MC6801/03 INSTRUCTION SET SUMMARY

The following was extracted from the “ MC6801 8-bit Single-Chip Mircocomputer REFERENCE MANUAL (including MC6803 and MC68701)”, Motorola literature number MC6801RM(AD2), copyright Motorola Inc., 1983.

A programming model of the MC6801 is shown in Figure 4-1. The MPU includes two 8-bit accumulators, A and B, which can be concatenated to form a double byte accumulator referred to as accumulator D or A:B where accumulator A contains the most significant byte. The MPU also includes a 16-bit Index Register, a 16-bit Stack Pointer, a 6-bit Condition Code Register, and a 16-bit Program Counter. The MC6801 programming model is identical to the model for the MC6800 except that the two accumulators can be concatenated for double byte instructions.

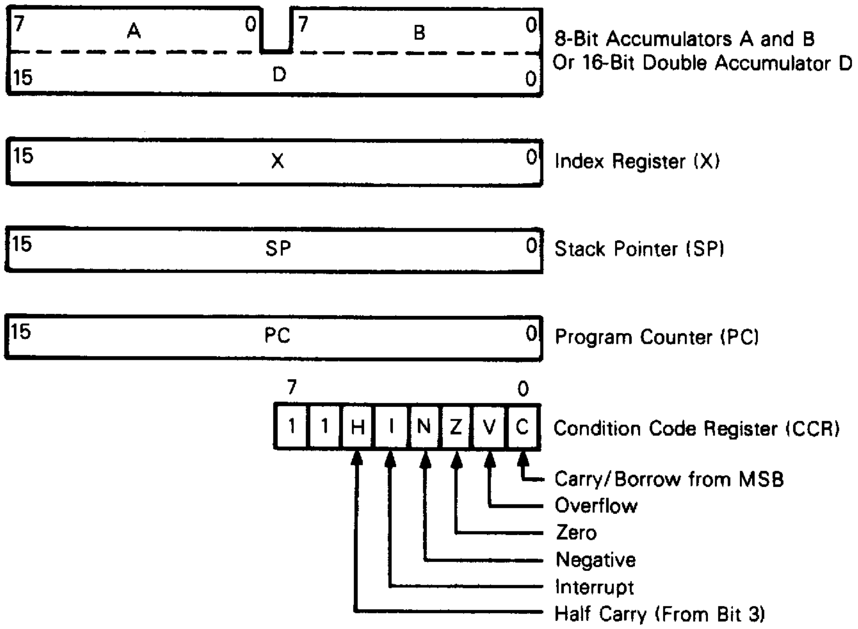
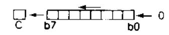
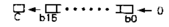
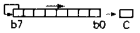
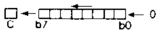
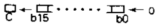
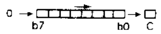
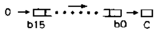
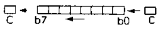
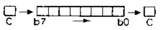


Figure 4.1 MPU Programming Model

6801/03 Instruction Set

Accumulator & Memory Operations	MNE	Immed			Direct			Index			Extend			Inher			Boolean Expression	Condition Codes					
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	●	↕	↕	↕	↕
Add B to X	ABX													3A	3	1	00:B + X → X	●	●	●	●	●	●
Add with Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → A	↕	●	↕	↕	↕	↕
Add Double	ADDD	C3	4	3	D3	5	2	E3	6	2	F3	6	3				D + M:M + 1 → D	●	●	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A•M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B•M → B	●	●	↕	↕	R	●
Shift Left,	ASL							68	6	2	78	6	3					●	●	↕	↕	↕	↕
Arithmetic	ASLA													48	2	1		●	●	↕	↕	↕	↕
	ASLB													58	2	1		●	●	↕	↕	↕	↕
Shift Left Dbl	ASLD													05	3	1		●	●	↕	↕	↕	↕
Shift Right,	ASR							67	6	2	77	6	3					●	●	↕	↕	↕	↕
Arithmetic	ASRA													47	2	1		●	●	↕	↕	↕	↕
	ASRB													57	2	1		●	●	↕	↕	↕	↕
Bit Test	BITA	85	2	2	95	3	2	A5	4	2	B5	4	3				A•M	●	●	↕	↕	R	●
	BITB	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B•M	●	●	↕	↕	R	●
Compare Acmltrs	CBA													11	2	1	A − B	●	●	↕	↕	↕	↕
Clear	CLR							6F	6	2	7F	6	3				00 → M	●	●	R	S	R	R
	CLRA													4F	2	1	00 → A	●	●	R	S	R	R
	CLRB													5F	2	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A − M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B − M	●	●	↕	↕	↕	↕
1's Complement	COM							63	6	2	73	6	3				$\overline{M}$ → M	●	●	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$ → A	●	●	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$ → B	●	●	↕	↕	R	S
Decimal Adj, A	DAA													19	2	1	Adj binary sum to BCD	●	●	↕	↕	↕	↕
Decrement	DEC							6A	6	2	7A	6	3				M − 1 → M	●	●	↕	↕	↕	●
	DECA													4A	2	1	A − 1 → A	●	●	↕	↕	↕	●
	DECB													5A	2	1	B − 1 → B	●	●	↕	↕	↕	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	●	●	↕	↕	R	●
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	●	●	↕	↕	↕	●
	INCA													4C	2	1	A + 1 → A	●	●	↕	↕	↕	●
	INCB													5C	2	1	B + 1 → B	●	●	↕	↕	↕	●
Load Acmltrs	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	●	●	↕	↕	R	●
Load Double	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M:M + 1 → D	●	●	↕	↕	R	●
Logical Shift,	LSL							65	6	2	78	6	3					●	●	↕	↕	↕	↕
Left	LSLA													48	2	1		●	●	↕	↕	↕	↕
	LSLB													58	2	1		●	●	↕	↕	↕	↕
	LSLD													05	3	1		●	●	↕	↕	↕	↕
Shift Right,	LSR							64	6	2	74	6	3					●	●	R	↕	↕	↕
Logical	LSRA													44	2	1		●	●	R	↕	↕	↕
	LSRB													54	2	1		●	●	R	↕	↕	↕
	LSRD													04	3	1		●	●	R	↕	↕	↕
Multiply	MUL													3D	10	1	A*B → D	●	●	●	●	●	↕
2's Complement	NEG							60	6	2	70	6	3				00 − M → M	●	●	↕	↕	↕	↕
(Negate)	NEGA													40	2	1	00 − A → A	●	●	↕	↕	↕	↕
	NEGB													50	2	1	00 − B → B	●	●	↕	↕	↕	↕
No Operation	NOP													01	2	1	PC + 1 → PC	●	●	●	●	●	●
Inclusive OR	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	3	1	A → Stack	●	●	●	●	●	●
	PSHB													37	3	1	B → Stack	●	●	●	●	●	●
Pull Data	PULA													32	4	1	Stack → A	●	●	●	●	●	●
	PULB													33	4	1	Stack → B	●	●	●	●	●	●
Rotate Left	ROL							69	6	2	79	6	3					●	●	↕	↕	↕	↕
	ROLA													49	2	1		●	●	↕	↕	↕	↕
	ROLB													59	2	1		●	●	↕	↕	↕	↕
Rotate Right	ROR							66	6	2	76	6	3					●	●	↕	↕	↕	↕
	RORA													46	2	1		●	●	↕	↕	↕	↕
	RORB													56	2	1		●	●	↕	↕	↕	↕
Subtract Acmltr	SBA													10	2	1	A − B → A	●	●	↕	↕	↕	↕
Subtract with	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A − M − C → A	●	●	↕	↕	↕	↕
Carry	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B − M − C → B	●	●	↕	↕	↕	↕
Store Acmltrs	STAA				97	3	2	A7	4	2	B7	4	3				A → M	●	●	↕	↕	R	●
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	●	●	↕	↕	R	●
	STD				DD	4	2	ED	5	2	FD	5	3				D → M:M + 1	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A − M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B − M → B	●	●	↕	↕	↕	↕
Subtract Double	SUBD	83	4	3	93	5	2	A3	6	2	B3	6	3				D − M:M + 1 → D	●	●	↕	↕	↕	↕
Transfer Acmltr	TAB													16	2	1	A → B	●	●	↕	↕	R	●
	TBA													17	2	1	B → A	●	●	↕	↕	R	●
Test, Zero or	TST							6D	6	2	7D	6	3				M − 00	●	●	↕	↕	R	R
Minus	TSTA													4D	2	1	A − 00	●	●	↕	↕	R	R
	TSTB													5D	2	1	B − 00	●	●	↕	↕	R	R

Jump and Branch Operations	MNE	Relative			Direct			Index			Extend			Inherent			Branch Test	Condition Codes					
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	3	2													C=0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	3	2													C=1	•	•	•	•	•	•
Branch If=Zero	BEQ	27	3	2													Z=1	•	•	•	•	•	•
Branch If≥Zero	BGE	2C	3	2													N⊕V=0	•	•	•	•	•	•
Branch If>Zero	BGT	2E	3	2													Z+(N⊕V)=0	•	•	•	•	•	•
Branch If Higher	BHI	22	3	2													C+Z=0	•	•	•	•	•	•
Branch If ≤Zero	BLE	2F	3	2													Z+(N⊕V)=1	•	•	•	•	•	•
Branch If Lower or Same	BLS	23	3	2													C+Z=1	•	•	•	•	•	•
Branch If <Zero	BLT	2D	3	2													N⊕V=1	•	•	•	•	•	•
Branch If Minus	BMI	2B	3	2													N=1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	3	2													Z=0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	3	2													V=0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	3	2													V=1	•	•	•	•	•	•
Branch If Plus	BPL	2A	3	2													N=0	•	•	•	•	•	•
Branch Never	BRN	21	3	2													None	•	•	•	•	•	•
Branch If Higher or Same	BHS	24	3	2													C=0	•	•	•	•	•	•
Branch If Lower	BLO	25	3	2													C=1	•	•	•	•	•	•
Branch to Subroutine	BSR	8D	6	2														•	•	•	•	•	•
Jump	JMP							6E	3	2	7E	3	3					•	•	•	•	•	•
Jump to Subroutine	JSR				9D	5	2	AD	6	2	BD	6	3					•	•	•	•	•	•
No Operation	NOP													01	2	1		•	•	•	•	•	•
Return from Interrupt	RTI													3B	10	1		↕	↕	↕	↕	↕	↕
Return from Subroutine	RTS													39	5	1		•	•	•	•	•	•
Software Interrupt	SWI													3F	12	1		•	S	•	•	•	•
Wait for Interrupt	WAI													3E	9	1		•	•	•	•	•	•

Index Register Operations	MNE	Immed			Direct			Index			Extend			Inherent			Boolean Expression	Condition Codes					
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#		H	I	N	Z	V	C
Compare Index Register	CPX	8C	4	3	9C	5	2	AC	6	2	BC	6	3				X−M:M+1	•	•	↕	↕	↕	↕
Decrement Index Register	DEX													09	3	1	X−1→X	•	•	•	↕	•	•
Increment Index Register	INX													08	3	1	X+1→X	•	•	•	↕	•	•
Load Index Register	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M:M+1→X	•	•	↕	↕	R	•
Store Index Register	STX				DF	4	2	EF	5	2	FF	5	3				X→M:M+1	•	•	↕	↕	R	•
Add B to Index Register	ABX													3A	3	1	00:B+X→X	•	•	•	•	•	•
Push Index Register	PSHX													3C	4	1	X→Stack	•	•	•	•	•	•
Pull Index Register	PULX													38	5	1	Stack→X	•	•	•	•	•	•
Transfer X to SP	TXS													35	3	1	X−1→SP	•	•	•	•	•	•
Transfer SP to X	TSX													30	3	1	SP+1→X	•	•	•	•	•	•

Stack Pointer Operations	MNE	Immed			Direct			Index			Extend			Inherent			Boolean Expression	Condition Codes					
		Op	~	#	Op	~	#	Op	~	#	Op	~	#	Op	~	#		H	I	N	Z	V	C
Decrement Stack Pointer	DES													34	3	1	SP−1→SP	•	•	•	•	•	•
Increment Stack Pointer	INS													31	3	1	SP+1→SP	•	•	•	•	•	•
Load Stack Pointer	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M:M+1→SP	•	•	↕	↕	R	•
Store Stack Pointer	STS				9F	4	2	AF	5	2	BF	5	3				SP→M:M+1	•	•	↕	↕	R	•
Transfer X to SP	TXS													35	3	1	X−1→SP	•	•	•	•	•	•
Transfer SP to X	TSX													30	3	1	SP+1→X	•	•	•	•	•	•

Condition Code Register Operations	MNE	Inherent			Boolean Operation	Condition Codes					
		Op	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0→C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0→I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0→V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1→C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1→I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1→V	•	•	•	•	S	•
Accumulator A→CCR	TAP	06	2	1	A→CCR	↕	↕	↕	↕	↕	↕
CCR→Accumulator A	TPA	07	2	1	CCR→A	•	•	•	•	•	•

LEGEND:

OP	Operation Code (Hexadecimal)	+	Boolean Inclusive OR	H	Half Carry From Bit 3	S	Set Always
~	Number of MPU Cycles	⊕	Boolean Exclusive OR	I	Interrupt Mask	↕	Affects the Particular CCR Bit
#	Number of Program Bytes	M	Memory Contents	Z	Zero (Byte)	•	Not Affected
+	Arithmetic Plus	A	Accumulator A	V	Overflow (2's Complement)	CCR	Condition Code Register
−	Arithmetic Minus	B	Accumulator B	C	Carry From Bit 7	:	Concatenate
•	Boolean AND	→	Transfer Into	R	Reset (Clear) Always	D	A:B