cute simple instructions at a high rate—perhaps one instruction per cycle. Others believe that a microprocessor should execute more complex instructions at a lower rate.

Operand types add complexity to an instruction set when a single general operation such as addition can be invoked with many different addressing modes. Motorola's CISC 68000 contains a basic addition instruction, among other addition operations, that can be decoded in many different ways according to the specified addressing mode. Table 7.1 shows the format of the basic ADD/ADDA/ADDX instruction word. ADD is used for operations primarily on data registers. ADDA is used for operations primarily on address registers. ADDX is used for special addition operations that incorporate the ALU extended carry bit, X, into the sum. The instruction word references Register1 directly and an effective address (EA) that can represent another register or various types of indirect and indexed addressing modes.

TABLE 7.1 68000 ADD/ADDA/ADDX Instruction Word

Bit Position	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Opcode = 1101				Register1			Opmode			Effective Address					
											Mode			Register2		

As listed in Table 7.2, the opmode field defines whether the operands are 8-, 16-, or 32-bit quantities and identifies the source and destination operands. In doing so, it also implies certain subclasses of instructions: ADD, ADDA, or ADDX.

TABLE 7.2 68000 ADD/ADDA/ADDX Instruction Opmode Field

Opmode Value	Operand Width	Definition of Register1	Operation	Instruction Mapping
000	8	Dn	EA + Dn ⇒ Dn	ADD
001	16	Dn	EA + Dn ⇒ Dn	ADD
010	32	Dn	EA + Dn ⇒ Dn	ADD
100	8	Dn	Dn + EA ⇒ EA	ADD/ADDX
101	16	Dn	Dn + EA ⇒ EA	ADD/ADDX
110	32	Dn	Dn + EA ⇒ EA	ADD/ADDX
011	16	An	EA + An ⇒ An	ADDA
111	32	An	EA + An ⇒ An	ADDA

The main complexity is introduced by the EA fields as defined in Table 7.3. For those modes that map to multiple functions, additional identifying fields and operands are identified by one or more extension words that follow the instruction word. One of the more complex modes involves using an address register as a base address, adding a displacement to that base to calculate a fetch address, fetching the data at that address, adding another register to the retrieved value, adding another displacement, and then using the resulting address to fetch a final operand value. ADD/ADDA/ADDX is

a powerful instruction that requires significant decode logic behind it. Additionally, when opmode indicates an ADD or ADDX instruction, the two mode values that normally indicate simple register references now map to one of two special ADDX operations.

TABLE 7.3 68000 Effective Address Field Definition

Mode Field	Definition of Register2	Operand Value	Function
000	Data register N	Dn	Data register value
001	Address register N	An	Address register value
010	Address register N	(An)	Indirect address register
011	Address register N	(An)+	Indirect with post-increment
100	Address register N	–(An)	Indirect with pre-decrement
101	Address register N	(An + d_{16},)	Indirect with 16-bit displacement
110	Address register N	(An + Xn + d_8)	Indirect with index register and 8-bit displacement (extension word follows)
		(An + Xn + $d_{16,32}$)	Indirect with index register and 32- or 16-bit displacement (extension words follow)
		((An + $d_{16,32}$) + Xn + $d_{16,32}$)	Indirect with displacement to fetch pointer added to index register and displacement (extension words follow)
		((An + Xn + $d_{16,32}$) + $d_{16,32}$)	Indirect with displacement and index register to fetch pointer added to displacement (extension words follow)
111	000	d_{16}	16-bit direct address (extension word follows)
111	001	d_{32}	32-bit direct address (extension words follow)
111	100	#data	Immediate follows in extension words
111	010	(PC + d_{16})	Indirect with 16-bit displacement
111	011	<multiple>	Same as mode=110, but with PC instead of address registers
Shaded modes are invalid when EA is specified as the destination by *opmode* and change their meaning as follows:			
000	Data register N	Dn	ADDX: Dregister2 + Dregister1 + X ⇒ Dregister1
001	Address register N	–(An)	ADDX: –(Aregister2) + –(Aregister1) + X ⇒ (Aregister1)

As can be readily observed, decoding an addition instruction on the 68000 is not as simple as adding two registers. For the most complex addressing modes, multiple registers must be added together to create an address from which another address is fetched that is added with an offset to yield a final address at which the true operand is located. This sounds complicated, and it is. There is really no succinct way to explain the operation of such instructions. The impact of these complex addressing modes on decoding logic is substantial, especially when it is realized that the 68000 contains dozens of instructions, each with its own permutations.