Table 6-20. **6x86 CPU Instruction Set Clock Count Summary**

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
AAA *ASCII Adjust AL after Add*	37	u - - - u u x u x	7	7		
AAD *ASCII Adjust AX before Divide*	D5 0A	u - - - x x u x u	7	7		
AAM ASCII *Adjust AX after Multiply*	D4 0A	u - - - x x u x u	13-21	13-21		
AAS ASCII *Adjust AL after Subtract*	3F	u - - - u u x u x	7	7		
ADC *Add with Carry*		x - - - x x x x x			b	h
Register to Register	1 [00dw] [11 reg r/m]		1	1		
Register to Memory	1 [000w] [mod reg r/m]		1	1		
Memory to Register	1 [001w] [mod reg r/m]		1	1		
Immediate to Register/Memory	8 [00sw] [mod 010 r/m]###		1	1		
Immediate to Accumulator	1 [010w] ###		1	1		
ADD *Integer Add*		x - - - x x x x x			b	h
Register to Register	0 [00dw] [11 reg r/m]		1	1		
Register to Memory	0 [000w] [mod reg r/m]		1	1		
Memory to Register	0 [001w] [mod reg r/m]		1	1		
Immediate to Register/Memory	8 [00sw] [mod 000 r/m]###		1	1		
Immediate to Accumulator	0 [010w] ###		1	1		
AND *Boolean AND*		0 - - - x x u x 0			b	h
Register to Register	2 [00dw] [11 reg r/m]		1	1		
Register to Memory	2 [000w] [mod reg r/m]		1	1		
Memory to Register	2 [001w] [mod reg r/m]		1	1		
Immediate to Register/Memory	8 [00sw] [mod 100 r/m]###		1	1		
Immediate to Accumulator	2 [010w] ###		1	1		
ARPL *Adjust Requested Privilege Level*		- - - - - x - - -		9	a	h
From Register/Memory	63 [mod reg r/m]					
BOUND *Check Array Boundaries*	62 [mod reg r/m]	- - - - - - - - -			b, e	g,h,j,k,r
If Out of Range (Int 5)			20	20+INT		
If In Range			11	11		
BSF *Scan Bit Forward*		- - - - - x - - -	3	3	b	h
Register, Register/Memory	0F BC [mod reg r/m]					
BSR *Scan Bit Reverse*		- - - - - x - - -	3	3	b	h
Register, Register/Memory	0F BD [mod reg r/m]					
BSWAP *Byte Swap*	0F C[1 reg]	- - - - - - - - -	4	4		
BT *Test Bit*		- - - - - - - - x			b	h
Register/Memory, Immediate	0F BA [mod 100 r/m]#		2	2		
Register/Memory, Register	0F A3 [mod reg r/m]		5/6	5/6		
BTC *Test Bit and Complement*		- - - - - - - - x			b	h
Register/Memory, Immediate	0F BA [mod 111 r/m]#		3	3		
Register/Memory, Register	0F BB [mod reg r/m]		5/6	5/6		

\# = immediate 8-bit data
= immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
BTR *Test Bit and Reset*		- - - - - - - - x			b	h
Register/Memory, Immediate	0F BA [mod 110 r/m]#		3	3		
Register/Memory, Register	0F B3 [mod reg r/m]		5/6	5/6		
BTS *Test Bit and Set*		- - - - - - - - x			b	h
Register/Memory	0F BA [mod 101 r/m]		3	3		
Register (short form)	0F AB [mod reg r/m]		5/6	5/6		
CALL *Subroutine Call*		- - - - - - - - - -			b	h,j,k,r
Direct Within Segment	E8 +++		1	1		
Register/Memory Indirect Within Segment	FF [mod 010 r/m]		1/3	1/3		
Direct Intersegment	9A [unsigned full offset,		3	4		
Call Gate to Same Privilege	selector]			15		
Call Gate to Different Privilege No Parameters				26		
Call Gate to Different Privilege m Par's				35+2m		
16-bit Task to 16-bit TSS				110		
16-bit Task to 32-bit TSS				118		
16-bit Task to V86 Task				96		
32-bit Task to 16-bit TSS				112		
32-bit Task to 32-bit TSS				120		
32-bit Task to V86 Task				98		
Indirect Intersegment	FF [mod 011 r/m]		5	8		
Call Gate to Same Privilege				20		
Call Gate to Different Privilege No Parameters				31		
Call Gate to Different Privilege Level m Par's				40+2m		
16-bit Task to 16-bit TSS				114		
16-bit Task to 32-bit TSS				122		
16-bit Task to V86 Task				100		
32-bit Task to 16-bit TSS				116		
32-bit Task to 32-bit TSS				124		
32-bit Task to V86 Task				102		
CBW *Convert Byte to Word*	98	- - - - - - - - - -	3	3		
CDQ *Convert Doubleword to Quadword*	99	- - - - - - - - - -	2	2		
CLC *Clear Carry Flag*	F8	- - - - - - - - - 0	1	1		
CLD *Clear Direction Flag*	FC	- 0 - - - - - - - -	7	7		
CLI *Clear Interrupt Flag*	FA	- - 0 - - - - - - -	7	7		m
CLTS *Clear Task Switched Flag*	0F 06	- - - - - - - - - -	10	10	c	l
CMC *Complement the Carry Flag*	F5	- - - - - - - - - x	2	2		

\# = immediate 8-bit data
= immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
CMP *Compare Integers*		x - - - x x x x x			b	h
Register to Register	3 [10dw] [11 reg r/m]		1	1		
Register to Memory	3 [101w] [mod reg r/m]		1	1		
Memory to Register	3 [100w] [mod reg r/m]		1	1		
Immediate to Register/Memory	8 [00sw] [mod 111 r/m] ###		1	1		
Immediate to Accumulator	3 [110w] ###		1	1		
CMPS *Compare String*	A [011w]	x - - - x x x x x	5	5	b	h
CMPXCHG *Compare and Exchange*		x - - - x x x x x				
Register1, Register2	0F B [000w] [11 reg2 reg1]		11	11		
Memory, Register	0F B [000w] [mod reg r/m]		11	11		
CPUID *CPU Identification*	0F A2	- - - - - - - - -	12	12		
CWD *Convert Word to Doubleword*	99	- - - - - - - - -	2	2		
CWDE *Convert Word to Doubleword Extended*	98	- - - - - - - - -	2	2		
DAA *Decimal Adjust AL after Add*	27	- - - - x x x x x	9	9		
DAS *Decimal Adjust AL after Subtract*	2F	- - - - x x x x x	9	9		
DEC *Decrement by 1*		x - - - x x x x -			b	h
Register/Memory	F [111w] [mod 001 r/m]		1	1		
Register (short form)	4 [1 reg]		1	1		
DIV *Unsigned Divide*	F [011w] [mod 110 r/m]	- - - - x x u u -			b,e	e,h
Accumulator by Register/Memory						
Divisor: Byte			13-17	13-17		
Word			13-25	13-25		
Doubleword			13-41	13-41		
ENTER *Enter New Stack Frame*	C8 ##,#	- - - - - - - - -			b	h
Level = 0			10	10		
Level = 1			13	13		
Level (L) > 1			10+L*3	10+L*3		
HLT Halt	F4	- - - - - - - - -	5	5		l
IDIV *Integer (Signed) Divide*		- - - - x x u u -			b,e	e,h
Accumulator by Register/Memory	F [011w] [mod 111 r/m]					
Divisor: Byte			16-20	16-20		
Word			16-28	16-28		
Doubleword			17-45	17-45		

\# = immediate 8-bit data
= immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
IMUL *Integer (Signed) Multiply*		x - - - x x u u x			b	h
Accumulator by Register/Memory	F [011w] [mod 101 r/m]					
Multiplier: Byte			4	4		
Word			4	4		
Doubleword			10	10		
Register with Register/Memory	0F AF [mod reg r/m]					
Multiplier: Word			4	4		
Doubleword			10	10		
Register/Memory with Immediate to Register2						
Multiplier: Word	6 [10s1] [mod reg r/m] ###		5	5		
Doubleword			11	11		
IN *Input from I/O Port*		- - - - - - - - - -				m
Fixed Port	E [010w] [#]		14	14/28		
Variable Port	E [110w]		14	14/28		
INC *Increment by 1*		x - - - x x x x -			b	h
Register/Memory	F [111w] [mod 000 r/m]		1	1		
Register (short form)	4 [0 reg]		1	1		
INS *Input String from I/O Port*	6 [110w]	- - - - - - - - - -	14	14/28	b	h,m
INT *Software Interrupt*		- - x 0 - - - - - -			b,e	g,j,k,r
INT i	CD #		9			
Protected Mode:						
Interrupt or Trap to Same Privilege				21		
Interrupt or Trap to Different Privilege				32		
16-bit Task to 16-bit TSS by Task Gate				114		
16-bit Task to 32-bit TSS by Task Gate				122		
16-bit Task to V86 by Task Gate				100		
16-bit Task to 16-bit TSS by Task Gate				116		
32-bit Task to 32-bit TSS by Task Gate				124		
32-bit Task to V86 by Task Gate				102		
V86 to 16-bit TSS by Task Gate				124		
V86 to 32-bit TSS by Task Gate				102		
V86 to Privilege 0 by Trap Gate/Int Gate				46		

Continued on the next page...

\# = immediate 8-bit data
\## = immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
INT *Software Interrupt* (Continued) INT 3 INTO If OF==0 If OF==1 (INT 4)	 CC CE 	- - x 0 - - - - -	 INT 6	 INT 6 15+INT	b,e	g,j,k,r
INVD *Invalidate Cache*	0F 08	- - - - - - - - -	12	12	t	t
INVLPG *Invalidate TLB Entry*	0F 01 [mod 111 r/m]	- - - - - - - - -	13	13		
IRET *Interrupt Return* Real Mode Protected Mode: Within Task to Same Privilege Within Task to Different Privilege 16-bit Task to 16-bit Task 16-bit Task to 32-bit TSS 16-bit Task to V86 Task 32-bit Task to 16-bit TSS 32-bit Task to 32-bit TSS 32-bit Task to V86 Task	CF	x x x x x x x x x	 7	 10 26 117 125 103 119 127 105		g,h,j,k,r
JB/JNAE/JC *Jump on Below/Not Above or Equal/ Carry* 8-bit Displacement Full Displacement	 72 + 0F 82 +++	- - - - - - - - -	 1 1	 1 1		r
JBE/JNA *Jump on Below or Equal/Not Above* 8-bit Displacement Full Displacement	 76 + 0F 86 +++	- - - - - - - - -	 1 1	 1 1		r
JCXZ/JECXZ *Jump on CX/ECX Zero*	E3 +	- - - - - - - - -	1	1		r
JE/JZ *Jump on Equal/Zero* 8-bit Displacement Full Displacement	 74 + 0F 84 +++	- - - - - - - - -	 1 1	 1 1		r
JL/JNGE *Jump on Less/Not Greater or Equal* 8-bit Displacement Full Displacement	 7C + 0F 8C +++	- - - - - - - - -	 1 1	 1 1		r
JLE/JNG *Jump on Less or Equal/Not Greater* 8-bit Displacement Full Displacement	 7E + 0F 8E +++	- - - - - - - - -	 1 1	 1 1		r

\# = immediate 8-bit data
\## = immediate 16-bit data
\### = full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
JMP *Unconditional Jump*		- - - - - - - - -			b	h,j,k,r
8-bit Displacement	EB +		1	1		
Full Displacement	E9 +++		1	1		
Register/Memory Indirect Within Segment	FF [mod 100 r/m]		1/3	1/3		
Direct Intersegment	EA [unsigned full offset, selector]		1	4		
Call Gate Same Privilege Level				14		
16-bit Task to 16-bit TSS				110		
16-bit Task to 32-bit TSS				118		
16-bit Task to V86 Task				96		
32-bit Task to 16-bit TSS				112		
32-bit Task to 32-bit TSS				120		
32-bit Task to V86 Task				98		
Indirect Intersegment	FF [mod 101 r/m]		5	7		
Call Gate Same Privilege Level				17		
16-bit Task to 16-bit TSS				113		
16-bit Task to 32-bit TSS				121		
16-bit Task to V86 Task				99		
32-bit Task to 16-bit TSS				115		
32-bit Task to 32-bit TSS				123		
32-bit Task to V86 Task				101		
JNB/JAE/JNC *Jump on Not Below/Above or Equal/Not Carry*		- - - - - - - - -				r
8-bit Displacement	73 +		1	1		
Full Displacement	0F 83 +++		1	1		
JNBE/JA *Jump on Not Below or Equal/Above*		- - - - - - - - -				r
8-bit Displacement	77 +		1	1		
Full Displacement	0F 87 +++		1	1		
JNE/JNZ *Jump on Not Equal/Not Zero*		- - - - - - - - -				r
8-bit Displacement	75 +		1	1		
Full Displacement	0F 85 +++		1	1		
JNL/JGE *Jump on Not Less/Greater or Equal*		- - - - - - - - -				r
8-bit Displacement	7D +		1	1		
Full Displacement	0F 8D +++		1	1		
JNLE/JG *Jump on Not Less or Equal/Greater*		- - - - - - - - -				r
8-bit Displacement	7F +		1	1		
Full Displacement	0F 8F +++		1	1		

\# = immediate 8-bit data
\## = immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
JNO *Jump on Not Overflow* 8-bit Displacement Full Displacement	 71 + 0F 81 +++	- - - - - - - - -	 1 1	 1 1		r
JNP/JPO *Jump on Not Parity/Parity Odd* 8-bit Displacement Full Displacement	 7B + 0F 8B +++	- - - - - - - - -	 1 1	 1 1		r
JNS *Jump on Not Sign* 8-bit Displacement Full Displacement	 79 + 0F 89 +++	- - - - - - - - -	 1 1	 1 1		r
JO *Jump on Overflow* 8-bit Displacement Full Displacement	 70 + 0F 80 +++	- - - - - - - - -	 1 1	 1 1		r
JP/JPE *Jump on Parity/Parity Even* 8-bit Displacement Full Displacement	 7A + 0F 8A +++	- - - - - - - - -	 1 1	 1 1		r
JS *Jump on Sign* 8-bit Displacement Full Displacement	 78 + 0F 88 +++	- - - - - - - - -	 1 1	 1 1		r
LAHF *Load AH with Flags*	9F	- - - - - - - - -	2	2		
LAR *Load Access Rights* From Register/Memory	 0F 02 [mod reg r/m]	- - - - - - x - - -		 8	a	g,h,j,p
LDS *Load Pointer to DS*	C5 [mod reg r/m]	- - - - - - - - -	2	4	b	h,i,j
LEA *Load Effective Address* No Index Register With Index Register	8D [mod reg r/m]	- - - - - - - - -	 1 1	 1 1		
LEAVE *Leave Current Stack Frame*	C9	- - - - - - - - -	4	4	b	h
LES *Load Pointer to ES*	C4 [mod reg r/m]	- - - - - - - - -	2	4	b	h,i,j
LFS *Load Pointer to FS*	0F B4 [mod reg r/m]	- - - - - - - - -	2	4	b	h,i,j
LGDT *Load GDT Register*	0F 01 [mod 010 r/m]	- - - - - - - - -	8	8	b,c	h,l
LGS *Load Pointer to GS*	0F B5 [mod reg r/m]	- - - - - - - - -	2	4	b	h,i,j
LIDT *Load IDT Register*	0F 01 [mod 011 r/m]	- - - - - - - - -	8	8	b,c	h,l
LLDT *Load LDT Register* From Register/Memory	 0F 00 [mod 010 r/m]	- - - - - - - - -	 5	 5	a	g,h,j,l
LMSW *Load Machine Status Word* From Register/Memory	 0F 01 [mod 110 r/m]	- - - - - - - - -	 13	 13	b,c	h,l
LODS *Load String*	A [110 w]	- - - - - - - - -	3	3	b	h
LOOP *Offset Loop/No Loop*	E2 +	- - - - - - - - -	1	1		r

\# = immediate 8-bit data
\## = immediate 16-bit data
\### = full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
LOOPNZ/LOOPNE *Offset*	E0 +	- - - - - - - - -	1	1		r
LOOPZ/LOOPE *Offset*	E1 +	- - - - - - - - -	1	1		r
LSL *Load Segment Limit*		- - - - - x - - -			a	g,h,j,p
From Register/Memory	0F 03 [mod reg r/m]			8		
LSS *Load Pointer to SS*	0F B2 [mod reg r/m]	- - - - - - - - -	2	4	a	h,i,j
LTR *Load Task Register*		- - - - - - - - -			a	g,h,j,l
From Register/Memory	0F 00 [mod 011 r/m]			7		
MOV *Move Data*		- - - - - - - - -			b	h,i,j
Register to Register	8 [10dw] [11 reg r/m]		1	1		
Register to Memory	8 [100w] [mod reg r/m]		1	1		
Register/Memory to Register	8 [101w] [mod reg r/m]		1	1		
Immediate to Register/Memory	C [011w] [mod 000 r/m] ###		1	1		
Immediate to Register (short form)	B [w reg] ###		1	1		
Memory to Accumulator (short form)	A [000w] +++		1	1		
Accumulator to Memory (short form)	A [001w] +++		1	1		
Register/Memory to Segment Register	8E [mod sreg3 r/m]		1	1/3		
Segment Register to Register/Memory	8C [mod sreg3 r/m]		1	1		
MOV *Move to/from Control/Debug/Test Regs*		- - - - - - - - -				l
Register to CR0/CR2/CR3	0F 22 [11 eee reg]		20/5/5	20/5/5		
CR0/CR2/CR3 to Register	0F 20 [11 eee reg]		6	6		
Register to DR0-DR3	0F 23 [11 eee reg]		16	16		
DR0-DR3 to Register	0F 21 [11 eee reg]		14	14		
Register to DR6-DR7	0F 23 [11 eee reg]		16	16		
DR6-DR7 to Register	0F 21 [11 eee reg]		14	14		
Register to TR3-5	0F 26 [11 eee reg]		10	10		
TR3-5 to Register	0F 24 [11 eee reg]		5	5		
Register to TR6-TR7	0F 26 [11 eee reg]		10	10		
TR6-TR7 to Register	0F 24 [11 eee reg]		6	6		
MOVS *Move String*	A [010w]	- - - - - - - - -	4	4	b	h
MOVSX *Move with Sign Extension*		- - - - - - - - -			b	h
Register from Register/Memory	0F B[111w] [mod reg r/m]		1	1		
MOVZX *Move with Zero Extension*		- - - - - - - - -			b	h
Register from Register/Memory	0F B[011w] [mod reg r/m]		1	1		

\# = immediate 8-bit data
= immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
MUL *Unsigned Multiply* Accumulator with Register/Memory Multiplier: Byte Word Doubleword	F [011w] [mod 100 r/m]	x - - - x x u u x	 4 4 10	 4 4 10	b	h
NEG *Negate Integer*	F [011w] [mod 011 r/m]	x - - - x x x x x	1	1	b	h
NOP *No Operation*	90	- - - - - - - - -	1	1		
NOT *Boolean Complement*	F [011w] [mod 010 r/m]	- - - - - - - - -	1	1	b	h
OIO *Official Invalid OpCode*	0F FF	- - x 0 - - - - -	1	8 - 125		
OR *Boolean OR* Register to Register Register to Memory Memory to Register Immediate to Register/Memory Immediate to Accumulator	 0 [10dw] [11 reg r/m] 0 [100w] [mod reg r/m] 0 [101w] [mod reg r/m] 8 [00sw] [mod 001 r/m] ### 0 [110w] ###	0 - - - x x u x 0	 1 1 1 1 1	 1 1 1 1 1	b	h
OUT *Output to Port* Fixed Port Variable Port	 E [011w] # E [111w]	- - - - - - - - -	 14 14	 14/28 14/28		m
OUTS *Output String*	6 [111w]	- - - - - - - - -	14	14/28	b	h,m
POP *Pop Value off Stack* Register/Memory Register (short form) Segment Register (ES, SS, DS) Segment Register (FS, GS)	 8F [mod 000 r/m] 5 [1 reg] [000 sreg2 111] 0F [10 sreg3 001]	- - - - - - - - -	 1 1 1 1	 1 1 3 3	b	h,i,j
POPA *Pop All General Registers*	61	- - - - - - - - -	6	6	b	h
POPF *Pop Stack into FLAGS*	9D	x x x x x x x x x	9	9	b	h,n
PREFIX BYTES Assert Hardware LOCK Prefix Address Size Prefix Operand Size Prefix Segment Override Prefix CS DS ES FS GS SS	 F0 67 66 2E 3E 26 64 65 36	- - - - - - - - -				m

\# = immediate 8-bit data
\## = immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
PUSH *Push Value onto Stack*		- - - - - - - - -			b	h
Register/Memory	FF [mod 110 r/m]		1	1		
Register (short form)	5 [0 reg]		1	1		
Segment Register (ES, CS, SS, DS)	[000 sreg2 110]		1	1		
Segment Register (FS, GS)	0F [10 sreg3 000]		1	1		
Immediate	6 [10s0] ###		1	1		
PUSHA *Push All General Registers*	60	- - - - - - - - -	6	6	b	h
PUSHF *Push FLAGS Register*	9C	- - - - - - - - -	2	2	b	h
RCL *Rotate Through Carry Left*					b	h
Register/Memory by 1	D [000w] [mod 010 r/m]	x - - - - - - - x	3	3		
Register/Memory by CL	D [001w] [mod 010 r/m]	u - - - - - - - x	8	8		
Register/Memory by Immediate	C [000w] [mod 010 r/m] #	u - - - - - - - x	8	8		
RCR *Rotate Through Carry Right*					b	h
Register/Memory by 1	D [000w] [mod 011 r/m]	x - - - - - - - x	4	4		
Register/Memory by CL	D [001w] [mod 011 r/m]	u - - - - - - - x	9	9		
Register/Memory by Immediate	C [000w] [mod 011 r/m] #	u - - - - - - - x	9	9		
REP INS *Input String*	F3 6[110w]	- - - - - - - - -	12+5n	12+5n\ 28+5n	b	h,m
REP LODS *Load String*	F3 A[110w]	- - - - - - - - -	10+n	10+n	b	h
REP MOVS *Move String*	F3 A[010w]	- - - - - - - - -	9+n	9+n	b	h
REP OUTS *Output String*	F3 6[111w]	- - - - - - - - -	12+5n	12+5n\ 28+5n	b	h,m
REP STOS *Store String*	F3 A[101w]	- - - - - - - - -	10+n	10+n	b	h
REPE CMPS *Compare String (Find non-match)*	F3 A[011w]	x - - - x x x x x	10+2n	10+2n	b	h
REPE SCAS *Scan String (Find non-AL/AX/EAX)*	F3 A[111w]	x - - - x x x x x	10+2n	10+2n	b	h
REPNE CMPS *Compare String (Find match)*	F2 A[011w]	x - - - x x x x x	10+2n	10+2n	b	h
REPNE SCAS *Scan String (Find AL/AX/EAX)*	F2 A[111w]	x - - - x x x x x	10+2n	10+2n	b	h
RET *Return from Subroutine*		- - - - - - - - -			b	g,h,j,k,r
Within Segment	C3		3	3		
Within Segment Adding Immediate to SP	C2 ##		4	4		
Intersegment	CB		4	7		
Intersegment Adding Immediate to SP	CA ##		4	7		
Protected Mode: Different Privilege Level						
Intersegment				23		
Intersegment Adding Immediate to SP				23		

\# = immediate 8-bit data
\## = immediate 16-bit data
\### = full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
ROL *Rotate Left*					b	h
Register/Memory by 1	D[000w] [mod 000 r/m]	x - - - - - - - x	1	1		
Register/Memory by CL	D[001w] [mod 000 r/m]	u - - - - - - - x	2	2		
Register/Memory by Immediate	C[000w] [mod 000 r/m] #	u - - - - - - - x	1	1		
ROR *Rotate Right*					b	h
Register/Memory by 1	D[000w] [mod 001 r/m]	x - - - - - - - x	1	1		
Register/Memory by CL	D[001w] [mod 001 r/m]	u - - - - - - - x	2	2		
Register/Memory by Immediate	C[000w] [mod 001 r/m] #	u - - - - - - - x	1	1		
RSDC *Restore Segment Register and Descriptor*	0F 79 [mod sreg3 r/m]	- - - - - - - - -	6	6	s	s
RSLDT *Restore LDTR and Descriptor*	0F 7B [mod 000 r/m]	- - - - - - - - -	6	6	s	s
RSM *Resume from SMM Mode*	0F AA	x x x x x x x x x	40	40	s	s
RSTS *Restore TSR and Descriptor*	0F 7D [mod 000 r/m]	- - - - - - - - -	6	6	s	s
SAHF *Store AH in FLAGS*	9E	- - - - x x x x x	1	1		
SAL *Shift Left Arithmetic*					b	h
Register/Memory by 1	D[000w] [mod 100 r/m]	x - - - x x u x x	1	1		
Register/Memory by CL	D[001w] [mod 100 r/m]	u - - - x x u x x	2	2		
Register/Memory by Immediate	C[000w] [mod 100 r/m] #	u - - - x x u x x	1	1		
SAR *Shift Right Arithmetic*					b	h
Register/Memory by 1	D[000w] [mod 111 r/m]	x - - - x x u x x	1	1		
Register/Memory by CL	D[001w] [mod 111 r/m]	u - - - x x u x x	2	2		
Register/Memory by Immediate	C[000w] [mod 111 r/m] #	u - - - x x u x x	1	1		
SBB *Integer Subtract with Borrow*		x - - - x x x x x			b	h
Register to Register	1[10dw] [11 reg r/m]		1	1		
Register to Memory	1[100w] [mod reg r/m]		1	1		
Memory to Register	1[101w] [mod reg r/m]		1	1		
Immediate to Register/Memory	8[00sw] [mod 011 r/m] ###		1	1		
Immediate to Accumulator (short form)	1[110w] ###		1	1		
SCAS *Scan String*	A [111w]	x - - - x x x x x	2	2	b	h
SETB/SETNAE/SETC *Set Byte on Below/Not Above or Equal/Carry*		- - - - - - - - -				h
To Register/Memory	0F 92 [mod 000 r/m]		1	1		
SETBE/SETNA *Set Byte on Below or Equal/Not Above*		- - - - - - - - -				h
To Register/Memory	0F 96 [mod 000 r/m]		1	1		
SETE/SETZ *Set Byte on Equal/Zero*		- - - - - - - - -				h
To Register/Memory	0F 94 [mod 000 r/m]		1	1		
SETL/SETNGE *Set Byte on Less/Not Greater or Equal*		- - - - - - - - -				h
To Register/Memory	0F 9C [mod 000 r/m]		1	1		

\# = immediate 8-bit data
= immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
SETLE/SETNG *Set Byte on Less or Equal/Not Greater* To Register/Memory	0F 9E [mod 000 r/m]	- - - - - - - - -	1	1		h
SETNB/SETAE/SETNC *Set Byte on Not Below/ Above or Equal/Not Carry* To Register/Memory	0F 93 [mod 000 r/m]	- - - - - - - - -	1	1		h
SETNBE/SETA *Set Byte on Not Below or Equal/Above* To Register/Memory	0F 97 [mod 000 r/m]	- - - - - - - - -	1	1		h
SETNE/SETNZ *Set Byte on Not Equal/Not Zero* To Register/Memory	0F 95 [mod 000 r/m]	- - - - - - - - -	1	1		h
SETNL/SETGE *Set Byte on Not Less/Greater or Equal* To Register/Memory	0F 9D [mod 000 r/m]	- - - - - - - - -	1	1		h
SETNLE/SETG *Set Byte on Not Less or Equal/Greater* To Register/Memory	0F 9F [mod 000 r/m]	- - - - - - - - -	1	1		h
SETNO *Set Byte on Not Overflow* To Register/Memory	0F 91 [mod 000 r/m]	- - - - - - - - -	1	1		h
SETNP/SETPO *Set Byte on Not Parity/Parity Odd* To Register/Memory	0F 9B [mod 000 r/m]	- - - - - - - - -	1	1		h
SETNS *Set Byte on Not Sign* To Register/Memory	0F 99 [mod 000 r/m]	- - - - - - - - -	1	1		h
SETO *Set Byte on Overflow* To Register/Memory	0F 90 [mod 000 r/m]	- - - - - - - - -	1	1		h
SETP/SETPE *Set Byte on Parity/Parity Even* To Register/Memory	0F 9A [mod 000 r/m]	- - - - - - - - -	1	1		h
SETS *Set Byte on Sign* To Register/Memory	0F 98 [mod 000 r/m]	- - - - - - - - -	1	1		h
SGDT *Store GDT Register* To Register/Memory	0F 01 [mod 000 r/m]	- - - - - - - - -	4	4	b,c	h
SHL *Shift Left Logical* Register/Memory by 1 Register/Memory by CL Register/Memory by Immediate	 D [000w] [mod 100 r/m] D [001w] [mod 100 r/m] C [000w] [mod 100 r/m] #	 x - - - x x u x x u - - - x x u x x u - - - x x u x x	 1 2 1	 1 2 1	b	h
SHLD *Shift Left Double* Register/Memory by Immediate Register/Memory by CL	 0F A4 [mod reg r/m] # 0F A5 [mod reg r/m]	u - - - x x u x x	 4 5	 4 5	b	h

\# = immediate 8-bit data
\## = immediate 16-bit data
\### = full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
\+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
SHR *Shift Right Logical*		x - - - x x u x x			b	h
Register/Memory by 1	D [000w] [mod 101 r/m]		1	1		
Register/Memory by CL	D [001w] [mod 101 r/m]	u - - - x x u x x	2	2		
Register/Memory by Immediate	C [000w] [mod 101 r/m] #	u - - - x x u x x	1	1		
SHRD *Shift Right Double*		u - - - x x u x x			b	h
Register/Memory by Immediate	0F AC [mod reg r/m] #		4	4		
Register/Memory by CL	0F AD [mod reg r/m]		5	5		
SIDT *Store IDT Register*		- - - - - - - - - -			b,c	h
To Register/Memory	0F 01 [mod 001 r/m]		4	4		
SLDT *Store LDT Register*		- - - - - - - - - -			a	h
To Register/Memory	0F 00 [mod 000 r/m]			1		
SMINT *Software SMM Entry*	0F 7E	- - - - - - - - - -	55	55	s	s
SMSW *Store Machine Status Word*	0F 01 [mod 100 r/m]	- - - - - - - - - -	6	6	b,c	h
STC *Set Carry Flag*	F9	- - - - - - - - - 1	1	1		
STD *Set Direction Flag*	FD	- 1 - - - - - - - -	7	7		
STI *Set Interrupt Flag*	FB	- - 1 - - - - - - -	7	7		m
STOS *Store String*	A [101w]	- - - - - - - - - -	2	2	b	h
STR *Store Task Register*		- - - - - - - - - -			a	h
To Register/Memory	0F 00 [mod 001 r/m]			4		
SUB *Integer Subtract*		x - - - x x x x x			b	h
Register to Register	2 [10dw] [11 reg r/m]		1	1		
Register to Memory	2 [100w] [mod reg r/m]		1	1		
Memory to Register	2 [101w] [mod reg r/m]		1	1		
Immediate to Register/Memory	8 [00sw] [mod 101 r/m] ###		1	1		
Immediate to Accumulator (short form)	2 [110w] ###		1	1		
SVDC *Save Segment Register and Descriptor*	0F 78 [mod sreg3 r/m]	- - - - - - - - - -	12	12	s	s
SVLDT *Save LDTR and Descriptor*	0F 7A [mod 000 r/m]	- - - - - - - - - -	12	12	s	s
SVTS *Save TSR and Descriptor*	0F 7C [mod 000 r/m]	- - - - - - - - - -	14	14	s	s
TEST *Test Bits*		0 - - - x x u x 0			b	h
Register/Memory and Register	8 [010w] [mod reg r/m]		1	1		
Immediate Data and Register/Memory	F [011w] [mod 000 r/m] ###		1	1		
Immediate Data and Accumulator	A [100w] ###		1	1		
VERR *Verify Read Access*		- - - - - - x - - -			a	g,h,j,p
To Register/Memory	0F 00 [mod 100 r/m]			7		
VERW *Verify Write Access*		- - - - - - x - - -			a	g,h,j,p
To Register/Memory	0F 00 [mod 101 r/m]			7		
WAIT *Wait Until FPU Not Busy*	9B	- - - - - - - - - -	5	5		

\# = immediate 8-bit data
\## = immediate 16-bit data
= full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Table 6-20. **6x86 CPU Instruction Set Clock Count Summary** (Continued)

INSTRUCTION	OPCODE	FLAGS	REAL MODE CLOCK COUNT	PROTECTED MODE CLOCK COUNT	NOTES	
		OF DF IF TF SF ZF AF PF CF	Reg/ Cache Hit	Reg/ Cache Hit	Real Mode	Protected Mode
WBINVD *Write-Back and Invalidate Cache*	0F 09	- - - - - - - - -	15	15	t	t
XADD *Exchange and Add*		x - - - x x x x x				
Register1, Register2	0F C[000w] [11 reg2 reg1]		2	2		
Memory, Register	0F C[000w] [mod reg r/m]		2	2		
XCHG *Exchange*		- - - - - - - - -			b,f	f,h
Register/Memory with Register	8[011w] [mod reg r/m]		2	2		
Register with Accumulator	9[0 reg]		2	2		
XLAT *Translate Byte*	D7	- - - - - - - - -	4	4		h
XOR *Boolean Exclusive OR*		0 - - - x x u x 0			b	h
Register to Register	3 [00dw] [11 reg r/m]		1	1		
Register to Memory	3 [000w] [mod reg r/m]		1	1		
Memory to Register	3 [001w] [mod reg r/m]		1	1		
Immediate to Register/Memory	8 [00sw] [mod 110 r/m] ###		1	1		
Immediate to Accumulator (short form)	3 [010w] ###		1	1		

\# = immediate 8-bit data
\## = immediate 16-bit data
\### = full immediate 32-bit data (8, 16, 32 bits)

\+ = 8-bit signed displacement
+++ = full signed displacement (16, 32 bits)

x = modified
\- = unchanged
u = undefined

Instruction Notes for Instruction Set Summary

Notes a through c apply to Real Address Mode only:

a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in exception 6 (invalid op-code).

b. Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS, or GS segment limit (FFFFH). Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.

c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

d. -

Notes e through g apply to Real Address Mode and Protected Virtual Address Mode:

e. An exception may occur, depending on the value of the operand.

f. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK prefix.

g. LOCK# is asserted during descriptor table accesses.

Notes h through r apply to Protected Virtual Address Mode only:

h. Exception 13 fault will occur if the memory operand in CS, DS, ES, FS, or GS cannot be used due to either a segment limit violation or an access rights violation. If a stack limit is violated, an exception 12 occurs.

i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an exception 13 fault. The segment's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 occurs.

Continued on next page...

j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.
k. JMP, CALL, INT, RET, and IRET instructions referring to another code segment will cause an exception 13, if an applicable privilege rule is violated.
l. An exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).
m. An exception 13 fault occurs if CPL is greater than IOPL.
n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.
o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CRO if desiring to reset the PE bit.
p. Any violation of privilege rules as apply to the selector operand does not cause a Protection exception, rather, the zero flag is cleared.
q. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault will occur before the ESC instruction is executed. An exception 12 fault will occur if the stack limit is violated by the operand's starting address.
r. The destination of a JMP, CALL, INT, RET, or IRET must be in the defined limit of a code segment or an exception 13 fault will occur.

Note s applies to Cyrix specific SMM instructions:

s. All memory accesses to SMM space are non-cacheable. An invalid opcode exception 6 occurs unless SMI is enabled and ARR3 size > 0, and CPL = 0 and [SMAC is set or if in an SMI handler].

Note t applies to cache invalidation instructions with the cache operating in write-back mode:

t. The total clock count is the clock count shown plus the number of clocks required to write all "modified" cache lines to external memory.