6. INSTRUCTION SET

This section summarizes the IBM 6x86 CPU instruction set and provides detailed information on the instruction encodings. All instructions are listed in the CPU Instruction Set Summary Table (Table 6-20, Page 6-14), and the FPU Instruction Set Summary Table (Table 6-22, Page 6-30). These tables provide information on the instruction encoding, and the instruction clock counts for each instruction. The clock count values for both tables are based on the assumptions described in Section 6.3.

6.1 Instruction Set Summary

Depending on the instruction, the IBM 6x86 CPU instructions follow the general instruction format shown in Figure 6-1. These instructions vary in length and can start at any byte address. An instruction consists of one or more bytes that can include: prefix byte(s), at least one opcode byte(s), mod r/m byte, s-i-b byte, address displacement byte(s) and immediate data byte(s). An instruction can be as short as one byte and as long as 15 bytes. If there are more than 15 bytes in the instruction a general protection fault (error code of 0) is generated.

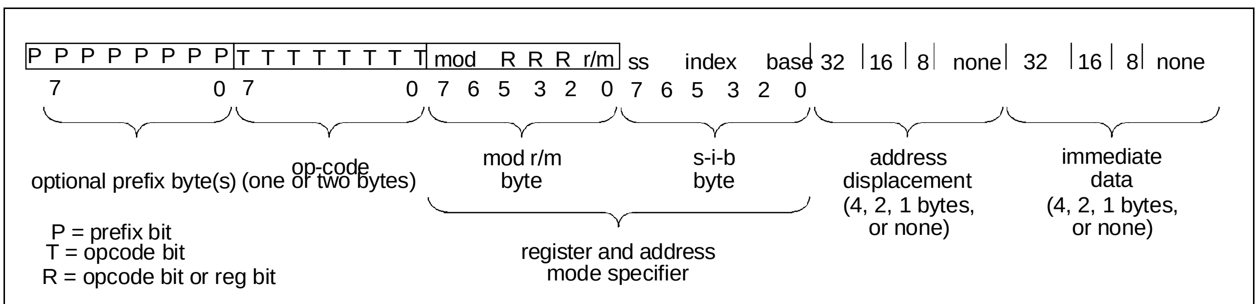

Figure 6-1. Instruction Set Format

6.2 General Instruction Fields

The fields in the general instruction format at the byte level are listed in Table 6-1.

Table 6-1. Instruction Fields

FIELD NAME	DESCRIPTION	WIDTH
Optional Prefix Byte(s)	Specifies segment register override, address and operand size, repeat elements in string instruction, LOCK# assertion.	1 or more bytes
Opcode Byte(s)	Identifies instruction operation.	1 or 2 bytes
mod and r/m Byte	Address mode specifier.	1 byte
s-i-b Byte	Scale factor, Index and Base fields.	1 byte
Address Displacement	Address displacement operand.	1, 2 or 4 bytes
Immediate data	Immediate data operand.	1, 2 or 4 bytes

6.2.1 Optional Prefix Bytes

Prefix bytes can be placed in front of any instruction. The prefix modifies the operation of the next instruction only. When more than one prefix is used, the order is not important. There are five type of prefixes as follows:

1. Segment Override explicitly specifies which segment register an instruction will use for effective address calculation.

2. Address Size switches between 16- and 32-bit addressing. Selects the inverse of the default.

3. Operand Size switches between 16- and 32-bit operand size. Selects the inverse of the default.

4. Repeat is used with a string instruction which causes the instruction to be repeated for each element of the string.

5. Lock is used to assert the hardware LOCK# signal during execution of the instruction.

Table 6-2 lists the encodings for each of the available prefix bytes.

Table 6-2. Instruction Prefix Summary

PREFIX	ENCODING	DESCRIPTION
ES:	26h	Override segment default, use ES for memory operand
CS:	2Eh	Override segment default, use CS for memory operand
SS:	36h	Override segment default, use SS for memory operand
DS:	3Eh	Override segment default, use DS for memory operand
FS:	64h	Override segment default, use FS for memory operand
GS:	65h	Override segment default, use GS for memory operand
Operand Size	66h	Make operand size attribute the inverse of the default
Address Size	67h	Make address size attribute the inverse of the default
LOCK	F0h	Assert LOCK# hardware signal.
REPNE	F2h	Repeat the following string instruction.
REP/REPE	F3h	Repeat the following string instruction.

6.2.2 Opcode Byte

The opcode field specifies the operation to be performed by the instruction. The opcode field is either one or two bytes in length and may be further defined by additional bits in the mod r/m byte. Some operations have more than one opcode, each specifying a different form of the operation. Some opcodes name instruction groups. For example, opcode 80h names a group of operations that have an immediate operand and a register or memory operand. The reg field may appear in the second opcode byte or in the mod r/m byte.

6.2.2.1 w Field

The 1-bit w field (Table 6-11) selects the operand size during 16 and 32 bit data operations.

Table 6-3. w Field Encoding

w FIELD	OPERAND SIZE	
	16-BIT DATA OPERATIONS	32-BIT DATA OPERATIONS
0	8 Bits	8 Bits
1	16 Bits	32 Bits

6.2.2.2 d Field

The d field (Table 6-10) determines which operand is taken as the source operand and which operand is taken as the destination.

Table 6-4. d Field Encoding

d FIELD	DIRECTION OF OPERATON	SOURCE OPERAND	DESTINATION OPERAND
0	Register --> Register or Register --> Memory	reg	mod r/m or mod ss-index-base
1	Register --> Register or Memory --> Register	mod r/m or mod ss-index-base	reg

6.2.2.3 s Field

The s field (Table 6-10) determines the size of the immediate data field. If the S bit is set, the immediate field of the OP code is 8-bits wide and is sign extened to match the operand size of the opcode.

Table 6-5**. s Field Encoding**

s FIELD	Immediate Field Size		
	8-Bit Operand Size	16-Bit Operand Size	32-Bit Operand Size
0 (or not present)	8 bits	16 bits	32 bits
1	8 bits	8 bits (sign extended)	8 bits (sign extended)

6.2.2.4 eee Field

The eee field (Table 6-6) is used to select the control, debug and test registers in the MOV instructions. The type of register and base registers selected by the eee field are listed in Table 6-6. The values shown in Table 6-6 are the only valid encodings for the eee bits.

Table 6-6. eee Field Encoding

eee FILED	REGISTER TYPE	BASE REGISTER
000	Control Register	CR0
010	Control Register	CR2
011	Control Register	CR3
000	Debug Register	DR0
001	Debug Register	DR1
010	Debug Register	DR2
011	Debug Register	DR3
110	Debug Register	DR6
111	Debug Register	DR7
011	Test Register	TR3
100	Test Register	TR4
101	Test Register	TR5
110	Test Register	TR6
111	Test Register	TR7

6.2.3 mod and r/m Byte

The mod and r/m fields (Table 6-7), within the mod r/m byte, select the type of memory addressing to be used. Some instructions use a fixed addressing mode (e.g., PUSH or POP) and therefore, these fields are not present. Table 6-7 lists the addressing method when 16-bit addressing is used and a mod r/m byte is present. Some mod r/m field encodings are dependent on the w field and are shown in Table 6-8 (Page 6-7).

Table 6-7. mod r/m Field Encoding

mod and r/m fields	16-BIT ADDRESS MODE with mod r/m Byte	32-BIT ADDRESS MODE with mod r/m Byte and No s-i-b Byte Present
00 000	DS:[BX+SI]	DS:[EAX]
00 001	DS:[BX+DI]	DS:[ECX]
00 010	DS:[BP+SI]	DS:[EDX]
00 011	DS:[BP+DI]	DS:[EBX]
00 100	DS:[SI]	s-i-b is present (See 6.2.4 (Page 6-9))
00 101	DS:[DI]	DS:[d32]
00 110	DS:[d16]	DS:[ESI]
00 111	DS:[BX]	DS:[EDI]
01 000	DS:[BX+SI+d8]	DS:[EAX+d8]
01 001	DS:[BX+DI+d8]	DS:[ECX+d8]
01 010	DS:[BP+SI+d8]	DS:[EDX+d8]
01 011	DS:[BP+DI+d8]	DS:[EBX+d8]
01 100	DS:[SI+d8]	s-i-b is present (See 6.2.4 (Page 6-9))
01 101	DS:[DI+d8]	SS:[EBP+d8]
01 110	SS:[BP+d8]	DS:[ESI+d8]
01 111	DS:[BX+d8]	DS:[EDI+d8]
10 000	DS:[BX+SI+d16]	DS:[EAX+d32]
10 001	DS:[BX+DI+d16]	DS:[ECX+d32]
10 010	DS:[BP+SI+d16]	DS:[EDX+d32]
10 011	DS:[BP+DI+d16]	DS:[EBX+d32]
10 100	DS:[SI+d16]	s-i-b is present (See 6.2.4 (Page 6-9))
10 101	DS:[DI+d16]	SS:[EBP+d32]
10 110	SS:[BP+d16]	DS:[ESI+d32]
10 111	DS:[BX+d16]	DS:[EDI+d32]
11 000-11 111	See Table 6-7	See Table 6-7

Table 6-8. mod r/m Field Encoding Dependent on w Field

mod r/m	16-BIT OPERATION w = 0	16-BIT OPERATION w = 1	32-BIT OPERATION w = 0	32-BIT OPERATION w = 1
11 000	AL	AX	AL	EAX
11 001	CL	CX	CL	ECX
11 010	DL	DX	DL	EDX
11 011	BL	BX	BL	EBX
11 100	AH	SP	AH	ESP
11 101	CH	BP	CH	EBP
11 110	DH	SI	DH	ESI
11 111	BH	DI	BH	EDI

6.2.3.1 reg Field

The reg field (Table 6-9) determines which general registers are to be used. The selected register is dependent on whether a 16 or 32 bit operation is current and the status of the w bit.

Table 6-9. reg Field

reg	16-BIT OPERATION w Field Not Present	32-BIT OPERATION w Field Not Present	16-BIT OPERATION w = 0	16-BIT OPERATION w = 1	32-BIT OPERATION w = 0	32-BIT OPERATION w = 1
000	AX	EAX	AL	AX	AL	EAX
001	CX	ECX	CL	CX	CL	ECX
010	DX	EDX	DL	DX	DL	EDX
011	BX	EBX	BL	BX	BL	EBX
100	SP	ESP	AH	SP	AH	ESP
101	BP	EBP	CH	BP	CH	EBP
110	SI	ESI	DH	SI	DH	ESI
111	DI	EDI	BH	DI	BH	EDI

6.2.3.2 sreg3 Field

The sreg3 field (Table 6-10) is 3-bit field that is similar to the sreg2 field, but allows use of the FS and GS segment registers.

Table 6-10. sreg3 Field Encoding

sreg3 FIELD	SEGMENT REGISTER SELECTED
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	undefined
111	undefined

6.2.3.3 sreg2 Field

The sreg2 field (Table 6-11) is a 2-bit field that allows one of the four 286-type segment registers to be specified.

Table 6-11. sreg2 Field Encoding

sreg2 FIELD	SEGMENT REGISTER SELECTED
00	ES
01	CS
10	SS
11	DS

6.2.4 s-i-b Byte

The s-i-b fields provide scale factor, indexing and a base field for address selection.

6.2.4.1 ss Field

The ss field (Table 6-12) specifies the scale factor used in the offset mechanism for address calculation. The scale factor multiplies the index value to provide one of the components used to calculate the offset address.

Table 6-12. ss Field Encoding

ss FIELD	SCALE FACTOR
00	x1
01	x2
01	x4
11	x8

6.2.4.2 index Field

The index field (Table 6-13) specifies the index register used by the offset mechanism for offset address calculation. When no index register is used (index field = 100), the ss value must be 00 or the effective address is undefined.

Table 6-13. index Field Encoding

Index FIELD	INDEX REGISTER
000	EAX
001	ECX
010	EDX
011	EBX
100	none
101	EBP
110	ESI
111	EDI

6.2.4.3 Base Field

In Table 6-7 (Page 6-6), the note “s-i-b present” for certain entries forces the use of the mod and base field as listed in Table 6-14. The first two digits in the first column of Table 6-14 identifies the mod bits in the mod r/m byte. The last three digits in the first column of this table identifies the base fields in the s-i-b byte.

Table 6-14. mod base Field Encoding

mod FIELD WITHIN mode/rm BYTE	base FIELD WITHIN s-i-b BYTE	32-BIT ADDRESS MODE with mod r/m and s-i-b Bytes Present
00	000	DS:[EAX+(scaled index)]
00	001	DS:[ECX+(scaled index)]
00	010	DS:[EDX+(scaled index)]
00	011	DS:[EBX+(scaled index)]
00	100	SS:[ESP+(scaled index)]
00	101	DS:[d32+(scaled index)]
00	110	DS:[ESI+(scaled index)]
00	111	DS:[EDI+(scaled index)]
01	000	DS:[EAX+(scaled index)+d8]
01	001	DS:[ECX+(scaled index)+d8]
01	010	DS:[EDX+(scaled index)+d8]
01	011	DS:[EBX+(scaled index)+d8]
01	100	SS:[ESP+(scaled index)+d8]
01	101	SS:[EBP+(scaled index)+d8]
01	110	DS:[ESI+(scaled index)+d8]
01	111	DS:[EDI+(scaled index)+d8]
10	000	DS:[EAX+(scaled index)+d32]
10	001	DS:[ECX+(scaled index)+d32]
10	010	DS:[EDX+(scaled index)+d32]
10	011	DS:[EBX+(scaled index)+d32]
10	100	SS:[ESP+(scaled index)+d32]
10	101	SS:[EBP+(scaled index)+d32]
10	110	DS:[ESI+(scaled index)+d32]
10	111	DS:[EDI+(scaled index)+d32]

6.3 CPUID Instruction

The IBM 6x86 CPU executes the CPUID instruction (opcode 0FA2) as documented in this section only if the CPUID bit in the CCR4 configuration register is set. The CPUID instruction may be used by software to determine the vendor and type of CPU.

When the CPUID instruction is executed with EAX = 0, the ASCII characters “CyrixInstead” are placed in the EBX, EDX, and ECX registers as shown in Table 6-15:

Table 6-15. CPUID Data Returned When EAX = 0

REGISTER	CONTENTS (D31 - D0)
EBX	69 72 79 43 i r y C*
EDX	73 6E 49 78 s n I x*
ECX	64 61 65 74 d a e t*

*ASCII equivalent

When the CPUID instruction is executed with EAX = 1, EAX and EDX contain the values shown in Table 6-16.

Table 6-16. CPUID Data Returned When EAX = 1

REGISTER	CONTENTS
EAX(3-0)	0
EAX(7-4)	2
EAX(11-8)	5
EAX(13-12)	0
EAX(31-14)	reserved
EDX	If EDX = 00, FPU not on-chip. If EDX = 01, FPU on-chip.

6.4 Instruction Set Tables

The IBM 6x86 CPU instruction set is presented in two tables: Table 6-20. "6x86 CPU Instruction Set Clock Count Summary" on page 6-14 and Table 6-22. "6x86 FPU Instruction Set Summary" on page 6-30. Additional information concerning the FPU Instruction Set is presented on page 6-29.

6.4.1 Assumptions Made in Determining Instruction Clock Count

The assumptions made in determining instruction clock counts are listed below:

1. All clock counts refer to the internal CPU internal clock frequency. For example, the clock counts for a clock-doubled IBM 6x86 CPU-100 refer to 100 MHz clocks while the external clock is 50 MHz.

2. The instruction has been prefetched, decoded and is ready for execution.

3. Bus cycles do not require wait states.

4. There are no local bus HOLD requests delaying processor access to the bus.

5. No exceptions are detected during instruction execution.

6. If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock count shown. However, if the effective address calculation uses two general register components, add 1 clock to the clock count shown.

7. All clock counts assume aligned 32-bit memory/IO operands.

8. If instructions access a 32-bit operand that crosses a 64-bit boundary, add 1 clock for read or write and add 2 clocks for read and write.

9. For non-cached memory accesses, add two clocks (IBM 6x86 CPU with 2x clock) or four clocks (IBM 6x86 CPU with 3x clock). (Assumes zero wait state memory accesses).

10. Locked cycles are not cacheable. Therefore, using the LOCK prefix with an instruction adds additional clocks as specified in paragraph 9 above.

11. No parallel execution of instructions.

6.4.2 CPU Instruction Set Summary Table Abbreviations

The clock counts listed in the CPU Instruction Set Summary Table are grouped by operating mode and whether there is a register/cache hit or a cache miss. In some cases, more than one clock count is shown in a column for a given instruction, or a variable is used in the clock count. The abbreviations used for these conditions are listed in Table 6-17.

Table 6-17. CPU Clock Count Abbreviations

<table>
<tr><th>CLOCK COUNT SYMBOL</th><th>EXPLANATION</th></tr>
<tr><td>/</td><td>Register operand/memory operand.</td></tr>
<tr><td>n</td><td>Number of times operation is repeated.</td></tr>
<tr><td>L</td><td>Level of the stack frame.</td></tr>
<tr><td>|</td><td>Conditional jump taken | Conditional jump not taken.
(e.g. “4|1” = 4 clocks if jump taken, 1 clock if jump not taken)</td></tr>
<tr><td>\</td><td>CPL ≤ IOPL \ CPL > IOPL
(where CPL = Current Privilege Level, IOPL = I/O Privilege Level)</td></tr>
<tr><td>m</td><td>Number of parameters passed on the stack.</td></tr>
</table>

6.4.3 CPU Instruction Set Summary Table Flags Table

The CPU Instruction Set Summary Table lists nine flags that are affected by the execution of instructions. The conventions shown in Table 6-18 are used to identify the different flags. Table 6-19 lists the conventions used to indicate what action the instruction has on the particular flag.

Table 6-18. Flag Abbreviations

ABBREVIATION	NAME OF FLAG
OF	Overflow Flag
DF	Direction Flag
IF	Interrupt Enable Flag
TF	Trap Flag
SF	Sign Flag
ZF	Zero Flag
AF	Auxiliary Flag
PF	Parity Flag
CF	Carry Flag

Table 6-19. Action of Instruction on Flag

INSTRUCTION TABLE SYMBOL	ACTION
x	Flag is modified by the instruction.
-	Flag is not changed by the instruction.
0	Flag is reset to “0”.
1	Flag is set to “1”.
u	Flag is undefined following execution of the instruction.