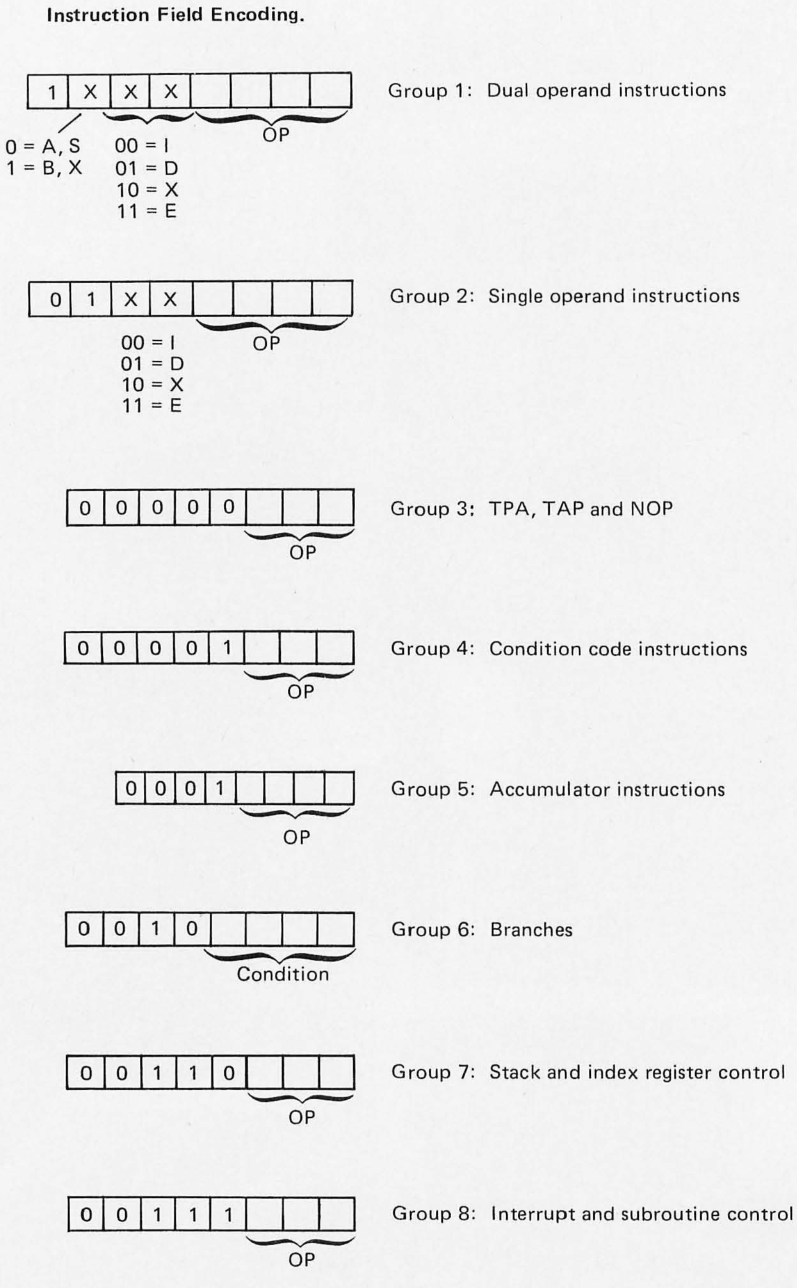
The Motorola 6800 Instruction Set

Two Programming Points of View

Paul M Jessop
1157 Warwick Rd
Solihull
West Midlands B91 3HQ
ENGLAND



Instruction Field Encoding.

1 X X X
0 = A, S
1 = B, X
00 = I
01 = D
10 = X
11 = E
OP
Group 1: Dual operand instructions
0 1 X X
00 = I
01 = D
10 = X
11 = E
OP
Group 2: Single operand instructions
0 0 0 0 0
OP
Group 3: TPA, TAP and NOP
0 0 0 0 1
OP
Group 4: Condition code instructions
0 0 0 1
OP
Group 5: Accumulator instructions
0 0 1 0
Condition
Group 6: Branches
0 0 1 1 0
OP
Group 7: Stack and index register control
0 0 1 1 1
OP
Group 8: Interrupt and subroutine control

Figure 1: One way to organize one's viewpoint of the Motorola 6800 instruction set is to view it as a number of instruction groupings, broken down by internal binary fields for selection of instructions within the group. This viewpoint is most appropriate for those working directly in binary, or organizing the code generation parts of an assembler or compiler.

When faced with the problem of trying to hand assemble a machine language program, the task of looking up each of the op codes in the manufacturer's data can be quite daunting. Admittedly, some become familiar before very long but the less common instructions still cause problems (do you 6800 users remember the hexadecimal op code for TPA?). Two solutions to this dilemma are suggested here. The first is suitable for "switch flippers" and the second for users of MIKBUG and other systems with hexadecimal dump and load functions.

The First Solution: Use Instruction Fields

Anyone who has seen the programming books for the DEC PDP 8 will be familiar with the principles involved. In the PDP 8 instruction set, the first three bits of the 12 bit word define the type of instruction and the remaining bits each have a separate function. This is of course a gross simplification and is not true for memory reference and IO instructions but it underlines the basic ideas. Now, study of the 6800 op codes reveals some interesting facts at the bit level. These are outlined in figure 1.

These patterns are naturally related to the instruction decoding which goes on inside the chip, but they are a godsend to the programmer who must work in binary. A couple of words of explanation are needed. Branch to subroutine occurs in an unexpected place but it is easy to remember if thought of as Jump, mode immediate. The generalization in group 1 bit 6 that a zero implies accumulator or stack pointer addressing does not hold true for compare index register (CPX), where it implies index register addressing. Naturally, the store instructions (STA, STS and STX) do not exist in immediate mode in the published definition of the 6800 instruction set.

The Second Solution: The Ordered Manual Lookup Table

The appropriate information is contained in figure 2. This should be a great boon to anyone who, for lack of memory or IO devices, has no assembler. The table is arranged in such a way that the first hexadecimal digit is the horizontal coordinate, just as the x component comes first in a pair of Cartesian coordinates. The credit for inspiring this technique must go to Mr Fugitt (March 77 BYTE, page 36) for his 6502 table, but this table for the 6800 is somewhat more useful for both assembling and disassembling because of the way the codes fall into groups and the addressing modes fall into neat vertical lines.

By way of a final word, the table can, if reduced small enough, make a very handy reference card. Mine has, on the front, tables to convert between hexadecimal, octal and decimal, and, on the reverse, the conditions required for branches, the restart vectors, details of the control code register and the stack register. ■

The Ordered Manual Lookup Table

🟨 Accumulator A as one operand

🟥 Accumulator B as one operand

🟩 Miscellaneous instructions

● Unimplemented

▨ Undocumented instruction:

NBA = And accumulators
HCF = Halt and catch fire
STS, STX, STA, STB = store immediates

See "Undocumented 6800 Instructions" by Gerry Wheeler, page 46, December 1977 BYTE.

Low (Second) Nybble					ACCA	ACCB	X	E	I	D	X	E	I	D	X	E
F	SEI	•	BLE	SWI	CLR	CLR	CLR	CLR	STS	STS	STS	STS	STX	STX	STX	STX
E	CLI	•	BGT	WAI	•	•	JMP	JMP	LDS	LDS	LDS	LDS	LDX	LDX	LDX	LDX
D	SEC	•	BLT	•	TST	TST	TST	TST	BSR	HCF	JSR	JSR	•	HCF	•	•
C	CLC	•	BGE	•	INC	INC	INC	INC	CPX	CPX	CPX	CPX	•	•	•	•
B	SEV	ABA	BMI	RTI	•	•	•	•	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD
A	CLV	•	BPL	•	DEC	DEC	DEC	DEC	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA
9	DEX	DAA	BVS	RTS	ROL	ROL	ROL	ROL	ADC	ADC	ADC	ADC	ADC	ADC	ADC	ADC
8	INX	•	BVC	•	ASL	ASL	ASL	ASL	EOR	EOR	EOR	EOR	EOR	EOR	EOR	EOR
7	TPA	TBA	BEQ	PSH B	ASR	ASR	ASR	ASR	STA	STA	STA	STA	STA	STA	STA	STA
6	TAP	TAB	BNE	PSH A	ROR	ROR	ROR	ROR	LDA	LDA	LDA	LDA	LDA	LDA	LDA	LDA
5	•	•	BCS	TXS	•	•	•	•	BIT	BIT	BIT	BIT	BIT	BIT	BIT	BIT
4	•	NBA	BCC	DES	LSR	LSR	LSR	LSR	AND	AND	AND	AND	AND	AND	AND	AND
3	•	•	BLS	PUL B	COM	COM	COM	COM	•	•	•	•	•	•	•	•
2	•	•	BHI	PUL A	•	•	•	•	SBC	SBC	SBC	SBC	SBC	SBC	SBC	SBC
1	NOP	CBA	•	INS	•	•	•	•	CMP	CMP	CMP	CMP	CMP	CMP	CMP	CMP
0	•	SBA	BRA	TSX	NEG	NEG	NEG	NEG	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

High (First) Nybble

Figure 2: A second way of viewing the 6800 instruction set is from the viewpoint of a hexadecimal matrix. Here a map of the 6800 instruction set has been broken up into several overall regions, with color coding indicating references to accumulators A and B. Unimplemented and undocumented instructions are shown with a black dot; undocumented, but implemented instructions are shown with cross hatching to indicate "use at own risk."