXCHG <i>dst, reg</i>	<i>acc-dst</i> , 等 <i>reg</i> ⇒ <i>dst</i> , 否则 <i>dst</i> ⇒ <i>acc</i> (486+)	CMPXCHG <i>reg/mem, reg</i>	5,6	O S Z A P C
CMPXCHG8B <i>dst</i>	EDX:EAX⇒ <i>dst</i> , 等 ECX:EBX ⇒ <i>dst</i> , 否则 EDX:EAX⇒ <i>dst</i> (486+)	CMPXCHG8B <i>mem64</i>	10	--Z---
CPUID	CPU标识⇒EAX,EBX,ECX,EDX	CPUID	14	
CWD	AX符号扩展成DX:AX	CWD	2	
CWDE	AX符号扩展成EAX	CWDE	3	
DAA	加法后的十进制调整 AL	DAA	3	? S Z A P C
DAS	减法后的十进制调整 AL	DAS	3	? S Z A P C
DEC <i>opr</i>	<i>opr</i> 自减 1, 即 <i>opr-1</i> ⇒ <i>opr</i>	DEC <i>reg</i>	1	O S Z A P -
		DEC <i>mem</i>	3	
DIV <i>src</i>	无符号除法 8 位: AX÷ <i>src</i> , 商⇒AL, 余数⇒AH 16 位: DX:AX÷ <i>src</i> , 商⇒AX, 余数⇒DX 32 位: EDX:EAX÷ <i>src</i> , 商⇒EAX, 余数⇒EDX	DIV <i>reg</i>	17~41	??????
		DIV <i>mem</i>		
ENTER <i>m, n</i>	建 <i>m</i> 字节局部空间, <i>n</i> 级的栈帧 (286+)	ENTER <i>imm16, imm8</i>	11+	
HLT	暂停 CPU, 直到 I/O 中断发生	HLT		
IDIV <i>src</i>	有符号除 8 位: AX÷ <i>src</i> , 商⇒AL, 余数⇒AH 16 位: DX:AX÷ <i>src</i> , 商⇒AX, 余数⇒DX 32 位: EDX:EAX÷ <i>src</i> , 商⇒EAX, 余数⇒EDX	IDIV <i>reg</i>	22~46	??????
		IDIV <i>mem</i>		
IMUL <i>src</i>	有符号乘法 8 位: AL× <i>src</i> ⇒AX 16 位: AX× <i>src</i> ⇒DX:AX 32 位: EAX× <i>src</i> ⇒EDX:EAX	IMUL <i>reg</i>	10~11	O ? ? ? ? C
		IMUL <i>mem</i>		
IMUL <i>reg, src</i>	有符号乘法 <i>reg</i> × <i>src</i> ⇒ <i>reg</i> (286+)	IMUL <i>reg, reg/mem</i>	10	O ? ? ? ? C
IMUL <i>reg, src, imm</i>	有符号乘法 <i>src</i> × <i>imm</i> ⇒ <i>reg</i> (286+)	IMUL <i>reg, reg/mem, imm</i>	10	O ? ? ? ? C
IN <i>acc, src</i>	端口数据⇒ <i>acc</i>	IN <i>acc, imm8</i>	7	
		IN <i>acc, DX</i>	7	
INC <i>opr</i>	<i>opr</i> 自加 1, 即 <i>opr+1</i> ⇒ <i>opr</i>	INC <i>reg</i>	1	O S Z A P -
		INC <i>mem</i>	3	



续表

指 令	功 能	指 令 形 式	周期数	影响标志位
INStx	端口 DX 数据⇒ES:[(E)DI], 然后(E)DI 增或减 $\Delta(1/2/4)$ x: B,W, D 对应字节(1)、字(2)、双字(4);若 DF=0 增,否则减	INStB	9	
		INStW		
		INStD		
INTn	FLAGS 进栈,IF,TF 置 0,从[4n]双字单元取 段和偏移地址,并转去执行 (实地址模式)	INT 3	INT+5	
		INT imm8	INT+6	
INTO	若 OF=1, 则执行 INT 4	INTO	4,INT+5	
INVD	使 Cache 无效	INVD	15	
INVLPG	使 TLB 入口无效	INVLPG	29	
IRET	中断返回: 从堆栈弹出返回的偏移 和段地址, 再弹出标志寄存器内容	IRET	7	
Jcc opr	条件满足, 则转移至 opr	Jcc label	1	
JAN/JNBE opr	高于(CF=0∧ZF=0)	JAN/JNBE label		
JAE/JNB/JNC opr	高于等于(CF=0)	JAE/JNB/JNC label		
JB/JC/JNAE opr	低于(CF=1)	JB/JC/JNAE label		
JBE/JNA opr	低于等于(CF=1∨ZF=1)	JBE/JNA label		
JE/JZ opr	等于(ZF=1)	JE/JZ label		
JG/JNLE opr	大于(ZF=0∧SF=OF)	JG/JNLE label		
JGE/JNL opr	大于等于(SF=OF)	JGE/JNL label		
JL/JNGE opr	小于(SF≠OF)	JL/JNGE label		
JLE/JNG opr	小于等于(ZF=1∨SF≠OF)	JLE/JNG label		
JNE/JNZ opr	不等于(ZF=0)	JNE/JNZ label		
JNO opr	无溢出(OF=0)	JNO label		
JNS opr	非负数(SF=0)	JNS label		
JO opr	溢出(OF=1)	JO label		
JP/JPE opr	有偶数个 1(PF=1)	JP/JPE label		
JPO/JNP opr	有奇数个 1(PF=0)	JPO/JNP label		
JS opr	负数(SF=1)	JS label		
JCXZ opr	若 CX=0, 则转移至 opr	JCXZ label	6/5	
JECXZ opr	若 ECX=0, 则转移至 opr	JECXZ label	6/5	
JMP opr	转移至 opr 近: 转移后仅可改变(E)IP 远: 转移后可改变(E)IP 和 CS	JMP label (near)	1	
		JMP reg (near)	2	
		JMP mem (near)	2	
		JMP label (far)	3	
		JMP mem (far)	4	
LAHF	标志寄存器低字节⇒AH	LAHF	2	-----
LAR reg, dst	将 dst 指定的选择器访问权⇒reg	LAR reg, reg/mem	8	--Z---
LDS reg, mem	将 mem 内容⇒DS:reg	LDS reg, mem	4	
LEA reg, mem	将 mem 的偏移地址⇒reg	LEA reg, mem	1	
LEAVE	释放栈帧, 即: (E)BP⇒(E)SP, POP (E)BP	LEAVE	3	
LES reg, mem	将 mem 内容⇒ES:reg	LES reg, mem	4	
LFS reg, mem	将 mem 内容⇒FS:reg (386+)	LFS reg, mem	4	

续表

指 令	功 能	指 令 形 式	周期数	影响标志位
LGDT <i>mem</i>	将 <i>mem</i> 内容 \Rightarrow GDTR (286+)	LGDT <i>mem</i>	6	
LGS <i>reg, mem</i>	将 <i>mem</i> 内容 \Rightarrow GS : <i>reg</i> (386+)	LGS <i>reg, mem</i>	4	
LIDT <i>mem</i>	将 <i>mem</i> 内容 \Rightarrow IDTR (286+)	LIDT <i>mem</i>	6	
LLDT <i>src</i>	<i>src</i> \Rightarrow LDTR (286+)	LLDT <i>reg/mem</i>	8	
LMSW <i>src</i>	<i>src</i> \Rightarrow 机器状态字(CR ₀ 低 16 位) (286+)	LMSW <i>reg/mem</i>	8	
LOCK	总线锁 (以便其他处理器处理指令)	LOCK	1	
LODSx	从串取: [(E)SI] \Rightarrow acc, 然后(E)SI 增或减 $\Delta(1/2/4)$ x: B, W, D 对应字节(1)、字(2)、双字(4);若 DF=0 增,否则减	LODSB	2	
		LODSW		
		LODSD		
LOOP <i>opr</i>	(E)CX 自减 1, 若(E)CX \neq 0 则转移	LOOP <i>label</i>	5/6	
LOOPE/LOOPZ <i>opr</i>	(E)CX 自减 1, 若 ZF=1 \wedge (E)CX \neq 0 则转移	LOOPE/LOOPZ <i>label</i>	7/8	
LOOPNE/LOOPNZ <i>opr</i>	(E)CX 自减 1, 若 ZF=0 \wedge (E)CX \neq 0 则转移	LOOPNE/LOOPNZ <i>label</i>	7/8	
LSL <i>reg, src</i>	<i>src</i> 选择器确定的段 \Rightarrow <i>reg</i> (286+)	LSL <i>reg, reg/mem</i>	8	--Z---
LSS <i>reg, mem</i>	将 <i>mem</i> 内容 \Rightarrow SS : <i>reg</i> (386+)	LSS <i>reg, mem</i>	4	
LTR <i>src</i>	<i>src</i> \Rightarrow 任务寄存器 TR (286+)	LTR <i>reg16/mem16</i>	10	
MOV <i>dst, src</i>	数据传送: <i>src</i> \Rightarrow <i>dst</i>	MOV <i>reg, reg</i>	1	
		MOV <i>reg, mem</i>	1	
		MOV <i>reg, imm</i>	1	
		MOV <i>mem, reg</i>	1	
		MOV <i>mem, imm</i>	1	
		MOV <i>acc, mem</i>	1	
		MOV <i>mem, acc</i>	1	
MOV <i>dst, src</i>	控制寄存器内容传送 (386+) CR _i \Rightarrow reg32, reg32 \Rightarrow CR _i (i=0,2,3,4)	MOV reg32, CR _i	4	
		MOV CR _i , reg32	12~22	
MOV <i>dst, src</i>	调试寄存器内容传送 (386+) DR _i \Rightarrow reg32, reg32 \Rightarrow DR _i (i=0~7)	MOV reg32, DR _i	2~12	
		MOV DR _i , reg32	11~12	
MOV <i>dst, src</i>	段寄存器内容传送 <i>seg</i> \Rightarrow <i>dst</i> , <i>src</i> \Rightarrow <i>seg</i> (CS 除外)	MOV <i>reg/mem, seg</i>	1	
		MOV <i>seg, reg/mem</i>	2~12	
MOVSx	串传送: [(E)SI] \Rightarrow ES:[(E)DI], 然 后(E)SI、(E)DI 增或减 $\Delta(1/2/4)$ x: B, W, D 对应字节(1)、字(2)、双字(4);若 DF=0 增,否则减	MOVSb	4	
		MOVSw		
		MOVSD		
MOVSX <i>reg, src</i>	<i>src</i> 经符号扩展后 \Rightarrow <i>reg</i> (386+)	MOVSX <i>reg, reg/mem</i>	3	
MOVZX <i>reg, src</i>	<i>src</i> 经 0 扩展后 \Rightarrow <i>reg</i> (386+)	MOVZX <i>reg, reg/mem</i>	3	
MUL <i>src</i>	无符号乘法 8 位: AL \times <i>src</i> \Rightarrow AX 16 位: AX \times <i>src</i> \Rightarrow DX:AX 32 位: EAX \times <i>src</i> \Rightarrow EDX:EAX	MUL <i>reg</i>	10~11	O????C
		MUL <i>mem</i>		
NEG <i>opr</i>	<i>opr</i> 求补(负), 即- <i>opr</i> \Rightarrow <i>opr</i>	NEG <i>reg</i>	1	OSZAPC
		NEG <i>mem</i>	3	
NOP	空操作	NOP	1	

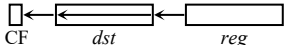

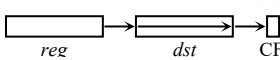
续表

指 令	功 能	指 令 形 式	周期数	影响标志位
NOT <i>opr</i>	<i>opr</i> 按位取反, 即: $\neg opr \Rightarrow opr$	NOT <i>reg</i>	1	O S Z A P C
		NOT <i>mem</i>	3	
OR <i>dst, src</i>	逻辑或, $dst \vee src \Rightarrow dst$	OR <i>reg, reg</i>	1	O S Z ? P O
		OR <i>reg, mem</i>	2	
		OR <i>reg, imm</i>	1	
		OR <i>mem, reg</i>	3	
		OR <i>mem, imm</i>	3	
		OR <i>acc, imm</i>	1	
OUT <i>dst, acc</i>	<i>acc</i> 内容 \Rightarrow 端口 <i>dst</i>	OUT <i>imm8, acc</i>	12	
		OUT DX, <i>acc</i>	12	
OUTSx	[(E)SI]内容 \Rightarrow DX 端口, (386+) 然后(E)SI 增或减 $\Delta(1/2/4)$ x: B, W, D 对应字节(1)、字(2)、双字(4);若 DF=0 增,否则减	OUTSB	13	
		OUTSW		
		OUTSD		
POP <i>dst</i>	从堆栈弹出数据 $\Rightarrow dst$ ((E)SP 增 2 或 4, <i>seg</i> 不能为 CS)	POP <i>reg</i>	1	
		POP <i>mem</i>	3	
		POP <i>seg</i>	3~12	
POPA	数据出栈 \Rightarrow DI, SI, BP, BX, DX, CX, AX (SP 增 2×8) (386+)	POPA	5	
POPAD	堆栈弹出数据 \Rightarrow EDI, ESI, EBP, EBX, EDX, ECX, EAX ((E)SP 增 4×8) (386+)	POPAD	5	
POPF	数据出栈 \Rightarrow FLAGS ((E)SP 增 2) (386+)	POPF	4	O S Z A P C
POPFD	数据出栈 \Rightarrow EFLAGS ((E)SP 增 4) (386+)	POPFD	4	O S Z A P C
PUSH <i>src</i>	<i>src</i> 数据进栈 ((E)SP 减 2/4) (<i>reg32, mem32, imm, 386+</i>)	PUSH <i>reg</i>	1	
		PUSH <i>mem</i>	2	
		PUSH <i>imm</i>	1	
		PUSH <i>seg</i>	1	
PUSHA	AX, CX, DX, BX, SP, BP, SI, DI 进栈, (SP 减 2×8) (386+)	PUSHA	5	
PUSHAD	EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI 进栈, ((E)SP 减 4×8) (386+)	PUSHAD	5	
PUSHF	FLAGS 进栈 ((E)SP 减 2) (386+)	PUSHF	3	
PUSHFD	EFLAGS 进栈 ((E)SP 减 4) (386+)	PUSHFD	3	
RCL <i>dst, n</i>	<i>dst</i> 带进位循环左移 <i>n</i> 位  注: <i>n</i> 为 <i>imm8</i> 是 386+支持	RCL <i>reg, 1</i>	1	O ---- C
		RCL <i>mem, 1</i>	3	
		RCL <i>reg, CL</i>	7~24	
		RCL <i>mem, CL</i>	9~26	
		RCL <i>reg, imm8</i>	8~25	
		RCL <i>mem, imm8</i>	10~27	
RCR <i>dst, n</i>	<i>dst</i> 带进位循环右移 <i>n</i> 位  注: <i>n</i> 为 <i>imm8</i> 是 386+支持	RCR <i>reg, 1</i>	1	O ---- C
		RCR <i>mem, 1</i>	3	
		RCR <i>reg, CL</i>	7~24	
		RCR <i>mem, CL</i>	9~26	
		RCR <i>reg, imm8</i>	8~25	
		RCR <i>mem, imm8</i>	10~27	

续表

指 令	功 能	指 令 形 式	周期数	影响标志位
RDMSR	MSR[ECX]⇒EDX:EAX (586+)	RDMSR	20~24	
RDTSC	自启动以来 CPU 执行的时钟周期数 ⇒EDX:EAX (586+)		28	
REP 串指令	当(E)CX≠0 重复{(E)CX 自减 1, 再执行其后的串指令}	REP INSt	11+3 <i>n</i>	
		REP LODSt	7+3 <i>n</i>	
		REP MOVSt	6,13 <i>n</i>	
		REP OUTSt	13+4 <i>n</i>	
		REP STOS <i>t</i>	6,9+3 <i>n</i>	
REPE/REPZ 串指令	当(E)CX≠0∧ZF=1 重复{(E)CX 自 减 1,再执行其后的串指令}	REPE/REPZ CMPS <i>t</i>	7,8+4 <i>n</i>	O S Z A P C
		REPE/REPZ SCAS <i>t</i>	7,8+4 <i>n</i>	
REPNE/REPNZ 串指令	当(E)CX≠0∧ZF=0 重复{(E)CX 自 减 1,再执行其后的串指令}	REPNE/REPNZ CMPS <i>t</i>	7,8+4 <i>n</i>	O S Z A P C
		REPNE/REPNZ SCAS <i>t</i>	7,8+4 <i>n</i>	
RET [<i>n</i>]	子程序返回: 从堆栈弹出返回地 址, 若有 <i>n</i> 则返回后(E)SP 再增 <i>n</i> 。 近返回 RETN: 只弹出偏移地址; 远返回 RETF: 弹出偏移和段地址;	RETN	2	
		RETF	4	
		RETN imm16	3	
		RETF imm16	4	
ROL <i>dst</i> , <i>n</i>	<i>dst</i> 循环左移 <i>n</i> 位  注: <i>n</i> 为 imm8 是 386+支持	ROL reg, 1	1	O ---- C
		ROL mem, 1	3	
		ROL reg, CL	4	
		ROL mem, CL	4	
		ROL reg, imm8	1	
		ROL mem, imm8	3	
ROR <i>dst</i> , <i>n</i>	<i>dst</i> 循环右移 <i>n</i> 位  注: <i>n</i> 为 imm8 是 386+支持	ROR reg, 1	1	O ---- C
		ROR mem, 1	3	
		ROR reg, CL	4	
		ROR mem, CL	4	
		ROR reg, imm8	1	
		ROR mem, imm8	3	
RSM	从系统管理方式恢复	RSM		O S Z A P C
SAHF	AH⇒标志寄存器的低 8 位	SAHF	1	- S Z A P C
SAL <i>dst</i> , <i>n</i>	<i>dst</i> 算术左移 <i>n</i> 位, 即 $dst \times 2^n \Rightarrow dst$  注: <i>n</i> 为 imm8 时, 386+支持	SAL reg, 1	1	O ---- C
		SAL mem, 1	3	
		SAL reg, CL	4	
		SAL mem, CL	4	
		SAL reg, imm8	1	
		SAL mem, imm8	3	
SAR <i>dst</i> , <i>n</i>	<i>dst</i> 算术右移 <i>n</i> 位, 即 $dst \div 2^n \Rightarrow dst$  注: <i>n</i> 为 imm8 是 386+支持	SAR reg, 1	1	O ---- C
		SAR mem, 1	3	
		SAR reg, CL	4	
		SAR mem, CL	4	
		SAR reg, imm8	1	
		SAR mem, imm8	3	

续表

指 令	功 能	指 令 形 式	周期数	影响标志位
SBB <i>dst, src</i>	带借位减法: $dst - src - CF \Rightarrow dst$	SBB <i>reg, reg</i>	1	O S Z A P C
		SBB <i>reg, mem</i>	2	
		SBB <i>reg, imm</i>	1	
		SBB <i>acc, imm</i>	1	
		SBB <i>mem, reg</i>	3	
		SBB <i>mem, imm</i>	3	
SCASx	串扫描: $acc \Rightarrow ES:[(E)DI]$, 然后(E)DI 增或减 $\Delta(1/2/4)$ x: B, W, D 对应字节(1)、字(2)、双字(4); 若 DF=0 增,否则减	SCASB	4	O S Z A P C
		SCASW		
		SCASD		
SETcc <i>dst</i>	条件真, $1 \Rightarrow dst$, 否则 $0 \Rightarrow dst$, cc 见 Jcc (386+)	SETcc <i>reg8/mem8</i>	3~8	
SGDT <i>mem</i>	GDTR $\Rightarrow mem$ (286+)	SGDT <i>mem</i>	4	
SHL <i>dst, n</i>	<i>dst</i> 逻辑左移 <i>n</i> 位, 与 SAL 相同	SHL/SAL 是一条指令		
SHLD <i>dst, reg, n</i>	双精度左移 (操作数:16/32 位)(386+) 	SHLD <i>reg/mem, reg, imm8</i>	4	? S Z ? P C
		SHLD <i>reg/mem, reg, CL</i>		
SHR <i>dst, n</i>	<i>dst</i> 逻辑右移 <i>n</i> 位  注: <i>n</i> 为 <i>imm8</i> 时, 386+支持	SAR <i>reg, 1</i>	1	O - - - - C
		SAR <i>mem, 1</i>	3	
		SAR <i>reg, CL</i>	4	
		SAR <i>mem, CL</i>	4	
		SAR <i>reg, imm8</i>	1	
		SAR <i>mem, imm8</i>	3	
SHRD <i>dst, reg, n</i>	双精度右移 (操作数:16/32 位)(386+) 	SHLD <i>reg/mem, reg, imm8</i>	4	? S Z ? P C
		SHLD <i>reg/mem, reg, CL</i>		
SIDT <i>mem</i>	IDTR $\Rightarrow mem$	SIDT <i>mem</i>	4	
SLDT <i>dst</i>	LDTR $\Rightarrow dst$	SLDT <i>reg/mem</i>	2	
SMSW <i>dst</i>	机器状态字(CR ₀ 低 16 位) $\Rightarrow dst$ (286+)	SMSW <i>reg/mem</i>	4	
STC	CF 置 1	STC	2	- - - - - 1
STD	DF 置 1	STD	2	
STI	IF 置 1, 即开中断	STI	7	
STOSx	串存入: $acc \Rightarrow ES:[(E)DI]$, 然后(E)DI 增或减 $\Delta(1/2/4)$ x: B, W, D 对应字节(1)、字(2)、双字(4);若 DF=0 增,否则减	STOSB	3	
		STOSW		
		STOSD		
STR <i>dst</i>	任务寄存器 TR $\Rightarrow dst$	STR <i>reg/mem16</i>	2	
SUB <i>dst, src</i>	减法: $dst - src \Rightarrow dst$	SUB <i>reg, reg</i>	1	O S Z A P C
		SUB <i>reg, mem</i>	2	
		SUB <i>reg, imm</i>	1	
		SUB <i>acc, imm</i>	1	
		SUB <i>mem, reg</i>	3	
		SUB <i>mem, imm</i>	3	

续表

指 令	功 能	指 令 形 式	周期数	影响标志位
TEST <i>dst, src</i>	与测试, $dst \wedge src$ 据此设置标志位	TEST <i>reg, reg</i>	2	0 S Z ? P 0
		TEST <i>reg, mem</i>	1	
		TEST <i>reg, imm</i>	1	
		TEST <i>acc, imm</i>	1	
		TEST <i>mem, imm</i>	2	
VERR <i>src</i>	若 <i>src</i> 确定的段可读, $1 \Rightarrow ZF$, 否则 $0 \Rightarrow ZF$	VERR <i>reg/mem</i> 16	7	-- Z ---
VERW <i>src</i>	若 <i>src</i> 确定的段可写, $1 \Rightarrow ZF$, 否则 $0 \Rightarrow ZF$	VERW <i>reg/mem</i> 16	7	-- Z ---
WAIT	等待, 检查挂起未屏蔽的浮点异常	WAIT	1	
WBINVD	写回 Cache, 并使之无效 (486+)	WBINVD	2000+	
WRMSR	EDX:EAX \Rightarrow MSR[ECX] (586+)	WRMSR	30~35	
XADD <i>dst, src</i>	$dst \Leftrightarrow src$, 再 $dst + src \Rightarrow dst$ (486+)	XADD <i>reg/mem, reg</i>	3,4	O S Z A P C
XCHG <i>dst, src</i>	dst, src 内容交换, 即 $dst \Leftrightarrow src$	XCHG <i>reg/mem, reg</i>	3	
		XCHG <i>acc, reg</i>	2	
XLAT/XLATB	查表换码:(E)BX+AL 确定的单元值 \Rightarrow AL	XLAT	4	
XOR <i>dst, src</i>	逻辑异或, $dst \oplus src \Rightarrow dst$	XOR <i>reg, reg</i>	1	0 S Z ? P 0
		XOR <i>reg, mem</i>	2	
		XOR <i>reg, imm</i>	1	
		XOR <i>acc, imm</i>	1	
		XOR <i>mem, reg</i>	3	
		XOR <i>mem, imm</i>	3	

附录 B 编程练习环境说明

1. 编程练习软件包

附带软件包 x86ASM 是在 Microsoft 的 MASM 6.15 软件包的基础上, 加入 CodeView、Win32 的开发工具及 Turbo C 2.0 等, 进行简单整理而成的, 以便初学者编程练习使用。

软件包中的基本文件有:

MASM.EXE	汇编程序
LINK.EXE	连接程序
ML.EXE	汇编连接程序(自动调用 LINK.EXE)
ML.ERR	汇编错误信息文件
LIB.EXE	子程序库管理程序
LIB16.EXE	16 位子程序管理程序
LINK16.EXE	生成 DOS 程序的连接程序
LIB32.EXE	Win32 的库管理程序
LINK32.EXE	生成 Win32 程序的连接程序
CV 目录	CodeView 调试程序 CV.EXE 及相应的环境
INC32 目录	Win32 的 API 的函数库声明文件
LIB32 目录	Win32 的 API 的函数库
TC 目录	Turbo C 2.0 命令行环境和集成环境
SET2ML16.BAT	ML 默认使用 LINK16.EXE 连接程序
SET2ML32.BAT	ML 默认使用 LINK32.EXE 连接程序

使用这个软件包既可以用来练习编写 DOS 环境下的应用程序, 也可以用来练习编写 Win32 环境下的应用程序。

提供 TC 的目的是用它来练习 16 位环境下汇编语言程序模块和 C 程序模块的连接。

2. DOS 系统下的编程练习环境

真正的 DOS 是运行在实模式下的一个操作系统, 所以 DOS 程序是运行在 16 位地址模式下的。这种模式下的程序具有这样的特点:

(1) 偏移地址是 16 位, 所表示的偏移地址只能是 $0 \sim 64K-1$ 。在默认情况下, 指令处理的数据类型是 16 位的, 但也可以处理 32 位数据。

(2) 应用程序可以访问所有的计算机系统资源, 可以使用 I/O 指令直接与外设交换数据, 也可以用 INT 指令调用 DOS 环境下的系统功能(DOS 和 BIOS)。

在 DOS 系统下有很多系统功能调用可用, 但是这里仅将 DOS 环境作为编程练习的平台, 所以只须如下所述的很少几个系统功能就足够了, 主要解决字符的输入、输出, 以及应用程序退出返回。如果读者需要开发 DOS 系统下的应用程序, 则必须另外参阅相关的系统资料手册。

1) 编程练习所用的 DOS 系统调用

(1) 功能 01h。从标准输入设备输入一个字符, 并回显。

入口: AH=01h

出口: AL=输入字符的 ASCII 码

(2) 功能 02h。向标准输出设备输出一个字符。

入口: AH=02h

DL=待输出字符的 ASCII 码

出口: 无

(3) 功能 08h。从标准输入设备输入一个字符, 无回显。

入口: AH=08h

出口: AL=输入字符的 ASCII 码

(4) 功能 09h。输出一个字符串到标准输出设备上。

入口: AH=09h

DS:DX=待输出字符串的地址(字符串须以'\$'作为其结束标志)

出口: 无

(5) 功能 0Ah。从标准输入设备上读入字符串(以回车结束, 有回显)。

入口: AH=0Ah

DS:DX=输入缓冲区地址(字节 0 须填入允许输入字符数)。

出口: 输入缓冲区字节 1 存放输入的字符数, 字节 2 起存放输入的字符串

(6) 功能 0Bh。检查标准输入设备上是否有字符可读。

入口: AH=0Bh

出口: AL=00h——无字符可读; FFh——有字符可读

(7) 功能 4Ch。终止程序的执行, 并可返回一个代码。

入口: AH=4Ch

AL=返回的代码

出口: 无

2) 示例程序 Demo16.ASM

编写程序 Demo16.ASM, 输入一个字符和一个字符串, 并显示。

```

_STACK SEGMENT STACK 'STACK' USE16      ; 定义堆栈段
      DB      2046 DUP(0)                ; 堆栈区长度: 2KB
TOS     DW      0                        ; 初始堆栈栈顶
_STACK ENDS                              ; 堆栈段定义结束
_DATA  SEGMENT 'DATA' USE16              ; 定义数据段
Msg     DB      13, 10, 'Hello, World!', 13, 10, '$'
C1      DB      13, 10, 'Character is: *', 13, 10, '$'
S2      DB      13, 10, 'Buffer content is: '
Buffer  DB      9, 0, 10 DUP('*'), 13, 10, '$'
_DATA  ENDS                              ; 数据段定义结束
_TEXT  SEGMENT 'CODE' USE16              ; 定义代码段
      ASSUME CS: _TEXT, DS: _DATA, SS: _STACK
Start:  MOV     AX, _DATA                  ; 取数据内存区段地址
      MOV     DS, AX                      ; 设置数据段寄存器
      CLI                                           ; 设置堆栈期间禁止响应中断
      MOV     AX, _STACK                  ; 取堆栈内存区段地址
      MOV     SS, AX                      ; 设置堆栈段寄存器

```

```

MOV     SP, Offset TOS           ; 设置初始状态时的堆栈指针
STI                                           ; 堆栈设置完毕允许中断
MOV     DX, Offset Msg
MOV     AH, 9
INT     21h                       ; 中断 21h 的 9 号功能, 显示字符串
MOV     AH, 1
INT     21h
MOV     S2-4, AL
MOV     DX, Offset C1
MOV     AH, 9
INT     21H
MOV     DX, Offset Buffer
MOV     AH, 0Ah
INT     21h
MOV     BL, Buffer[1]
MOV     BH, 0
MOV     Buffer[BX+2], '#'
ADD     Buffer[0], '0'
ADD     Buffer[1], '0'
MOV     DX, Offset S2
MOV     AH, 9
INT     21H
MOV     AX, 4C00h
INT     21h                       ; 运行结束, 返回 DOS
_TEXT   ENDS                       ; 代码段定义结束
END     Start                       ; 源程序到此为止

```

3) 汇编连接

须汇编成 OMF 格式的目标代码(.OBJ), 使用 LINK16.EXE 连接程序。如果 ML 默认使用的是 LINK32.EXE, 那么可执行 SET2ML16, (用 LINK16.EXE 和 LIB16.EXE 覆盖原来的 LINK.EXE 和 LIB.EXE)将 LINK16.EXE 设置成为 ML 默认调用的连接程序。

ML 的 /omf 选项是生成 OMF 格式的目标码, 未指定则默认使用 /omf。

汇编: ML /c Demo16.ASM;

连接: LINK Demo16.OBJ;

或汇编、连接: ML Demo16.ASM。