MC404 Organização de Computadores
e Linguagem de Montagem

IC – UNICAMP

Segundo Trabalho - 2S2008

1 Entrega

O trabalho deverá ser feito em grupo de até 2 alunos. Para entrega do trabalho deverá ser criado um grupo com os alunos membros no sistema teleduc e anexado o(s) arquivo(s) no respectivo portifólio. Favor deixar o(s) arquivo(s) compartilhado(s) somente com os formadores. O prazo de entrega é dia **30/11/2008**.

2 Avaliação

A avaliação será baseada na corretude do programa, qualidade e documentação do código. O grupo também deverá entregar um relatório (em torno de duas páginas) descrevendo os passos que realizaram ao escrever o programa. Também descrever quais foram as principais dificuldades e possíveis sugestões.

3 Descrição do Trabalho

Você deve implementar um programa que, dado um código executável 8086 no formato `.COM` (ver seção 4), gere a listagem em linguagem de montagem desse código em um arquivo de saída. Em resumo, implemente um desmontador de código 8086.

Seu desmontador deve apresentar o seguinte comportamento:

- Aceitar pela linha de comando o nome do arquivo a ser desmontado. Não é necessário verificar o tipo de arquivo;

- Gerar um arquivo de saída com o código em linguagem de montagem do 8086. Seu desmontador deve interpretar corretamente todas as instruções existentes na tabela resumida de instruções usadas no curso, dada na seção 5. Note que basta implementar as instruções do 8086 (não considere as do 386);

- Caso uma instrução inválida seja detectada, seu programa deve copiar o byte indefinido diretamente para saída sem abortar a execução. Por exemplo, supondo que o byte `0xF0` não corresponda a nenhuma instrução, seu programa deve gerar `DB 0xF0` e continuar o processamento a partir desse ponto;

- Não é necessário gerar rótulos para as instruções de salto. Você pode simplesmente gerar os operandos como simples números;

- O arquivo gerado deve ser *compilável* com o NASM ou TASM (favor descrever no relatório qual é suportado). O código remontado deve ser semanticamente idêntico ao original (formato `.COM`).

Como referência para construir seu desmontador use a tabela apresentada na seção 6 que lhe fornece a forma de codificação das instruções no 8086. Na página do curso você encontra programas testes para verificar seu desmontador.

3.1 Extras

Os seguintes itens serão considerados como pontos extras:

- Arquivos do tipo `.EXE` também são interpretados e desmontados corretamente (0,5 ponto);
- Instruções de salto usam rótulos (1 ponto).

4 Formato de Arquivo `.COM`

Um arquivo `.COM` é um arquivo executável que armazena código e dados em um único segmento (e portanto restrito a 64Kb). O DOS sempre carrega este arquivo no offset inicial `0x0100`. Todos os registradores de segmentos apontam para o mesmo segmento, sendo que o topo da pilha aponta sempre para o final da memória. Portanto, não é necessário ajustar os segmentos de código, dado e pilha.

Esse formato não possui nenhum cabeçalho. O primeiro byte do arquivo é o primeiro byte da primeira instrução a ser interpretada quando o arquivo for carregado para execução. Como o endereço inicial é 0x100, é necessário usar uma diretiva do montador para forçar o código a iniciar nesse endereço. No NASM, teríamos:

```
  org 100h

  section .text

start:
  ; put your code here

section .data
  ; put data items here

section .bss
  ; put uninitialised data here
```

Na hora de gerar o executável a seguinte linha de comando é usada:

```
nasm myprog.asm -fbin -o myprog.com
```

Mais informações podem ser encontradas nas seções 6.1 e 7.2.1 do manual do NASM.

5 Tabela Resumida de Instruções

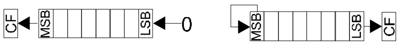
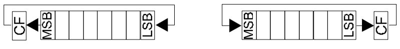
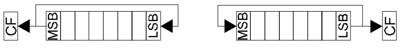
Intel Assembler 80186 and higher

CodeTable 1/2

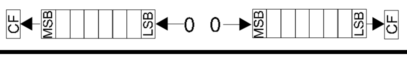
V 2.3 English. Also available in Spanish

TRANSFER				Flags								
Name	**Comment**	**Code**	**Operation**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
MOV	Move (copy)	MOV Dest,Source	Dest:=Source									
XCHG	Exchange	XCHG Op1,Op2	Op1:=Op2 , Op2:=Op1									
STC	Set Carry	STC	CF:=1									1
CLC	Clear Carry	CLC	CF:=0									0
CMC	Complement Carry	CMC	CF:= Ø CF									±
STD	Set Direction	STD	DF:=1 (string op's downwards)		1							
CLD	Clear Direction	CLD	DF:=0 (string op's upwards)		0							
STI	Set Interrupt	STI	IF:=1			1						
CLI	Clear Interrupt	CLI	IF:=0			0						
PUSH	Push onto stack	PUSH Source	DEC SP, [SP]:=Source									
PUSHF	Push flags	PUSHF	O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL									
PUSHA	Push all general registers	PUSHA	AX, CX, DX, BX, SP, BP, SI, DI									
POP	Pop from stack	POP Dest	Dest:=[SP], INC SP									
POPF	Pop flags	POPF	O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL	±	±	±	±	±	±	±	±	±
POPA	Pop all general registers	POPA	DI, SI, BP, SP, BX, DX, CX, AX									
CBW	Convert byte to word	CBW	AX:=AL (signed)									
CWD	Convert word to double	CWD	DX:AX:=AX (signed)	±				±	±	±	±	±
CWDE	Conv word extended double	CWDE 386	EAX:=AX (signed)									
IN *i*	Input	IN Dest, Port	AL/AX/EAX := byte/word/double of specified port									
OUT *i*	Output	OUT Port, Source	Byte/word/double of specified port := AL/AX/EAX									

i for more information see instruction specifications Flags: ±=affected by this instruction ?=undefined after this instruction

ARITHMETIC				Flags								
Name	**Comment**	**Code**	**Operation**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
ADD	Add	ADD Dest,Source	Dest:=Dest+Source	±				±	±	±	±	±
ADC	Add with Carry	ADC Dest,Source	Dest:=Dest+Source+CF	±				±	±	±	±	±
SUB	Subtract	SUB Dest,Source	Dest:=Dest-Source	±				±	±	±	±	±
SBB	Subtract with borrow	SBB Dest,Source	Dest:=Dest-(Source+CF)	±				±	±	±	±	±
DIV	Divide (unsigned)	DIV Op	Op=byte: AL:=AX / Op AH:=Rest	?				?	?	?	?	?
DIV	Divide (unsigned)	DIV Op	Op=word: AX:=DX:AX / Op DX:=Rest	?				?	?	?	?	?
DIV 386	Divide (unsigned)	DIV Op	Op=doublew.: EAX:=EDX:EAX / Op EDX:=Rest	?				?	?	?	?	?
IDIV	Signed Integer Divide	IDIV Op	Op=byte: AL:=AX / Op AH:=Rest	?				?	?	?	?	?
IDIV	Signed Integer Divide	IDIV Op	Op=word: AX:=DX:AX / Op DX:=Rest	?				?	?	?	?	?
IDIV 386	Signed Integer Divide	IDIV Op	Op=doublew.: EAX:=EDX:EAX / Op EDX:=Rest	?				?	?	?	?	?
MUL	Multiply (unsigned)	MUL Op	Op=byte: AX:=AL*Op if AH=0 ♦	±				?	?	?	?	±
MUL	Multiply (unsigned)	MUL Op	Op=word: DX:AX:=AX*Op if DX=0 ♦	±				?	?	?	?	±
MUL 386	Multiply (unsigned)	MUL Op	Op=double: EDX:EAX:=EAX*Op if EDX=0 ♦	±				?	?	?	?	±
IMUL *i*	Signed Integer Multiply	IMUL Op	Op=byte: AX:=AL*Op if AL sufficient ♦	±				?	?	?	?	±
IMUL	Signed Integer Multiply	IMUL Op	Op=word: DX:AX:=AX*Op if AX sufficient ♦	±				?	?	?	?	±
IMUL 386	Signed Integer Multiply	IMUL Op	Op=double: EDX:EAX:=EAX*Op if EAX sufficient ♦	±				?	?	?	?	±
INC	Increment	INC Op	Op:=Op+1 (Carry not affected !)	±				±	±	±	±	
DEC	Decrement	DEC Op	Op:=Op-1 (Carry not affected !)	±				±	±	±	±	
CMP	Compare	CMP Op1,Op2	Op1-Op2	±				±	±	±	±	±
SAL	Shift arithmetic left (≡ SHL)	SAL Op,Quantity		*i*				±	±	?	±	±
SAR	Shift arithmetic right	SAR Op,Quantity		*i*				±	±	?	±	±
RCL	Rotate left through Carry	RCL Op,Quantity		*i*								±
RCR	Rotate right through Carry	RCR Op,Quantity		*i*								±
ROL	Rotate left	ROL Op,Quantity		*i*								±
ROR	Rotate right	ROR Op,Quantity		*i*								±

i for more information see instruction specifications ♦ then CF:=0, OF:=0 else CF:=1, OF:=1

LOGIC				Flags								
Name	**Comment**	**Code**	**Operation**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
NEG	Negate (two-complement)	NEG Op	Op:=0-Op if Op=0 then CF:=0 else CF:=1	±				±	±	±	±	±
NOT	Invert each bit	NOT Op	Op:=Ø Op (invert each bit)									
AND	Logical and	AND Dest,Source	Dest:=DestÙ Source	0				±	±	?	±	0
OR	Logical or	OR Dest,Source	Dest:=DestÚSource	0				±	±	?	±	0
XOR	Logical exclusive or	XOR Dest,Source	Dest:=Dest (exor) Source	0				±	±	?	±	0
SHL	Shift logical left (≡ SAL)	SHL Op,Quantity		*i*				±	±	?	±	±
SHR	Shift logical right	SHR Op,Quantity		*i*				±	±	?	±	±

Intel Assembler 80186 and higher

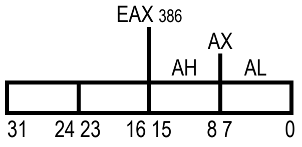
CodeTable 2/2

MISC												
				Flags								
Name	**Comment**	**Code**	**Operation**	**O**	**D**	**I**	**T**	**S**	**Z**	**A**	**P**	**C**
NOP	No operation	NOP	No operation									
LEA	Load effective address	LEA Dest,Source	Dest := address of Source									
INT	Interrupt	INT Nr	interrupts current program, runs spec. int-program			0	0					

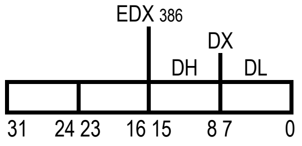
JUMPS (flags remain unchanged)							
Name	**Comment**	**Code**	**Operation**	**Name**	**Comment**	**Code**	**Operation**
CALL	Call subroutine	CALL Proc		RET	Return from subroutine	RET	
JMP	Jump	JMP Dest					
JE	Jump if Equal	JE Dest	(≡ JZ)	JNE	Jump if not Equal	JNE Dest	(≡ JNZ)
JZ	Jump if Zero	JZ Dest	(≡ JE)	JNZ	Jump if not Zero	JNZ Dest	(≡ JNE)
JCXZ	Jump if CX Zero	JCXZ Dest		JECXZ	Jump if ECX Zero	JECXZ Dest	386
JP	Jump if Parity (Parity Even)	JP Dest	(≡ JPE)	JNP	Jump if no Parity (Parity Odd)	JNP Dest	(≡ JPO)
JPE	Jump if Parity Even	JPE Dest	(≡ JP)	JPO	Jump if Parity Odd	JPO Dest	(≡ JNP)

JUMPS Unsigned (Cardinal)				**JUMPS Signed** (Integer)			
JA	Jump if Above	JA Dest	(≡ JNBE)	JG	Jump if Greater	JG Dest	(≡ JNLE)
JAE	Jump if Above or Equal	JAE Dest	(≡ JNB ≡ JNC)	JGE	Jump if Greater or Equal	JGE Dest	(≡ JNL)
JB	Jump if Below	JB Dest	(≡ JNAE ≡ JC)	JL	Jump if Less	JL Dest	(≡ JNGE)
JBE	Jump if Below or Equal	JBE Dest	(≡ JNA)	JLE	Jump if Less or Equal	JLE Dest	(≡ JNG)
JNA	Jump if not Above	JNA Dest	(≡ JBE)	JNG	Jump if not Greater	JNG Dest	(≡ JLE)
JNAE	Jump if not Above or Equal	JNAE Dest	(≡ JB ≡ JC)	JNGE	Jump if not Greater or Equal	JNGE Dest	(≡ JL)
JNB	Jump if not Below	JNB Dest	(≡ JAE ≡ JNC)	JNL	Jump if not Less	JNL Dest	(≡ JGE)
JNBE	Jump if not Below or Equal	JNBE Dest	(≡ JA)	JNLE	Jump if not Less or Equal	JNLE Dest	(≡ JG)
JC	Jump if Carry	JC Dest		JO	Jump if Overflow	JO Dest	
JNC	Jump if no Carry	JNC Dest		JNO	Jump if no Overflow	JNO Dest	
				JS	Jump if Sign (= negative)	JS Dest	
				JNS	Jump if no Sign (= positive)	JNS Dest	

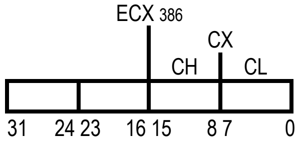
General Registers:



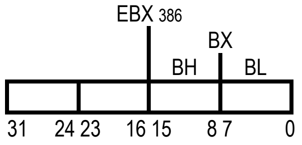
Accumulator



Data mul, div, IO



Count loop, shift



BaseX data ptr

Example:

```
        .DOSSEG              ; Demo program
        .MODEL SMALL
        .STACK 1024
Two     EQU 2                ; Const
        .DATA
VarB    DB ?                 ; define Byte, any value
VarW    DW 1010b             ; define Word, binary
VarW2   DW 257               ; define Word, decimal
VarD    DD 0AFFFFh           ; define Doubleword, hex
S       DB "Hello !",0       ; define String
        .CODE
main:   MOV AX,DGROUP        ; resolved by linker
        MOV DS,AX            ; init datasegment reg
        MOV [VarB],42        ; init VarB
        MOV [VarD],-7        ; set VarD
        MOV BX,Offset[S]     ; addr of "H" of "Hello !"
        MOV AX,[VarW]        ; get value into accumulator
        ADD AX,[VarW2]       ; add VarW2 to AX
        MOV [VarW2],AX       ; store AX in VarW2
        MOV AX,4C00h         ; back to system
        INT 21h
        END main
```

KEEP IT SIMPLE, STUPID (AND SHORT!) - KISS

Flags: | - | - | - | - | O | D | I | T | S | Z | - | A | - | P | - | C |

Control Flags (how instructions are carried out):

D: Direction	1 = string op's process down from high to low address
I: Interrupt	whether interrupts can occur. 1= enabled
T: Trap	single step for debugging

Status Flags (result of operations):

C: Carry	result of unsigned op. is too large or below zero. 1 = carry/borrow
O: Overflow	result of signed op. is too large or small. 1 = overflow/underflow
S: Sign	sign of result. Reasonable for Integer only. 1 = neg. / 0 = pos.
Z: Zero	result of operation is zero. 1 = zero
A: Aux. carry	similar to Carry but restricted to the low nibble only
P: Parity	1 = result has even number of set bits

6 Tabela com Opcodes

APPENDIX D
INSTRUCTION SET OPCODES AND CLOCK CYCLES

This appendix provides reference information for the 80C186 Modular Core family instruction set. Table D-1 defines the variables used in Table D-2, which lists the instructions with their formats and execution times. Table D-3 is a guide for decoding machine instructions. Table D-4 is a guide for encoding instruction mnemonics, and Table D-5 defines Table D-4 abbreviations.

Table D-1. Operand Variables

Variable	Description
mod	*mod* and *r/m* determine the Effective Address (EA).
r/m	*r/m* and *mod* determine the Effective Address (EA).
reg	*reg* represents a register.
MMM	*MMM* and *PPP* are opcodes to the math coprocessor.
PPP	*PPP* and *MMM* are opcodes to the math coprocessor.
TTT	*TTT* defines which shift or rotate instruction is executed.

r/m	EA Calculation
0 0 0	(BX) + (SI) + DISP
0 0 1	(BX) + (DI) + DISP
0 1 0	(BP) + (SI) + DISP
0 1 1	(BP) + (DI) + DISP
1 0 0	(SI) + DISP
1 0 1	(DI) + DISP
1 1 0	(BP) + DISP, if mod €≠00 disp-high:disp-low, if mod =00
1 1 1	(BX) + DISP

mod	Effect on EA Calculation
0 0	if r/m €≠110, DISP = 0; disp-low and disp-high are absent
0 0	if r/m = 110, EA = disp-high:disp-low
0 1	DISP = disp-low, sign-extended to 16 bits; disp-high is absent
1 0	DISP = disp-high:disp-low
1 1	r/m is treated as a reg field

DISP follows the second byte of the instruction (before any required data).

Physical addresses of operands addressed by the BP register are computed using the SS segment register. Physical addresses of destination operands of string primitives (addressed by the DI register) are computed using the ES segment register, which cannot be overridden.

reg	16-bit (w=1)	8-bit (w=0)
0 0 0	AX	AL
0 0 1	CX	CL
0 1 0	DX	DL
0 1 1	BP	BL
1 0 0	SP	AH
1 0 1	BP	CH
1 1 0	SI	DH
1 1 1	DI	BH

TTT	Instruction
0 0 0	ROL
0 0 1	ROR
0 1 0	RCL
0 1 1	RCR
1 0 0	SHL/SAL
1 0 1	SHR
1 1 0	—
1 1 1	SAR

Table D-2. Instruction Set Summary

Function	Format				Clocks	Notes
DATA TRANSFER INSTRUCTIONS						
MOV = Move						
register to register/memory	1 0 0 0 1 0 0 w	mod reg r/m			2/12	
register/memory to register	1 0 0 0 1 0 1 w	mod reg r/m			2/9	
immediate to register/memory	1 1 0 0 0 1 1 w	mod 000 r/m	data	data if w=1	12/13	(1)
immediate to register	1 0 1 1 w reg	data	data if w=1		3/4	(1)
memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high		9	
accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high		8	
register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m			2/9	
segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m			2/11	
PUSH = Push						
memory	1 1 1 1 1 1 1 1	mod 110 r/m			16	
register	0 1 0 1 0 reg				10	
segment register	0 0 0 reg 1 1 0				9	
immediate	0 1 1 0 1 0 s 0	data	data if s=0		10	
POP = Pop						
memory	1 0 0 0 1 1 1 1	mod 000 r/m			20	
register	0 1 0 1 1 reg				10	
segment register	0 0 0 reg 1 1 1	(reg ?01)			8	
PUSHA = Push all	0 1 1 0 0 0 0 0				36	
POPA = Pop all	0 1 1 0 0 0 0 1				51	
XCHG = Exchange						
register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m			4/17	
register with accumulator	1 0 0 1 0 reg				3	
XLAT = Translate byte to AL	1 1 0 1 0 1 1 1				11	
IN = Input from						
fixed port	1 1 1 0 0 1 0 w	port			10	
variable port	1 1 1 0 1 1 0 w				8	
OUT = Output from						
fixed port	1 1 1 0 0 1 0 w	port			9	
variable port	1 1 1 0 1 1 0 w				7	

NOTES:

1. Clock cycles are given for 8-bit/16-bit operations.
2. Clock cycles are given for jump not taken/jump taken.
3. Clock cycles are given for interrupt taken/interrupt not taken.
4. If $\overline{\text{TEST}}$ = 0

Shading indicates additions and enhancements to the 8086/8088 instruction set. See Appendix A, “80C186 Instruction Set Additions and Extensions,” for details.

Table D-2. Instruction Set Summary (Continued)

Function	Format				Clocks	Notes
DATA TRANSFER INSTRUCTIONS (Continued)						
LEA = Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m			6	
LDS = Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m	(mod ?11)		18	
LES = Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m	(mod ?11)		18	
ENTER = Build stack frame	1 1 0 0 1 0 0 0	data-low	data-high	L		
L = 0					15	
L = 1					25	
L > 1					22+16(n-1)	
LEAVE = Tear down stack frame	1 1 0 0 1 0 0 1				8	
LAHF = Load AH with flags	1 0 0 1 1 1 1 1				2	
SAHF = Store AH into flags	1 0 0 1 1 1 1 0				3	
PUSHF = Push flags	1 0 0 1 1 1 0 0				9	
POPF = Pop flags	1 0 0 1 1 1 0 1				8	
ARITHMETIC INSTRUCTIONS						
ADD = Add						
reg/memory with register to either	0 0 0 0 0 0 d w	mod reg r/m			3/10	
immediate to register/memory	1 0 0 0 0 0 s w	mod 000 r/m	data	data if sw=01	4/16	
immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1		3/4	(1)
ADC = Add with carry						
reg/memory with register to either	0 0 0 1 0 0 d w	mod reg r/m			3/10	
immediate to register/memory	1 0 0 0 0 0 s w	mod 010 r/m	data	data if sw=01	4/16	
immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1		3/4	(1)
INC = Increment						
register/memory	1 1 1 1 1 1 1 w	mod 000 r/m			3/15	
register	0 1 0 0 0 reg				3	
AAA = ASCII adjust for addition	0 0 1 1 0 1 1 1				8	
DAA = Decimal adjust for addition	0 0 1 0 0 1 1 1				4	

NOTES:

1. Clock cycles are given for 8-bit/16-bit operations.
2. Clock cycles are given for jump not taken/jump taken.
3. Clock cycles are given for interrupt taken/interrupt not taken.
4. If $\overline{TEST}$ = 0

Shading indicates additions and enhancements to the 8086/8088 instruction set. See Appendix A, “80C186 Instruction Set Additions and Extensions,” for details.

Table D-2. Instruction Set Summary (Continued)

Function	Format				Clocks	Notes
ARITHMETIC INSTRUCTIONS (Continued)						
SUB = Subtract						
reg/memory with register to either	0 0 1 0 1 0 d w	mod reg r/m			3/10	
immediate from register/memory	1 0 0 0 0 0 s w	mod 101 r/m	data	data if sw=01	4/16	
immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1		3/4	(1)
SBB = Subtract with borrow						
reg/memory with register to either	0 0 0 1 1 0 d w	mod reg r/m			3/10	
immediate from register/memory	1 0 0 0 0 0 s w	mod 011 r/m	data	data if sw=01	4/16	
immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1		3/4	(1)
DEC = Decrement						
register/memory	1 1 1 1 1 1 1 w	mod 001 r/m			3/15	
register	0 1 0 0 1 reg				3	
NEG = Change sign	1 1 1 1 0 1 1 w	mod reg r/m			3	
CMP = Compare						
register/memory with register	0 0 1 1 1 0 1 w	mod reg r/m			3/10	
register with register/memory	0 0 1 1 1 0 0 w	mod reg r/m			3/10	
immediate with register/memory	1 0 0 0 0 0 s w	mod 111 r/m	data	data if sw=01	3/10	
immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w=1		3/4	(1)
AAS = ASCII adjust for subtraction	0 0 1 1 1 1 1 1				7	
DAS = Decimal adjust for subtraction	0 0 1 0 1 1 1 1				4	
MUL = multiply (unsigned)	1 1 1 1 0 1 1 w	mod 100 r/m				
register-byte					26-28	
register-word					35-37	
memory-byte					32-34	
memory-word					41-43	
IMUL = Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 101 r/m				
register-byte					25-28	
register-word					34-37	
memory-byte					31-34	
memory-word					40-43	
integer **immediate** multiply (signed)	0 1 1 0 1 0 s 1	mod reg r/m	data	data if s=0	22-25/	
					29-32	

NOTES:

1. Clock cycles are given for 8-bit/16-bit operations.
2. Clock cycles are given for jump not taken/jump taken.
3. Clock cycles are given for interrupt taken/interrupt not taken.
4. If $\overline{\text{TEST}}$ = 0

Shading indicates additions and enhancements to the 8086/8088 instruction set. See Appendix A, “80C186 Instruction Set Additions and Extensions,” for details.

Table D-2. Instruction Set Summary (Continued)

Function	Format				Clocks	Notes
ARITHMETIC INSTRUCTIONS (Continued)						
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0			19	
DIV = Divide (unsigned)	1 1 1 1 0 1 1 w	mod 110 r/m				
register-byte					29	
register-word					38	
memory-byte					35	
memory-word					44	
IDIV = Integer divide (signed)	1 1 1 1 0 1 1 w	mod 111 r/m				
register-byte					29	
register-word					38	
memory-byte					35	
memory-word					44	
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0			15	
CBW = Convert byte to word	1 0 0 1 1 0 0 0				2	
CWD = Convert word to double-word	1 0 0 1 1 0 0 1				4	
BIT MANIPULATION INSTRUCTIONS						
NOT= Invert register/memory	1 1 1 1 0 1 1 w	mod 010 r/m			3	
AND = And						
reg/memory and register to either	0 0 1 0 0 0 d w	mod reg r/m			3/10	
immediate to register/memory	1 0 0 0 0 0 0 w	mod 100 r/m	data	data if w=1	4/16	
immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1		3/4	(1)
OR = Or						
reg/memory and register to either	0 0 0 0 1 0 d w	mod reg r/m			3/10	
immediate to register/memory	1 0 0 0 0 0 0 w	mod 001 r/m	data	data if w=1	4/10	
immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1		3/4	(1)
XOR = Exclusive or						
reg/memory and register to either	0 0 1 1 0 0 d w	mod reg r/m			3/10	
immediate to register/memory	1 0 0 0 0 0 0 w	mod 110 r/m	data	data if w=1	4/10	
immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1		3/4	(1)

NOTES:

1. Clock cycles are given for 8-bit/16-bit operations.
2. Clock cycles are given for jump not taken/jump taken.
3. Clock cycles are given for interrupt taken/interrupt not taken.
4. If $\overline{\text{TEST}}$ = 0

Shading indicates additions and enhancements to the 8086/8088 instruction set. See Appendix A, “80C186 Instruction Set Additions and Extensions,” for details.

Table D-2. Instruction Set Summary (Continued)

Function	Format				Clocks	Notes
BIT MANIPULATION INSTRUCTIONS (Continued)						
TEST= And function to flags, no result						
register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m			3/10	
immediate data and register/memory	1 1 1 1 0 1 1 w	mod 000 r/m	data	data if w=1	4/10	
immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1		3/4	(1)
Shifts/Rotates						
register/memory by 1	1 1 0 1 0 0 0 w	mod TTT r/m			2/15	
register/memory by CL	1 1 0 1 0 0 1 w	mod TTT r/m			5+n/17+n	
register/memory by Count	1 1 0 0 0 0 0 w	mod TTT r/m	count		5+n/17+n	
STRING MANIPULATION INSTRUCTIONS						
MOVS = Move byte/word	1 0 1 0 0 1 0 w				14	
INS = Input byte/word from DX port	0 1 1 0 1 1 0 w				14	
OUTS = Output byte/word to DX port	0 1 1 0 1 1 1 w				14	
CMPS = Compare byte/word	1 0 1 0 0 1 1 w				22	
SCAS = Scan byte/word	1 0 1 0 1 1 1 w				15	
LODS = Load byte/word to AL/AX	1 0 1 0 1 1 0 w				12	
STOS = Store byte/word from AL/AX	1 0 1 0 1 0 1 w				10	
Repeated by count in CX:						
MOVS = Move byte/word	1 1 1 1 0 0 1 0	1 0 1 0 0 1 0 w			8+8n	
INS = Input byte/word from DX port	1 1 1 1 0 0 1 0	0 1 1 0 1 1 0 w			8-8n	
OUTS = Output byte/word to DX port	1 1 1 1 0 0 1 0	0 1 1 0 1 1 1 w			8+8n	
CMPS = Compare byte/word	1 1 1 1 0 0 1 z	1 0 1 0 0 1 1 w			5+22n	
SCAS = Scan byte/word	1 1 1 1 0 0 1 z	1 0 1 0 1 1 1 w			5+15n	
LODS = Load byte/word to AL/AX	1 1 1 1 0 0 1 0	0 1 0 1 0 0 1 w			6+11n	
STOS = Store byte/word from AL/AX	1 1 1 1 0 1 0 0	0 1 0 1 0 0 1 w			6+9n	

NOTES:

1. Clock cycles are given for 8-bit/16-bit operations.
2. Clock cycles are given for jump not taken/jump taken.
3. Clock cycles are given for interrupt taken/interrupt not taken.
4. If $\overline{\text{TEST}}$ = 0

Shading indicates additions and enhancements to the 8086/8088 instruction set. See Appendix A, "80C186 Instruction Set Additions and Extensions," for details.

Table D-2. Instruction Set Summary (Continued)

Function	Format			Clocks	Notes
PROGRAM TRANSFER INSTRUCTIONS					
Conditional Transfers — jump if:					
JE/JZ= equal/zero	0 1 1 1 0 1 0 0	disp		4/13	(2)
JL/JNGE = less/not greater or equal	0 1 1 1 1 1 0 0	disp		4/13	(2)
JLE/JNG = less or equal/not greater	0 1 1 1 1 1 1 0	disp		4/13	(2)
JB/JNAE = below/not above or equal	0 1 1 1 0 0 1 0	disp		4/13	(2)
JC = carry	0 1 1 1 0 0 1 0	disp		4/13	(2)
JBE/JNA = below or equal/not above	0 1 1 1 0 1 1 0	disp		4/13	(2)
JP/JPE = parity/parity even	0 1 1 1 1 0 1 0	disp		4/13	(2)
JO = overflow	0 1 1 1 0 0 0 0	disp		4/13	(2)
JS = sign	0 1 1 1 1 0 0 0	disp		4/13	(2)
JNE/JNZ = not equal/not zero	0 1 1 1 0 1 0 1	disp		4/13	(2)
JNL/JGE = not less/greater or equal	0 1 1 1 1 1 0 1	disp		4/13	(2)
JNLE/JG = not less or equal/greater	0 1 1 1 1 1 1 1	disp		4/13	(2)
JNB/JAE = not below/above or equal	0 1 1 1 0 0 1 1	disp		4/13	(2)
JNC = not carry	0 1 1 1 0 0 1 1	disp		4/13	(2)
JNBE/JA = not below or equal/above	0 1 1 1 0 1 1 1	disp		4/13	(2)
JNP/JPO = not parity/parity odd	0 1 1 1 1 0 1 1	disp		4/13	(2)
JNO = not overflow	0 1 1 1 0 0 0 1	disp		4/13	(2)
JNS = not sign	0 1 1 1 1 0 0 1	disp		5/15	(2)
Unconditional Transfers					
CALL = Call procedure					
direct within segment	1 1 1 0 1 0 0 0	disp-low	disp-high	15	
reg/memory indirect within segment	1 1 1 1 1 1 1 1	mod 010 r/m		13/19	
indirect intersegment	1 1 1 1 1 1 1 1	mod 011 r/m	(mod ?11)	38	
direct intersegment	1 0 0 1 1 0 1 0	segment offset		23	
		selector			

NOTES:
1. Clock cycles are given for 8-bit/16-bit operations.
2. Clock cycles are given for jump not taken/jump taken.
3. Clock cycles are given for interrupt taken/interrupt not taken.
4. If $\overline{\text{TEST}}$ = 0

Shading indicates additions and enhancements to the 8086/8088 instruction set. See Appendix A, “80C186 Instruction Set Additions and Extensions,” for details.

Table D-2. Instruction Set Summary (Continued)

Function	Format			Clocks	Notes
PROGRAM TRANSFER INSTRUCTIONS (Continued)					
RET = Return from procedure					
within segment	1 1 0 0 0 0 1 1			16	
within segment adding immed to SP	1 1 0 0 0 0 1 0	data-low	data-high	18	
intersegment	1 1 0 0 1 0 1 1			22	
intersegment adding immed to SP	1 1 0 0 1 0 1 0	data-low	data-high	25	
JMP = Unconditional jump					
short/long	1 1 1 0 1 0 1 1	disp-low		14	
direct within segment	1 1 1 0 1 0 0 1	disp-low	disp-high	14	
reg/memory indirect within segment	1 1 1 1 1 1 1 1	mod 100 r/m		26	
indirect intersegment	1 1 1 1 1 1 1 1	mod 101 r/m	(mod ?11)	11/17	
direct intersegment	1 1 1 0 1 0 1 0	segment offset		14	
		selector			
Iteration Control					
LOOP = Loop CX times	1 1 1 0 0 0 1 0	disp		6/16	(2)
LOOPZ/LOOPE =Loop while zero/equal	1 1 1 0 0 0 0 1	disp		5/16	(2)
LOOPNZ/LOOPNE = Loop while not zero/not equal	1 1 1 0 0 0 0 0	disp		5/16	(2)
JCXZ = Jump if CX = zero	1 1 1 0 0 0 1 1	disp		6/16	(2)
Interrupts					
INT = Interrupt					
Type specified	1 1 0 0 1 1 0 1	type		47	
Type 3	1 1 0 0 1 1 0 0			45	
INTO = Interrupt on overflow	1 1 0 0 1 1 1 0			48/4	(3)
BOUND = Detect value out of range	0 1 1 0 0 0 1 0	mod reg r/m		33-35	
IRET = Interrupt return	1 1 0 0 1 1 1 1			28	

NOTES:

1. Clock cycles are given for 8-bit/16-bit operations.
2. Clock cycles are given for jump not taken/jump taken.
3. Clock cycles are given for interrupt taken/interrupt not taken.
4. If $\overline{\text{TEST}}$ = 0

Shading indicates additions and enhancements to the 8086/8088 instruction set. See Appendix A, “80C186 Instruction Set Additions and Extensions,” for details.

Table D-2. Instruction Set Summary (Continued)

Function	Format		Clocks	Notes
PROCESSOR CONTROL INSTRUCTIONS				
CLC = Clear carry	1 1 1 1 1 0 0 0		2	
CMC = Complement carry	1 1 1 1 0 1 0 1		2	
STC = Set carry	1 1 1 1 1 0 0 1		2	
CLD = Clear direction	1 1 1 1 1 1 0 0		2	
STD = Set direction	1 1 1 1 1 1 0 1		2	
CLI = Clear interrupt	1 1 1 1 1 0 1 0		2	
STI = Set interrupt	1 1 1 1 1 0 1 1		2	
HLT = Halt	1 1 1 1 0 1 0 0		2	
WAIT = Wait	1 0 0 1 1 0 1 1		6	(4)
LOCK = Bus lock prefix	1 1 1 1 0 0 0 0		2	
ESC = Math coprocessor escape	1 1 0 1 1 M M M	mod PPP r/m	6	
NOP = No operation	1 0 0 1 0 0 0 0		3	
SEGMENT OVERRIDE PREFIX				
CS	0 0 1 0 1 1 1 0		2	
SS	0 0 1 1 0 1 1 0		2	
DS	0 0 1 1 1 1 1 0		2	
ES	0 0 1 0 0 1 1 0		2	

NOTES:

1. Clock cycles are given for 8-bit/16-bit operations.
2. Clock cycles are given for jump not taken/jump taken.
3. Clock cycles are given for interrupt taken/interrupt not taken.
4. If $\overline{\text{TEST}}$ = 0

Shading indicates additions and enhancements to the 8086/8088 instruction set. See Appendix A, “80C186 Instruction Set Additions and Extensions,” for details.

Table D-3. Machine Instruction Decoding Guide

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	**Binary**				
00	0000 0000	mod reg r/m	(disp-lo),(disp-hi)	add	reg8/mem8, reg8
01	0000 0001	mod reg r/m	(disp-lo),(disp-hi)	add	reg16/mem16,reg16
02	0000 0010	mod reg r/m	(disp-lo),(disp-hi)	add	reg8,reg8/mem8
03	0000 0011	mod reg r/m	(disp-lo),(disp-hi)	add	reg16,reg16/mem16
04	0000 0100	data-8		add	AL,immed8
05	0000 0101	data-lo	data-hi	add	AX,immed16
06	0000 0110			push	ES
07	0000 0111			pop	ES
08	0000 0100	mod reg r/m	(disp-lo),(disp-hi)	or	reg8/mem8,reg8

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
09	0000 1001	mod reg r/m	(disp-lo),(disp-hi)	or	reg16/mem16,reg16
0A	0000 1010	mod reg r/m	(disp-lo),(disp-hi)	or	reg8,reg8/mem8
0B	0000 1011	mod reg r/m	(disp-lo),(disp-hi)	or	reg16,reg16/mem16
0C	0000 1100	data-8		or	AL, immed8
0D	0000 1101	data-lo	data-hi	or	AX,immed16
0E	0000 1110			push	CS
0F	0000 1111			—	
10	0001 0000	mod reg r/m	(disp-lo),(disp-hi)	adc	reg8/mem8,reg8
11	0001 0001	mod reg r/m	(disp-lo),(disp-hi)	adc	reg16/mem16,reg16
12	0001 0010	mod reg r/m	(disp-lo),(disp-hi)	adc	reg8,reg8/mem8
13	0001 0011	mod reg r/m	(disp-lo),(disp-hi)	adc	reg16,reg16/mem16
14	0001 0100	data-8		adc	AL,immed8
15	0001 0101	data-lo	data-hi	adc	AX,immed16
16	0001 0110			push	SS
17	0001 0111			pop	SS
18	0001 1000	mod reg r/m	(disp-lo),(disp-hi)	sbb	reg8/mem8,reg8
19	0001 1001	mod reg r/m	(disp-lo),(disp-hi)	sbb	reg16/mem16,reg16
1A	0001 1010	mod reg r/m	(disp-lo),(disp-hi)	sbb	reg8,reg8/mem8
1B	0001 1011	mod reg r/m	(disp-lo),(disp-hi)	sbb	reg16,reg16/mem16
1C	0001 1100	data-8		sbb	AL,immed8
1D	0001 1101	data-lo	data-hi	sbb	AX,immed16
1E	0001 1110			push	DS
1F	0001 1111			pop	DS
20	0010 0000	mod reg r/m	(disp-lo),(disp-hi)	and	reg8/mem8,reg8
21	0010 0001	mod reg r/m	(disp-lo),(disp-hi)	and	reg16/mem16,reg16
22	0010 0010	mod reg r/m	(disp-lo),(disp-hi)	and	reg8,reg8/mem8
23	0010 0011	mod reg r/m	(disp-lo),(disp-hi)	and	reg16,reg16/mem16
24	0010 0100	data-8		and	AL,immed8
25	0010 0101	data-lo	data-hi	and	AX,immed16
26	0010 0110			ES:	(segment override prefix)
27	0010 0111			daa	
28	0010 1000	mod reg r/m	(disp-lo),(disp-hi)	sub	reg8/mem8,reg8
29	0010 1001	mod reg r/m	(disp-lo),(disp-hi)	sub	reg16/mem16,reg16
2A	0010 1010	mod reg r/m	(disp-lo),(disp-hi)	sub	reg8,reg8/mem8
2B	0010 1011	mod reg r/m	(disp-lo),(disp-hi)	sub	reg16,reg16/mem16
2C	0010 1100	data-8		sub	AL,immed8
2D	0010 1101	data-lo	data-hi	sub	AX,immed16

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
2E	0010 1110			DS:	(segment override prefix)
2F	0010 1111			das	
30	0011 0000	mod reg r/m	(disp-lo),(disp-hi)	xor	reg8/mem8,reg8
31	0011 0001	mod reg r/m	(disp-lo),(disp-hi)	xor	reg16/mem16,reg16
32	0011 0010	mod reg r/m	(disp-lo),(disp-hi)	xor	reg8,reg8/mem8
33	0011 0011	mod reg r/m	(disp-lo),(disp-hi)	xor	reg16,reg16/mem16
34	0011 0100	data-8		xor	AL,immed8
35	0011 0101	data-lo	data-hi	xor	AX,immed16
36	0011 0110			SS:	(segment override prefix)
37	0011 0111			aaa	
38	0011 1000	mod reg r/m	(disp-lo),(disp-hi)	xor	reg8/mem8,reg8
39	0011 1001	mod reg r/m	(disp-lo),(disp-hi)	xor	reg16/mem16,reg16
3A	0011 1010	mod reg r/m	(disp-lo),(disp-hi)	xor	reg8,reg8/mem8
3B	0011 1011	mod reg r/m	(disp-lo),(disp-hi)	xor	reg16,reg16/mem16
3C	0011 1100	data-8		xor	AL,immed8
3D	0011 1101	data-lo	data-hi	xor	AX,immed16
3E	0011 1110			DS:	(segment override prefix)
3F	0011 1111			aas	
40	0100 0000			inc	AX
41	0100 0001			inc	CX
42	0100 0010			inc	DX
43	0100 0011			inc	BX
44	0100 0100			inc	SP
45	0100 0101			inc	BP
46	0100 0110			inc	SI
47	0100 0111			inc	DI
48	0100 1000			dec	AX
49	0100 1001			dec	CX
4A	0100 1010			dec	DX
4B	0100 1011			dec	BX
4C	0100 1100			dec	SP
4D	0100 1101			dec	BP
4E	0100 1110			dec	SI
4F	0100 1111			dec	DI
50	0101 0000			push	AX
51	0101 0001			push	CX
52	0101 0010			push	DX

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
53	0101 0011			push	BX
54	0101 0100			push	SP
55	0101 0101			push	BP
56	0101 0110			push	SI
57	0101 0111			push	DI
58	0101 1000			pop	AX
59	0101 1001			pop	CX
5A	0101 1010			pop	DX
5B	0101 1011			pop	BX
5C	0101 1100			pop	SP
5D	0101 1101			pop	BP
5E	0101 1110			pop	SI
5F	0101 1111			pop	DI
60	0110 0000			pusha	
61	0110 0001			popa	
62	0110 0010	mod reg r/m		bound	reg16,mem16
63	0110 0011			—	
64	0110 0100			—	
65	0110 0101			—	
66	0110 0110			—	
67	0110 0111			—	
68	0110 1000	data-lo	data-hi	push	immed16
69	0110 1001	mod reg r/m	data-lo, data-hi	imul	immed16
70	0111 0000	IP-inc-8		jo	short-label
71	0111 0001	IP-inc-8		jno	short-label
72	0111 0010	IP-inc-8		jb/jnae/jc	short-label
73	0111 0011	IP-inc-8		jnb/jae/jnc	short-label
74	0111 0100	IP-inc-8		je/jz	short-label
75	0111 0101	IP-inc-8		jne/jnz	short-label
76	0111 0110	IP-inc-8		jbe/jna	short-label
77	0111 0111	IP-inc-8		jnbe/ja	short-label
78	0111 1000	IP-inc-8		js	short-label
79	0111 1001	IP-inc-8		jns	short-label
7A	0111 1010	IP-inc-8		jp/jpe	short-label
7B	0111 1011	IP-inc-8		jnp/jpo	short-label
7C	0111 1100	IP-inc-8		jl/jnge	short-label
7D	0111 1101	IP-inc-8		jnl/jge	short-label

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
7E	0111 1110	IP-inc-8		jle/jng	short-label
7F	0111 1111	IP-inc-8		jnle/jg	short-label
80	1000 0000	mod 000 r/m	(disp-lo),(disp-hi), data-8	add	reg8/mem8,immed8
		mod 001 r/m	(disp-lo),(disp-hi), data-8	or	reg8/mem8,immed8
		mod 010 r/m	(disp-lo),(disp-hi), data-8	adc	reg8/mem8,immed8
		mod 011 r/m	(disp-lo),(disp-hi), data-8	sbb	reg8/mem8,immed8
		mod 100 r/m	(disp-lo),(disp-hi), data-8	and	reg8/mem8,immed8
		mod 101 r/m	(disp-lo),(disp-hi), data-8	sub	reg8/mem8,immed8
		mod 110 r/m	(disp-lo),(disp-hi), data-8	xor	reg8/mem8,immed8
		mod 111 r/m	(disp-lo),(disp-hi), data-8	cmp	reg8/mem8,immed8
81	1000 0001	mod 000 r/m	(disp-lo),(disp-hi), data-lo,data-hi	add	reg16/mem16,immed16
		mod 001 r/m	(disp-lo),(disp-hi), data-lo,data-hi	or	reg16/mem16,immed16
		mod 010 r/m	(disp-lo),(disp-hi), data-lo,data-hi	adc	reg16/mem16,immed16
		mod 011 r/m	(disp-lo),(disp-hi), data-lo,data-hi	sbb	reg16/mem16,immed16
		mod 100 r/m	(disp-lo),(disp-hi), data-lo,data-hi	and	reg16/mem16,immed16
81	1000 0001	mod 101 r/m	(disp-lo),(disp-hi), data-lo,data-hi	sub	reg16/mem16,immed16
		mod 110 r/m	(disp-lo),(disp-hi), data-lo,data-hi	xor	reg16/mem16,immed16
		mod 111 r/m	(disp-lo),(disp-hi), data-lo,data-hi	cmp	reg16/mem16,immed16
82	1000 0010	mod 000 r/m	(disp-lo),(disp-hi), data-8	add	reg8/mem8,immed8
		mod 001 r/m		—	
		mod 010 r/m	(disp-lo),(disp-hi), data-8	adc	reg8/mem8,immed8
		mod 011 r/m	(disp-lo),(disp-hi), data-8	sbb	reg8/mem8,immed8
		mod 100 r/m		—	
		mod 101 r/m	(disp-lo),(disp-hi), data-8	sub	reg8/mem8,immed8
		mod 110 r/m		—	
		mod 111 r/m	(disp-lo),(disp-hi), data-8	cmp	reg8/mem8,immed8
83	1000 0011	mod 000 r/m	(disp-lo),(disp-hi), data-SX	add	reg16/mem16,immed8
		mod 001 r/m		—	
		mod 010 r/m	(disp-lo),(disp-hi), data-SX	adc	reg16/mem16,immed8
		mod 011 r/m	(disp-lo),(disp-hi), data-SX	sbb	reg16/mem16,immed8
		mod 100 r/m		—	
		mod 101 r/m	(disp-lo),(disp-hi), data-SX	sub	reg16/mem16,immed8
		mod 110 r/m		—	
		mod 111 r/m	(disp-lo),(disp-hi), data-SX	cmp	reg16/mem16,immed8
84	1000 0100	mod reg r/m	(disp-lo),(disp-hi)	test	reg8/mem8,reg8
85	1000 0101	mod reg r/m	(disp-lo),(disp-hi)	test	reg16/mem16,reg16
86	1000 0110	mod reg r/m	(disp-lo),(disp-hi)	xchg	reg8,reg8/mem8

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
87	1000 0111	mod reg r/m	(disp-lo),(disp-hi)	xchg	reg16,reg16/mem16
88	1000 0100	mod reg r/m	(disp-lo),(disp-hi)	mov	reg8/mem8,reg8
89	1000 1001	mod reg r/m	(disp-lo),(disp-hi)	mov	reg16/mem16,reg16
8A	1000 1010	mod reg r/m	(disp-lo),(disp-hi)	mov	reg8,reg8/mem8
8B	1000 1011	mod reg r/m	(disp-lo),(disp-hi)	mov	reg16,reg16/mem16
8C	1000 1100	mod OSR r/m	(disp-lo),(disp-hi)	mov	reg16/mem16,SEGREG
		mod 1 - r/m		—	
8D	1000 1101	mod reg r/m	(disp-lo),(disp-hi)	lea	reg16,mem16
8E	1000 1110	mod OSR r/m	(disp-lo),(disp-hi)	mov	SEGREG,reg16/mem16
		mod 1 - r/m		—	
8F	1000 1111			pop	mem16
90	1001 0000			nop	(xchg AX,AX)
91	1001 0001			xchg	AX,CX
92	1001 0010			xchg	AX,DX
93	1001 0011			xchg	AX,BX
94	1001 0100			xchg	AX,SP
95	1001 0101			xchg	AX,BP
96	1001 0110			xchg	AX,SI
97	1001 0111			xchg	AX,DI
98	1001 1000			cbw	
99	1001 1001			cwd	
9A	1001 1010	disp-lo	disp-hi,seg-lo,seg-hi	call	far-proc
9B	1001 1011			wait	
9C	1001 1100			pushf	
9D	1001 1101			popf	
9E	1001 1110			sahf	
9F	1001 1111			lahf	
A0	1010 0000	addr-lo	addr-hi	mov	AL,mem8
A1	1010 0001	addr-lo	addr-hi	mov	AX,mem16
A2	1010 0010	addr-lo	addr-hi	mov	mem8,AL
A3	1010 0011	addr-lo	addr-hi	mov	mem16,AL
A4	1010 0100			movs	dest-str8,src-str8
A5	1010 0101			movs	dest-str16,src-str16
A6	1010 0110			cmps	dest-str8,src-str8
A7	1010 0111			cmps	dest-str16,src-str16
A8	1010 1000	data-8		test	AL,immed8
A9	1010 1001	data-lo	data-hi	test	AX,immed16

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
AA	1010 1010			stos	dest-str8
AB	1010 1011			stos	dest-str16
AC	1010 1100			lods	src-str8
AD	1010 1101			lods	src-str16
AE	1010 1110			scas	dest-str8
AF	1010 1111			scas	dest-str16
B0	1011 0000	data-8		mov	AL,immed8
B1	1011 0001	data-8		mov	CL,immed8
B2	1011 0010	data-8		mov	DL,immed8
B3	1011 0011	data-8		mov	BL,immed8
B4	1011 0100	data-8		mov	AH,immed8
B5	1011 0101	data-8		mov	CH,immed8
B6	1011 0110	data-8		mov	DH,immed8
B7	1011 0111	data-8		mov	BH,immed8
B8	1011 1000	data-lo	data-hi	mov	AX,immed16
B9	1011 1001	data-lo	data-hi	mov	CX,immed16
BA	1011 1010	data-lo	data-hi	mov	DX,immed16
BB	1011 1011	data-lo	data-hi	mov	BX,immed16
BC	1011 1100	data-lo	data-hi	mov	SP,immed16
BD	1011 1101	data-lo	data-hi	mov	BP,immed16
BE	1011 1110	data-lo	data-hi	mov	SI,immed16
BF	1011 1111	data-lo	data-hi	mov	DI,immed16
C0	1100 0000	mod 000 r/m	data-8	rol	reg8/mem8, immed8
		mod 001 r/m	data-8	ror	reg8/mem8, immed8
		mod 010 r/m	data-8	rcl	reg8/mem8, immed8
		mod 011 r/m	data-8	rcr	reg8/mem8, immed8
		mod 100 r/m	data-8	shl/sal	reg8/mem8, immed8
		mod 101 r/m	data-8	shr	reg8/mem8, immed8
		mod 110 r/m		—	
		mod 111 r/m	data-8	sar	reg8/mem8, immed8
C1	1100 0001	mod 000 r/m	data-8	rol	reg16/mem16, immed8
		mod 001 r/m	data-8	ror	reg16/mem16, immed8
		mod 010 r/m	data-8	rcl	reg16/mem16, immed8
		mod 011 r/m	data-8	rcr	reg16/mem16, immed8
		mod 100 r/m	data-8	shl/sal	reg16/mem16, immed8
		mod 101 r/m	data-8	shr	reg16/mem16, immed8
		mod 110 r/m		—	

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
		mod 111 r/m	data-8	sar	reg16/mem16, immed8
C2	1100 0010	data-lo	data-hi	ret	immed16 (intrasegment)
C3	1100 0011			ret	(intrasegment)
C4	1100 0100	mod reg r/m	(disp-lo),(disp-hi)	les	reg16,mem16
C5	1100 0101	mod reg r/m	(disp-lo),(disp-hi)	lds	reg16,mem16
C6	1100 0110	mod 000 r/m	(disp-lo),(disp-hi),data-8	mov	mem8,immed8
		mod 001 r/m		—	
		mod 010 r/m		—	
		mod 011 r/m		—	
		mod 100 r/m		—	
		mod 101 r/m		—	
		mod 110 r/m		—	
C6	1100 0110	mod 111 r/m		—	
C7	1100 0111	mod 000 r/m	(disp-lo),(disp-hi),data-lo,data-hi	mov	mem16,immed16
		mod 001 r/m		—	
		mod 010 r/m		—	
		mod 011 r/m		—	
		mod 100 r/m		—	
		mod 101 r/m		—	
		mod 110 r/m		—	
		mod 111 r/m		—	
C8	1100 1000	data-lo	data-hi, level	enter	immed16, immed8
C9	1100 1001			leave	
CA	1100 1010	data-lo	data-hi	ret	immed16 (intersegment)
CB	1100 1011			ret	(intersegment)
CC	1100 1100			int	3
CD	1100 1101	data-8		int	immed8
CE	1100 1110			into	
CF	1100 1111			iret	
D0	1101 0000	mod 000 r/m	(disp-lo),(disp-hi)	rol	reg8/mem8,1
		mod 001 r/m	(disp-lo),(disp-hi)	ror	reg8/mem8,1
		mod 010 r/m	(disp-lo),(disp-hi)	rcl	reg8/mem8,1
		mod 011 r/m	(disp-lo),(disp-hi)	rcr	reg8/mem8,1
		mod 100 r/m	(disp-lo),(disp-hi)	sal/shl	reg8/mem8,1
		mod 101 r/m	(disp-lo),(disp-hi)	shr	reg8/mem8,1
		mod 110 r/m		—	
		mod 111 r/m	(disp-lo),(disp-hi)	sar	reg8/mem8,1

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format
Hex	Binary			
D1	1101 0001	mod 000 r/m	(disp-lo),(disp-hi)	rol reg16/mem16,1
		mod 001 r/m	(disp-lo),(disp-hi)	ror reg16/mem16,1
D1	1101 0001	mod 010 r/m	(disp-lo),(disp-hi)	rcl reg16/mem16,1
		mod 011 r/m	(disp-lo),(disp-hi)	rcr reg16/mem16,1
		mod 100 r/m	(disp-lo),(disp-hi)	sal/shl reg16/mem16,1
		mod 101 r/m	(disp-lo),(disp-hi)	shr reg16/mem16,1
		mod 110 r/m		—
		mod 111 r/m	(disp-lo),(disp-hi)	sar reg16/mem16,1
D2	1101 0010	mod 000 r/m	(disp-lo),(disp-hi)	rol reg8/mem8,CL
		mod 001 r/m	(disp-lo),(disp-hi)	ror reg8/mem8,CL
		mod 010 r/m	(disp-lo),(disp-hi)	rcl reg8/mem8,CL
		mod 011 r/m	(disp-lo),(disp-hi)	rcr reg8/mem8,CL
		mod 100 r/m	(disp-lo),(disp-hi)	sal/shl reg8/mem8,CL
		mod 101 r/m	(disp-lo),(disp-hi)	shr reg8/mem8,CL
		mod 110 r/m		—
		mod 111 r/m	(disp-lo),(disp-hi)	sar reg8/mem8,CL
D3	1101 0011	mod 000 r/m	(disp-lo),(disp-hi)	rol reg16/mem16,CL
		mod 001 r/m	(disp-lo),(disp-hi)	ror reg16/mem16,CL
		mod 010 r/m	(disp-lo),(disp-hi)	rcl reg16/mem16,CL
		mod 011 r/m	(disp-lo),(disp-hi)	rcr reg16/mem16,CL
		mod 100 r/m	(disp-lo),(disp-hi)	sal/shl reg16/mem16,CL
		mod 101 r/m	(disp-lo),(disp-hi)	shr reg16/mem16,CL
		mod 110 r/m		—
		mod 111 r/m	(disp-lo),(disp-hi)	sar reg16/mem16,CL
D4	1101 0100	0000 1010		aam
D5	1101 0101	0000 1010		aad
D6	1101 0110			—
D7	1101 0111			xlat source-table
D8	1101 1000	mod 000 r/m	(disp-lo),(disp-hi)	esc opcode,source
D9	1101 1001	mod 001 r/m	(disp-lo),(disp-hi)	esc opcode,source
DA	1101 1010	mod 010 r/m	(disp-lo),(disp-hi)	esc opcode,source
DB	1101 1011	mod 011 r/m	(disp-lo),(disp-hi)	esc opcode,source
DC	1101 1100	mod 100 r/m	(disp-lo),(disp-hi)	esc opcode,source
DD	1101 1101	mod 101 r/m	(disp-lo),(disp-hi)	esc opcode,source
DE	1101 1110	mod 110 r/m	(disp-lo),(disp-hi)	esc opcode,source
DF	1101 1111	mod 111 r/m	(disp-lo),(disp-hi)	esc opcode,source
E0	1110 0000	IP-inc-8		loopne/loopnz short-label

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
E1	1110 0001	IP-inc-8		loope/loopz	short-label
E2	1110 0010	IP-inc-8		loop	short-label
E3	1110 0011	IP-inc-8		jcxz	short-label
E4	1110 0100	data-8		in	AL,immed8
E5	1110 0101	data-8		in	AX,immed8
E6	1110 0110	data-8		out	AL,immed8
E7	1110 0111	data-8		out	AX,immed8
E8	1110 1000	IP-inc-lo	IP-inc-hi	call	near-proc
E9	1110 1001	IP-inc-lo	IP-inc-hi	jmp	near-label
EA	1110 1010	IP-lo	IP-hi,CS-lo,CS-hi	jmp	far-label
EB	1110 1011	IP-inc-8		jmp	short-label
EC	1110 1100			in	AL,DX
ED	1110 1101			in	AX,DX
EE	1110 1110			out	AL,DX
EF	1110 1111			out	AX,DX
F0	1111 0000			lock	(prefix)
F1	1111 0001			—	
F2	1111 0010			repne/repnz	
F3	1111 0011			rep/repe/repz	
F4	1111 0100			hlt	
F5	1111 0101			cmc	
F6	1111 0110	mod 000 r/m	(disp-lo),(disp-hi),data-8	test	reg8/mem8,immed8
		mod 001 r/m		—	
		mod 010 r/m	(disp-lo),(disp-hi)	not	reg8/mem8
		mod 011 r/m	(disp-lo),(disp-hi)	neg	reg8/mem8
		mod 100 r/m	(disp-lo),(disp-hi)	mul	reg8/mem8
		mod 101 r/m	(disp-lo),(disp-hi)	imul	reg8/mem8
		mod 110 r/m	(disp-lo),(disp-hi)	div	reg8/mem8
		mod 111 r/m	(disp-lo),(disp-hi)	idiv	reg8/mem8
F7	1111 0111	mod 000 r/m	(disp-lo),(disp-hi),data-lo,data-hi	test	reg16/mem16,immed16
		mod 001 r/m		—	
		mod 010 r/m	(disp-lo),(disp-hi)	not	reg16/mem16
		mod 011 r/m	(disp-lo),(disp-hi)	neg	reg16/mem16
		mod 100 r/m	(disp-lo),(disp-hi)	mul	reg16/mem16
		mod 101 r/m	(disp-lo),(disp-hi)	imul	reg16/mem16
		mod 110 r/m	(disp-lo),(disp-hi)	div	reg16/mem16
		mod 111 r/m	(disp-lo),(disp-hi)	idiv	reg16/mem16

Table D-3. Machine Instruction Decoding Guide (Continued)

Byte 1		Byte 2	Bytes 3–6	ASM-86 Instruction Format	
Hex	Binary				
F8	1111 1000			clc	
F9	1111 1001			stc	
FA	1111 1010			cli	
FB	1111 1011			sti	
FC	1111 1100			cld	
FD	1111 1101			std	
FE	1111 1110	mod 000 r/m	(disp-lo),(disp-hi)	inc	mem16
		mod 001 r/m	(disp-lo),(disp-hi)	dec	mem16
		mod 010 r/m		—	
FE	1111 1110	mod 011 r/m		—	
		mod 100 r/m		—	
		mod 101 r/m		—	
		mod 110 r/m		—	
		mod 111 r/m		—	
FF	1111 1111	mod 000 r/m	(disp-lo),(disp-hi)	inc	mem16
		mod 001 r/m	(disp-lo),(disp-hi)	dec	mem16
		mod 010 r/m	(disp-lo),(disp-hi)	call	reg16/mem16 (intrasegment)
		mod 011 r/m	(disp-lo),(disp-hi)	call	mem16 (intersegment)
		mod 100 r/m	(disp-lo),(disp-hi)	jmp	reg16/mem16 (intrasegment)
		mod 101 r/m	(disp-lo),(disp-hi)	jmp	mem16 (intersegment)
		mod 110 r/m	(disp-lo),(disp-hi)	push	mem16
		mod 111 r/m		—	

Table D-4. Mnemonic Encoding Matrix (Left Half)

	x0	x1	x2	x3	x4	x5	x6	x7
0x	ADD b,f,r/m	ADD w,f,r/m	ADD b,t,r/m	ADD w,t,r/m	ADD b,ia	ADD w,ia	PUSH ES	POP ES
1x	ADC b,f,r/m	ADC w,f,r/m	ADC b,t,r/m	ADC w,t,r/m	ADC b,i	ADC w,i	PUSH SS	POP SS
2x	AND b,f,r/m	AND w,f,r/m	AND b,t,r/m	AND w,t,r/m	AND b,i	AND w,i	SEG =ES	DAA
3x	XOR b,f,r/m	XOR w,f,r/m	XOR b,t,r/m	XOR w,t,r/m	XOR b,i	XOR w,i	SEG =SS	AAA
4x	INC AX	INC CX	INC DX	INC BX	INC SP	INC BP	INC SI	INC DI
5x	PUSH AX	PUSH CX	PUSH DX	PUSH BX	PUSH SP	PUSH BP	PUSH SI	PUSH DI
6x	PUSHA	POPA	BOUND w,f,r/m	▨	▨	▨	▨	▨
7x	JO	JNO	JB/ JNAE/ JC	JNB/ JAE/ JNC	JE/ JZ	JNE/ JNZ	JBE/ JNA	JNBE/ JA
8x	Immed b,r/m	Immed w,r/m	Immed b,r/m	Immed is,r/m	TEST b,r/m	TEST w,r/m	XCHG b,r/m	XCHG w,r/m
9x	NOP (XCHG) AX	XCHG CX	XCHG DX	XCHG BX	XCHG SP	XCHG BP	XCHG SI	XCHG DI
Ax	MOV m→AL	MOV m→AX	MOV AL→m	MOV AX→m	MOVS	MOVS	CMPS	CMPS
Bx	MOV i→AL	MOV i→CL	MOV i→DL	MOV i→BL	MOV i→AH	MOV i→CH	MOV i→DH	MOV i→BH
Cx	Shift b,i	Shift w,i	RET (i+SP)	RET	LES	LDS	MOV b,i,r/m	MOV w,i,r/m
Dx	Shift b	Shift w	Shift b,v	Shift w,v	AAM	AAD	▨	XLAT
Ex	LOOPNZ/ LOOPNE	LOOPZ/ LOOPE	LOOP	JCXZ	IN	IN	OUT	OUT
Fx	LOCK	▨	REP	REP z	HLT	CMC	Grp1 b,r/m	Grp1 w,r/m

NOTE: Table D-5 defines abbreviations used in this matrix. Shading indicates reserved opcodes.

Table D-4. Mnemonic Encoding Matrix (Right Half)

x8	x9	xA	xB	xC	xD	xE	xF	
OR b,f,r/m	OR w,f,r/m	OR b,t,r/m	OR w,t,r/m	OR b,i	OR w,i	PUSH CS	▨	**0x**
SBB b,f,r/m	SBB w,f,r/m	SBB b,t,r/m	SBB w,t,r/m	SBB b,i	SBB w,i	PUSH DS	POP DS	**1x**
SUB b,f,r/m	SUB w,f,r/m	SUB b,t,r/m	SUB w,t,r/m	SUB b,i	SUB w,i	SEG =CS	DAS	**2x**
CMP b,f,r/m	CMP w,f,r/m	CMP b,t,r/m	CMP w,t,r/m	CMP b,i	CMP w,i	SEG =DS	AAS	**3x**
DEC AX	DEC CX	DEC DX	DEC BX	DEC SP	DEC BP	DEC SI	DEC DI	**4x**
POP AX	POP CX	POP DX	POP BX	POP SP	POP BP	POP SI	POP DI	**5x**
PUSH w,i	IMUL w,i	PUSH b,i	IMUL w,i	INS b	INS w	OUTS b	OUTS w	**6x**
JS	JNS	JP/ JPE	JNP/ JPO	JL/ JNGE	JNL/ JGE	JLE/ JNG	JNLE/ JG	**7x**
MOV b,f,r/m	MOV w,f,r/m	MOV b,t,r/m	MOV w,t,r/m	MOV sr,f,r/m	LEA	MOV sr,t,r/m	POP r/m	**8x**
CBW	CWD	CALL L,D	WAIT	PUSHF	POPF	SAHF	LAHF	**9x**
TEST b,ia	TEST w,ia	STOS	STOS	LODS	LODS	SCAS	SCAS	**Ax**
MOV i→AX	MOV i→CX	MOV i→DX	MOV i→BX	MOV i→SP	MOV i→BP	MOV i→SI	MOV i→DI	**Bx**
ENTER	LEAVE	RET l(i+SP)	RET l	INT type 3	INT (any)	INTO	IRET	**Cx**
ESC 0	ESC 1	ESC 2	ESC 3	ESC 4	ESC 5	ESC 6	ESC 7	**Dx**
CALL	JMP	JMP	JMP	IN	IN	OUT	OUT	**Ex**
CLC	STC	CLI	STI	CLS	STD	Grp2 b,r/m	Grp2 w,r/m	**Fx**

NOTE: Table D-5 defines abbreviations used in this matrix. Shading indicates reserved opcodes.

Table D-5. Abbreviations for Mnemonic Encoding Matrix

Abbr	Definition	Abbr	Definition	Abbr	Definition	Abbr	Definition
b	byte operation	ia	immediate to accumulator	m	memory	t	to CPU register
d	direct	id	indirect	r/m	EA is second byte	v	variable
f	from CPU register	is	immediate byte, sign extended	si	short intrasegment	w	word operation
i	immediate	l	long (intersegment)	sr	segment register	z	zero

Byte 2	Immed	Shift	Grp1	Grp2
mod 000 r/m	ADD	ROL	TEST	INC
mod 001 r/m	OR	ROR	—	DEC
mod 010 r/m	ADC	RCL	NOT	CALL id
mod 011 r/m	SBB	RCR	NEG	CALL l, id
mod 100 r/m	AND	SHL/SAL	MUL	JMP id
mod 101 r/m	SUB	SHR	IMUL	JMP i, id
mod 110 r/m	XOR	—	DIV	PUSH
mod 111 r/m	CMP	SAR	IDIV	—
mod and *r/m* determine the Effective Address (EA) calculation. See Table D-1 for definitions.				