บรรณานุกรม

1. Lance A. Leventhal in Interoduction to Microprocessors pp. 127-165, Prentice/Hall International, 1978.

2. MOTOROLA. M6800 Programming Reference Manual pp. 2-1-4-7, Motorola, Inc, 1976.

3. Kenneth J. Danhof, Carol L. Smith. Computer System Fundamentals pp. 289-299, Addison-Wesley Publishing Company, 1981.

4. D.W. Barron. Assembler and Loaders, Second Edition, pp. 30-79, Macdonald, 1972.

5. NEC. PC8001B N-Basic Reference Manual, Nippon Electric Co, Ltd. 1981.

ภาคผนวก

Complete ASCII

David M Ciemiewicz
533 N Holly St
Elizabethtown PA 17022

Most of the time when you see a magazine article that requires an ASCII table, the table accompanying the article is either incomplete or is in a numeric system that you cannot use without converting it.

The table I have devised is complete 128 character ASCII. Each character is accompanied by its binary, octal, decimal and hexadecimal equivalent.

This table has proven invaluable to me, as I am sure it will to you.■

Character	Binary	Bit 7 to Bit 0	Octal	Decimal	Hexadecimal
`	01100000		140	096	60
a	01100001		141	097	61
b	01100010		142	098	62
c	01100011		143	099	63
d	01100100		144	100	64
e	01100101		145	101	65
f	01100110		146	102	66
g	01100111		147	103	67
h	01101000		150	104	68
i	01101001		151	105	69
j	01101010		152	106	6A
k	01101011		153	107	6B
l	01101100		154	108	6C
m	01101101		155	109	6D
n	01101110		156	110	6E
o	01101111		157	111	6F
p	01110000		160	112	70
q	01110001		161	113	71
r	01110010		162	114	72
s	01110011		163	115	73
t	01110100		164	116	74
u	01110101		165	117	75
v	01110110		166	118	76
w	01110111		167	119	77
x	01111000		170	120	78
y	01111001		171	121	79
z	01111010		172	122	7A
{	01111011		173	123	7B
|	01111100		174	124	7C
}	01111101		175	125	7D
~	01111110		176	126	7E
DEL	01111111		177	127	7F

Note: The bit 7 in the binary column is sometimes a 1 or is sometimes used as a parity bit.

128 CHARACTER ASCII TABLE

Character	Binary	Bit 7 to Bit 0	Octal	Decimal	Hexadecimal
NUL	00000000		000	000	00
SOH	00000001		001	001	01
STX	00000010		002	002	02
ETX	00000011		003	003	03
EOT	00000100		004	004	04
ENQ	00000101		005	005	05
ACK	00000110		006	006	06
BEL	00000111		007	007	07
BS	00001000		010	008	08
HT	00001001		011	009	09
LF	00001010		012	010	0A
VT	00001011		013	011	0B
FF	00001100		014	012	0C
CR	00001101		015	013	0D
SO	00001110		016	014	0E
SI	00001111		017	015	0F
DLE	00010000		020	016	10
DC1	00010001		021	017	11
DC2	00010010		022	018	12
DC3	00010011		023	019	13
DC4	00010100		024	020	14
NAK	00010101		025	021	15
SYN	00010110		026	022	16
ETB	00010111		027	023	17
CAN	00011000		030	024	18
EM	00011001		031	025	19
SUB	00011010		032	026	1A
ESC	00011011		033	027	1B
FS	00011100		034	028	1C
GS	00011101		035	029	1D
RS	00011110		036	030	1E
US	00011111		037	031	1F
SP	00100000		040	032	20
!	00100001		041	033	21
"	00100010		042	034	22
#	00100011		043	035	23
$	00100100		044	036	24
%	00100101		045	037	25
&	00100110		046	038	26
'	00100111		047	039	27
(	00101000		050	040	28
)	00101001		051	041	29
*	00101010		052	042	2A
+	00101011		053	043	2B
,	00101100		054	044	2C
-	00101101		055	045	2D
.	00101110		056	046	2E
/	00101111		057	047	2F
0	00110000		060	048	30
1	00110001		061	049	31
2	00110010		062	050	32
3	00110011		063	051	33
4	00110100		064	052	34
5	00110101		065	053	35
6	00110110		066	054	36
7	00110111		067	055	37
8	00111000		070	056	38
9	00111001		071	057	39
:	00111010		072	058	3A
;	00111011		073	059	3B
<	00111100		074	060	3C
=	00111101		075	061	3D
>	00111110		076	062	3E
?	00111111		077	063	3F
@	01000000		100	064	40
A	01000001		101	065	41
B	01000010		102	066	42
C	01000011		103	067	43
D	01000100		104	068	44
E	01000101		105	069	45
F	01000110		106	070	46
G	01000111		107	071	47
H	01001000		110	072	48
I	01001001		111	073	49
J	01001010		112	074	4A
K	01001011		113	075	4B
L	01001100		114	076	4C
M	01001101		115	077	4D
N	01001110		116	078	4E
O	01001111		117	079	4F
P	01010000		120	080	50
Q	01010001		121	081	51
R	01010010		122	082	52
S	01010011		123	083	53
T	01010100		124	084	54
U	01010101		125	085	55
V	01010110		126	086	56
W	01010111		127	087	57
X	01011000		130	088	58
Y	01011001		131	089	59
Z	01011010		132	090	5A
[	01011011		133	091	5B
\	01011100		134	092	5C
]	01011101		135	093	5D
^	01011110		136	094	5E
_	01011111		137	095	5F

Abbreviations for Control Characters:

NUL – null, or all zeros
SOH – start of heading
STX – start of text
ETX – end of text
EOT – end of transmission
ENQ – enquiry
ACK – acknowledge
BEL – bell
BS – backspace
HT – horizontal tabulation
LF – line feed
VT – vertical tabulation
FF – form feed
CR – carriage return
SO – shift out
SI – shift in
DLE – data link escape
DC1 – device control 1
DC2 – device control 2
DC3 – device control 3
DC4 – device control 4
NAK – negative acknowledge
SYN – synchronous idle
ETB – end of transmission block
CAN – cancel
EM – end of medium
SUB – substitute
ESC – escape
FS – file separator
GS – group separator
RS – record separator
US – unit separator
SP – space
DEL – delete

ภาคผนวก ข.

The Motorola 6800 Instruction Set[1]

MPU INSTRUCTION SET

The MC6800 has a set of 72 different instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions (Tables 2 thru 6).

MPU ADDRESSING MODES

The MC6800 eight-bit microprocessing unit has seven address modes that can be used by a programmer, with the addressing mode a function of both the type of instruction and the coding within the instruction. A summary of the addressing modes for a particular instruction can be found in Table 7 along with the associated instruction execution time that is given in machine cycles. With a clock frequency of 1 MHz, these times would be microseconds

Accumulator (ACCX) Addressing – In accumulator only addressing, either accumulator A or accumulator B is specified. These are one-byte instructions.

Immediate Addressing – In immediate addressing, the operand is contained in the second byte of the instruction except LDS and LDX which have the operand in the second and third bytes of the instruction. The MPU addresses this location when it fetches the immediate instruction for execution. These are two or three-byte instructions.

Direct Addressing – In direct addressing, the address of the operand is contained in the second byte of the instruction. Direct addressing allows the user to directly address the lowest 256 bytes in the machine i.e., locations zero through 255. Enhanced execution times are achieved by storing data in these locations. In most configurations, it should be a random access memory. These are two-byte instructions.

Extended Addressing – In extended addressing, the address contained in the second byte of the instruction is used as the higher eight-bits of the address of the operand. The third byte of the instruction is used as the lower eight-bits of the address for the operand. This is an absolute address in memory. These are three-byte instructions.

Indexed Addressing – In indexed addressing, the address contained in the second byte of the instruction is added to the index register's lowest eight bits in the MPU. The carry is then added to the higher order eight bits of the index register. This result is then used to address memory. The modified address is held in a temporary address register so there is no change to the index register. These are two-byte instructions.

Implied Addressing – In the implied addressing mode the instruction gives the address (i.e., stack pointer, index register, etc.). These are one-byte instructions.

Relative Addressing – In relative addressing, the address contained in the second byte of the instruction is added to the program counter's lowest eight bits plus two. The carry or borrow is then added to the high eight bits. This allows the user to address data within a range of -125 to +129 bytes of the present instruction. These are two-byte instructions.

---

[1]Courtesy of Motorola Semiconductor Products, Inc.

TABLE 2 – MICROPROCESSOR INSTRUCTION SET – ALPHABETIC SEQUENCE

ABA	Add Accumulators
ADC	Add with Carry
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
BCC	Branch if Carry Clear
BCS	Branch if Carry Set
BEQ	Branch if Equal to Zero
BGE	Branch if Greater or Equal Zero
BGT	Branch if Greater than Zero
BHI	Branch if Higher
BIT	Bit Test
BLE	Branch if Less or Equal
BLS	Branch if Lower or Same
BLT	Branch if Less than Zero
BMI	Branch if Minus
BNE	Branch if Not Equal to Zero
BPL	Branch if Plus
BRA	Branch Always
BSR	Branch to Subroutine
BVC	Branch if Overflow Clear
BVS	Branch if Overflow Set
CBA	Compare Accumulators
CLC	Clear Carry
CLI	Clear Interrupt Mask
CLR	Clear
CLV	Clear Overflow
CMP	Compare
COM	Complement
CPX	Compare Index Register
DAA	Decimal Adjust
DEC	Decrement
DES	Decrement Stack Pointer
DEX	Decrement Index Register
EOR	Exclusive OR
INC	Increment
INS	Increment Stack Pointer
INX	Increment Index Register
JMP	Jump
JSR	Jump to Subroutine
LDA	Load Accumulator
LDS	Load Stack Pointer
LDX	Load Index Register
LSR	Logical Shift Right
NEG	Negate
NOP	No Operation
ORA	Inclusive OR Accumulator
PSH	Push Data
PUL	Pull Data
ROL	Rotate Left
ROR	Rotate Right
RTI	Return from Interrupt
RTS	Return from Subroutine
SBA	Subtract Accumulators
SBC	Subtract with Carry
SEC	Set Carry
SEI	Set Interrupt Mask
SEV	Set Overflow
STA	Store Accumulator
STS	Store Stack Register
STX	Store Index Register
SUB	Subtract
SWI	Software Interrupt
TAB	Transfer Accumulators
TAP	Transfer Accumulators to Condition Code Reg.
TBA	Transfer Accumulators
TPA	Transfer Condition Code Reg. to Accumulator
TST	Test
TSX	Transfer Stack Pointer to Index Register
TXS	Transfer Index Register to Stack Pointer
WAI	Wait for Interrupt

TABLE 3 — ACCUMULATOR AND MEMORY INSTRUCTIONS

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED =	DIRECT OP	DIRECT ~	DIRECT =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	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	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A — 0	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B C b7 b0	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B b7 b0 C	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A 0 —	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B b7 b0 C	•	•	R	↕	⑥	↕
Store Acmltr	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

ADDRESSING MODES

COND. CODE REG

LEGEND

OP	Operation Code (Hexadecimal)	+	Boolean Inclusive OR
~	Number of MPU Cycles	⊕	Boolean Exclusive OR
=	Number of Program Bytes	M̄	Complement of M
+	Arithmetic Plus	→	Transfer Into
−	Arithmetic Minus	0	Bit = Zero
·	Boolean AND	00	Byte = Zero
M_{SP}	Contents of memory location pointed to be Stack Pointer		

Note — Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS

H	Half-carry from bit 3
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry from bit 7
R	Reset Always
S	Set Always
↕	Test and set if true, cleared otherwise
•	Not Affected

TABLE 4 — INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED =	DIRECT OP	DIRECT ~	DIRECT =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	⑦	↕	↕	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	↕	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	↕	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	1	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	1	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

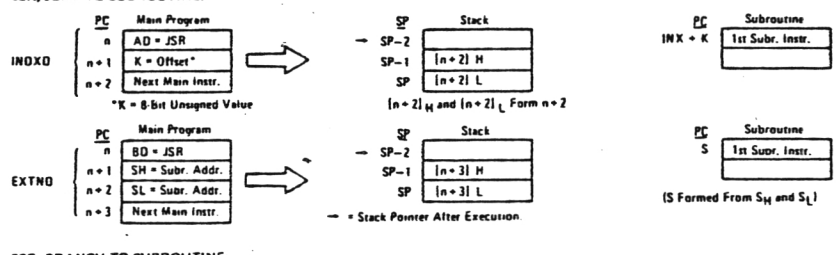
TABLE 5 — JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE =	INDEX OP	INDEX ~	INDEX =	EXTND OP	EXTND ~	EXTND =	IMPLIED OP	IMPLIED ~	IMPLIED =	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	80	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	⑪	•	•	•	•

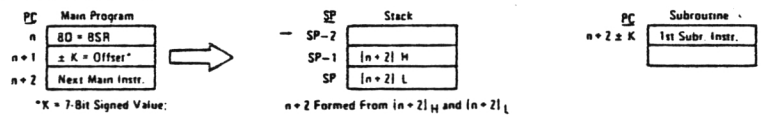
*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

SPECIAL OPERATIONS

JSR, JUMP TO SUBROUTINE:

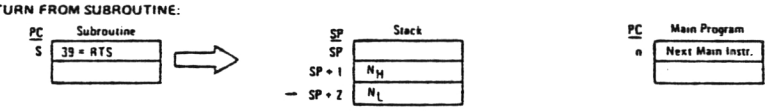


BSR, BRANCH TO SUBROUTINE:



JMP, JUMP:

RTS, RETURN FROM SUBROUTINE:



RTI, RETURN FROM INTERRUPT:

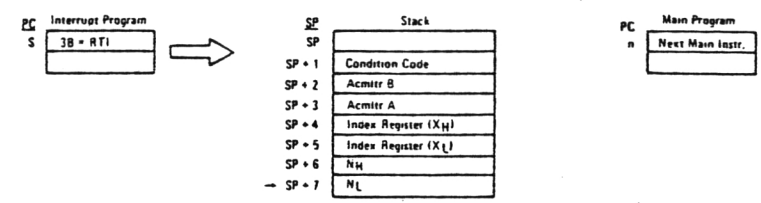


TABLE 6 – CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

		IMPLIED				COND. CODE REG.					
						5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	=	BOOLEAN OPERATION	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	•
Acmltr A → CCR	TAP	06	2	1	A → CCR	(12)					
CCR → Acmltr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES: (Bit set if test is true and cleared otherwise)

1 (Bit V) Test: Result = 10000000?
2 (Bit C) Test: Result = 00000000?
3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4 (Bit V) Test: Operand = 10000000 prior to execution?
5 (Bit V) Test: Operand = 01111111 prior to execution?
6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.
7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?
8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?
9 (Bit N) Test: Result less than zero? (Bit 15 = 1)
10 (All) Load Condition Code Register from Stack. (See Special Operations)
11 (Bit I) Set when interrupt occurs. If previously set, a Non Maskable Interrupt is required to exit the wait state.
12 (All) Set according to the contents of Accumulator A.

ภาคผนวก ค.

```
100 CLEAR500
110 PRINT CHR$(12)
120 PRINT"*****************************************"
130 PRINT"*     cross assembler for M6800         *"
140 PRINT"*****************************************":PRINT
150 PRINT"your 6800 program must have previously been keyed in
160 PRINT"and save in ASCII basic mode":PRINT
170 INPUT "1.name of program files";DA$
180 IF DA$="" THEN 170
190 LE=LEN(DA$)
200 IF LE>6 THEN PRINT" name must be less than 6 characters":GOTO170
210 OPEN DA$     FOR INPUT  AS #1
220 INPUT "2.name of object file you want saved"; TI$
230 IF TI$=""  THEN 220
240 L = LEN(TI$)
250 IF L>6 THEN PRINT"  name must be less than 6 characters":GOTO220
260 INPUT"  Do you want listing ? (y/n)";DO$
270 IF MID$(DO$,1,1)<>"y" THEN300
280 PR=1
290 GOTO 310
300 PR=0
310 OPEN "temp" FOR OUTPUT AS #3
320 OPEN"liter" FOR OUTPUT AS#3
330 GOSUB3890
340 CLOSE 3
350 OPEN"symbol" FOR OUTPUT AS#4
360 U=26624:S$="line#"
370 Q$="SYMBOL  TABLE"
380 PRINT#4,Q$ :PRINT#4,U:PRINT#4,S$
390 CLOSE4
400 T9=A=E=0:YY=0
410 LC=0
420 DIM L$(10),M$(10),O$(30),Z$(72),P$(72),T$(72)
430 DIM A$(72),N$(72),I$(72)
440 PRINT CHR$(12)
450 LOCATE30,12
460 PRINT"assembler begin"
470 FOR I=1 TO 1000:NEXT I
480 IF EOF(1) THEN 2000
490 L$="":I$="":M$="":O$="":C$=""
500 Z$="
```

```
510 Z=0:JX=0
520 '***    separate tokens  *****
530 LINEINPUT#1,I$
540 PRINT I$
550 MID$(Z$,17,LEN(I$))=I$
560 TS=LC:P=1:GOSUB2610:S$=P$:I$=MID$(I$,P+1,LEN(I$)-P)
570 IF I$="" THEN870
580 P=1:P$=";":GOSUB2760
590 IF P1=0 THEN610
600 IF P1=1 THEN870
610 IF MID$(I$,1,1)=" " THEN630
620 GOSUB2610:L$=P$
630 GOSUB2610:M$=P$
640 IF M$<>"ORG" THEN660
650 GOSUB2610:N$=P$:GOSUB2840:LC=F1:TS=LC:GOSUB3910:GOTO480
660 IF M$="END" THEN1990
670 IF L$<>"" THEN880
680 IF M$="FCB" THEN3380
690 IF M$="FDB" THEN3380
700 IF M$="FCC" THEN3380
710 IF M$="RMB" THEN1110
720 IF M$<>"LORG" THEN740
730 GOSUB3910:YY=1:GOTO5120
740 IF M$=""THEN820
750 RESTORE3930
760 FOR I=1 TO 107
770 READ T$
780 IF MID$(T$,1,LEN(T$))=MID$(M$,1,LEN(M$)) THEN840
790 NEXT I
800 GOSUB3960:PRINT#3,"                . unknown opcode"
810 GOSUB5860:GOTO5750
820 GOSUB3910:PRINT#3,"                    missing opcode"
830 GOSUB5860:GOTO5750
840 O=I:GOSUB2570:O$=P$
850 IF O$="" THEN1120
860 X=P-LEN(O$)+LEN(S$)+16:MID$(Z$,X,1)="!":GOTO1120
870 GOSUB3910:GOTO480
880 IF M$="EQU" THEN 960
890 GOSUB3930
900 IF EOF(4)THEN940
910 GOSUB3880
```

```
920 IF B$<>M$ THEN900
930 GOSUB3910:GOSUB3950 :GOTO480
940 CLOSE4:GOSUB3940
950 PRINT#4,L$:PRINT#4,T3:PRINT#4,S$:CLOSE#4:GOTO680
960 GOSUB 2610:O$=P$
970 IF O$<>"" THEN 1000
980 GOSUB3910:PRINT#3,"                    missing operand"
990 GOSUB3860:GOTO3750
1000 N$=O$:GOSUB2840
1010 IF T4 <> 2 THEN  1040
1020 GOSUB3910:PRINT#3,"                      undefind label"
1030 GOSUB3860:GOTO3750
1040 I=F1:GOSUB3930
1050 IF EOF(4) THEN 1090
1060 GOSUB3880
1070 IF B$<>M$ THEN 1050
1080 GOSUB3910:GOSUB3950:GOTO480
1090 CLOSE4:GOSUB3940:PRINT#4,L$:PRINT#4,I:PRINT#4,S$:CLOSE4
1100 I=T3:GOSUB3760:GOSUB3900:GOSUB3870:T3=LC:GOTO480
1110 GOSUB 2570:N$=P$:GOSUB2840:LC=LC+F1:GOTO1100
1120 '******  find addressing modes, load effective address  ******'
1130 IF MID$(O$,1,1)="*" THEN 1200
1140 IF O$<>"X" THEN1180
1150 M=4:A=0:GOTO1690
1160 P$="*":GOSUB3970:PX=P1
1170 IF PX<>0 THEN M=3:A=-1000:GOTO1690
1180 P$="/":GOSUB3970:PX=P1
1190 IF PX<>0 THEN M=3:A=-1000:GOTO1690
1200 IF O$<>"" THEN3620
1210 M=1:GOTO1840
1220 IF MID$(O$,1,1)<>"#" THEN1370
1230 IF M$="CPX" THEN1260
1240 IF M$="LDX" THEN1260
1250 IF M$<>"LDS" THEN1270
1260 Z=1
1270 M=2:P=P+1:N$=MID$(O$,Z,LEN(O$)-1):GOSUB2840:A=F1
1280 IF Z=1 THEN 1340
1290 IF A>255 THEN 1330
1300 IF T4<>2 THEN 1320
1310 A=-200
1320 GOTO1690
1330 GOSUB3910:GOSUB3850:GOTO3750
1340 IF T4<>2 THEN 1360
```

```
1350 A=-1000
1360 GOTO1690
1370 IF MID$(M$,1,1)<>"B" THEN 1490
1380 IF MID$(M$,1,3)="BIT" THEN 1490
1390 M=6:N$=O$:GOSUB2840
1400 IF T4=2 THEN 1470
1410 A=P1-LC-2
1420 IF A>=0 THEN 1440
1430 A=256+A
1440 IF ABS(P1-LC)<=127 THEN 1480
1450 GOSUB3910:PRINT#3,"                    branch out of range"
1460 GOSUB5860:GOTO5750
1470 A=-200
1480 GOTO 1690
1490 P=P-LEN(O$):P$=",":GOSUB2760:P6=P1:P7=0
1500 IF P6=0 THEN 1530
1510 IF MID$(I$,P6+1,1)<>"X" THEN 1610
1520 P7=1
1530 GOSUB2610:N$=P$
1540 IF P7=1 THEN 1630
1550 M=3:GOSUB2840:A=P1
1560 IF  T4<>2 THEN 1580
1570 A=-1000
1580 IF ABS(A)<=255 THEN 1600
1590 M=5
1600 GOTO 1690
1610 GOSUB3910:PRINT#3,"                    invalid addressing"
1620 GOSUB5860:GOTO5750
1630 M=4:GOSUB2840:A=P1
1640 IF T4<>2 THEN 1660
1650 A=-200:GOTO1690
1660 IF A<=255 THEN 1690
1670 GOSUB3910:GOSUB5850:GOTO5750
1680 '*****   print opcode & on file  *****'
1690 IF A>=0 THEN 1770
1700 MID$(Z$,10,2)="**":LC=LC+1
1710 IF M<>6 THEN 1730
1720 MID$(Z$,11,1)="R"
1730 W9=A+256
1740 IF W9>=0 THEN 1840
1750 MID$(Z$,13,2)="**"
1760 LC=LC+1:GOTO1840
1770 I=A:GOSUB5760
```

```
1780 A1$=A$
1790 IF M=5 OR Z=1 THEN 1820
1800 MID$(Z$,10,2)=MID$(A$,LEN(A$)-1,2)
1810 LC=LC+1:GOTO1840
1820 GOSUB5910
1830 LC=LC+2
1840 I=T5:GOSUB5760
1850 GOSUB5900
1860 RESTORE 4050
1870 FOR I=1 TO (O-1)*6+M
1880 READ T$
1890 NEXT I
1900 IF T$<> "   " THEN 1960
1910 IF M<> 3  THEN 1970
1920 READ T$:READ T$
1930 IF T$="   "  THEN 1970
1940 A$=A1$:GOSUB5910
1950 LC=LC+1
1960 MID$(Z$,7,2)=T$:LC=LC+1:GOSUB5870:GOTO480
1970 GOSUB5910:PRINT#3,"                    illegal addressing"
1980 GOSUB5860:GOTO5750
1990 GOSUB 5910
2000 '******  second pass: resolve fwd reference  ******'
2010 GOTO 5110
2020 PRINT "*********  second  ************"
2030 OPEN  TI$  FOR OUTPUT AS #2
2040 OPEN "temp" FOR INPUT AS #3
2050 IF EOF(3) THEN 2370
2060 LINEINPUT#3,I$:INPUT#3 ,T5:LINEINPUT#3,S$
2070 LPRINT" i$=";I$
2080 P=1
2090 IF I$="" THEN 2360
2100 P$="!":GOSUB2760
2110 IF P1<17  THEN 2360
2120 P=P1:MID$(I$,P,1)=" "
2130 IF MID$(I$,10,1)<>"*" THEN 2360
2140 GOSUB2570:N$=P$
2150 IF MID$(N$,1,1)<>"=" THEN 2170
2160 A$=N$:JX=1:FZ=0:F1=0:GOSUB5010:F1=FZ:GOTO2220
2170 O$=N$:P$="*":GOSUB5970:PX=P1
2180 IF PX<>0THEN 6030
2190 P$="/":GOSUB5970:PX=P1
2200 IF PX<>0 THEN6030
```

```
2210 GOSUB 2840
2220 IF T4<>2 THEN 2260
2230 PRINT#2,I$
2240 PRINT#2,"                    irresolve fwd ref. bad label"
2250 E=E+1:GOTO 2050
2260 I=F1
2270 IF MID$(I$,11,1)<>"R" THEN 2310
2280 I=F1-T5-2
2290 IF I>=0 THEN 2310
2300 I=I+256
2310 GOSUB5760
2320 IF MID$(I$,13,2)<>"**" THEN 2350
2330 MID$(I$,10,2)=MID$(A$,LEN(A$)-3,2)
2340 MID$(I$,13,2)=MID$(A$,LEN(A$)-1,2):GOTO2360
2350 MID$(I$,10,2)=MID$(A$,LEN(A$)-1,2)
2360 PRINT#2,I$:GOTO2050
2370 CLOSE2,3
2380 OPEN  TI$  FOR INPUT AS #2
2390 IF EOF(2) THEN 2440
2400 LINEINPUT#2,I$
2410 IF PR=0 THEN  2430
2420 LPRINT I$:GOTO 2390
2430 PRINT I$:GOTO2390
2440 CLOSE2
2450 GOSUB5930
2460 OPEN TI$ FOR APPEND AS#2
2470 IF EOF(4) THEN 2540
2480 GOSUB5880:I=K:GOSUB5760
2490 A$=MID$(A$,LEN(A$)-3,4)
2500 PRINT#2,B$ TAB(15) A$ TAB(25)SS$
2510 IF PR=0 THEN 2530
2520 LPRINT B$ TAB(15) A$ TAB(25)SS$:GOTO 2470
2530 PRINT B$ TAB(15) A$ TAB(25)SS$:GOTO2470
2540 CLOSE2:IF PR=0 THEN 2560
2550 LPRINT"TOTAL ERROR ";E:CLOSE4 :END
2560 PRINT"TOTLE ERROR ";E:CLOSE4:END
2570 '****** routine to isolate token  **********
2580 'start looking for token at p, puts it in p$,and
2590 'update p. if entered here, stop scan at ' '
2600 T9=1
2610 'if enter here, stop scan at ':',','
2620 FOR I1=P TO LEN(I$)
2630 IF MID$(I$,I1,1)<>" "THENI2650
```

```
2640 NEXT Ii
2650 P$=""
2660 FORIZ=Ii TO LEN(I$)
2670 IF MID$(I$,IZ,i)=" "THENZ720
2680 IF T9=i THENZ700
2690 IF MID$(I$,IZ,i)=","THENZ720
2700 P$=P$+MID$(I$,IZ,i)
2710 NEXT IZ
2720 P=IZ
2730 IF LEN(P$)<>0THENZ750
2740 P=P+i
2750 T9=0:RETURN
2760 '*******   find symbol routine   **********
2770 'return pi=symloc  if it found,
2780 'pi=0  if symbol not found
2790 FOR I=P TO LEN(I$)
2800 IF MID$(I$,I,i)=P$THENZ830
2810 NEXT I
2820 Pi=0:RETURN
2830 Pi=I:RETURN
2840 '*********   numeric string interpreter  *********
2850 'simplifiles string of label and numeric expressions
2860 'of number in any base, plus ascii constants.
2870 Fi=W=0=FZ:A$=""
2880 FOR I=i TO LEN(N$)
2890 IF MID$(N$,I,i)="+"THENZ950
2900 IF MID$(N$,I,i)="-"THENZ950
2910 IF MID$(N$,I,i)=","THENZ950
2920 A$=A$+MID$(N$,I,i)
2930 NEXT I
2940 IF A$="X" THEN RETURN
2950 IF A$<>"*" THENZ970
2960 FZ=LC:GOTO3240
2970 IF MID$(A$,i,i)<>"#" THENZ990
2980 A$=MID$(A$,Z,LEN(A$)-i)
2990 IF MID$(A$,i,i)>"Z"THEN3130
3000 IF MID$(A$,i,i)<"A"THEN3130
3010 GOSUB5930
3020 OPEN"ssss" FOR OUTPUT AS#5
3030 IF EOF(4)THEN3120
3040 T$="":Ti=0
3050 LINEINPUT#4,T$:INPUT#4,Ti:LINEINPUT#4,S5$
3060 IF T$ <> A$  THEN PRINT#5,T$:PRINT#5,Ti:PRINT#5,S5$:GOTO3030
```

```
3070 FZ=T1:T4=3:SS$=SS$+"      "+S$
3080 PRINT#3,T$:PRINT#3,T1:PRINT#3,SS$;:GOSUB3770
3090 IF JX=0 THEN 3110
3100 JX=0:RETURN
3110 GOTO 3240
3120 T4=2:GOSUB3770:RETURN
3130 IF MID$(A$,1,1)<>"'" THEN3150
3140 A$=MID$(A$,2,LEN(A$)-1):FZ=ASC(A$):GOTO3240
3150 B=10
3160 IF MID$(A$,1,1)<>"%" THEN3180
3170 B=2:GOTO3220
3180 IF MID$(A$,1,1)<>"$" THEN3200
3190 B=16:GOTO3220
3200 IF MID$(A$,1,1)<>"@" THEN3230
3210 B=8
3220 A$=MID$(A$,2,LEN(A$)-1):GOSUB3370:FZ=F:GOTO3240
3230 FZ=VAL(A$)+.03
3240 IF W=2 THEN 3260
3250 F1=F1+FZ:GOTO3270
3260 F1=F1-FZ
3270 IF I>=LEN(N$) THEN 3360
3280 T$="+-,"
3290 FOR W=1 TO LEN(T$)
3300 IF MID$(T$,W,1)=MID$(N$,I,1) THEN  3330
3310 NEXT W
3320 GOSUB3910
3330 PRINT#3,"                illegal operator in line"
3340 GOSUB3860:GOTO3730
3350 A$="":GOTO2930
3360 T4=0:RETURN
3370 '****  multi-radix string to number converter  ****'
3380 'b is base of number in a$, f is product'
3390 F=0:I1=0
3400 FOR I2=LEN(A$) TO 1 STEP-1
3410 RESTORE 3350
3420 FOR N=0 TO B-1
3430 READ F$
3440 IF F$=MID$(A$,I2,1) THEN 3490
3450 NEXT N
3460 GOSUB3910
3470 PRINT#3,"                              invalid number  ":GOSUB3860
3480 GOTO3730
3490 F=F+N*(B^I1)
```

```
3500 Ii=Ii+i
3510 NEXT IZ
3520 IF Ii <4 THEN 3540
3530 F=F+.i
3540 RETURN
3550 DATA"0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F
3560 DATA"G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V
3570 DATA"W","X","Y","Z"
3580 '******  process memory load  *******
3590 Q7=i
3600 IF M$ <> "FCC"  THEN  3700
3610 IF Q7 <> i      THEN  3630
3620 GOSUB 2570:P=P-LEN(P$):O$=MID$(I$,P,i):P=P+i
3630 IF P <= 72  THEN  3660
3640 GOSUB3910
3650 PRINT#3,"                bad delimiter" :GOSUB5860:GOTO5750
3660 MID$(P$,i,i)="'"
3670 MID$(P$,2,i)=MID$(I$,P,i)
3680 IF  MID$(P$,2,i) = O$  THEN  480
3690 GOTO 3710
3700 GOSUB 2610
3710 P=P+i
3720 IF LEN(P$) = 0  THEN  480
3730 N$=P$:GOSUB2840
3740 IF T4<>2 THEN 3780
3750 GOSUB3910
3760 PRINT#3,"                invalid label in memory assignment"
3770 GOSUB5860:GOTO5750
3780 I=Fi:GOSUB5760
3790 IF M$<>"FDB" THEN 3830
3800 MID$(Z$,7,2)=MID$(A$,LEN(A$)-3,2)
3810 MID$(Z$,10,2)=MID$(A$,LEN(A$)-i,2)
3820 LC=LC+2:GOTO3860
3830 IF Fi<256 THEN 3850
3840 GOSUB3910:GOSUB5850:GOTO5750
3850 MID$(Z$,7,2)=MID$(A$,LEN(A$)-i,2):LC=LC+i
3860 I=T5:GOSUB5760:GOSUB5900
3870 IF Q7=i THEN 3890
3880 Z$=MID$(Z$,i,15)
3890 Q7=0
3900 GOSUB3910:T5=LC:GOTO3600
3910 I=T5:GOSUB5760:GOSUB5900:GOSUB5870:RETURN
```

```
3920 REM ****** mnemonic table **************
3930 DATA"ADDA","ADDB","ABA","ADCA","ADCB","ANDA","ANDB","BITA","BITB"
3940 DATA"CLR","CLRA","CLAB","CMPA","CMPB","CBA","COM","COMA","COMB"
3950 DATA"NEG","NEGA","NAGB","DAA","DEC","DECA","DECB","EORA","EORB"
3960 DATA"INC","INCA","INCB","LDAA","LDAB","ORAA","ORAB","PSHA","PSHB"
3970 DATA"PULA","PULB","ROL","ROLA","ROLB","ROR","RORA","RORB","ASL"
3980 DATA"ASLA","ASLB","ASR","ASRA","ASRB","LSR","LSRA","LSRB","STAA"
3990 DATA"STAB","SUBA","SUBB","SBA","SBCA","SBAB","TAB","TBA","TST"
4000 DATA"TSTA","TSTB","CPX","DEX","DES","INX","INS","LDX","LDS"
4010 DATA"STX","STS","TXS","TSX","BRA","BCC","BCS","BEQ","BGE"
4020 DATA"BGT","BHI","BLE","BLS","BLT","BMI","BNE","BVC","BVS"
4030 DATA"BPL","BSR","JMP","JSR","NOP","RTI","RTS","SWI","WAI"
4040 DATA"CLC","CLI","CLV","SEC","SEI","SEV","TAP","TPA"
4050 DATA"  ","8B","9B","AB","BB","  "
4060 DATA"  ","CB","DB","EB","FB","  "
4070 DATA"1B","  ","  ","  ","  ","  "
4080 DATA"  ","89","99","A9","B9","  "
4090 DATA"  ","C9","D9","E9","F9","  "
4100 DATA"  ","84","94","A4","B4","  "
4110 DATA"  ","C4","D4","E4","F4","  "
4120 DATA"  ","85","95","A5","B5","  "
4130 DATA"  ","C5","D5","E5","F5","  "
4140 DATA"  ","  ","  ","6F","7F","  "
4150 DATA"4F","  ","  ","  ","  ","  "
4160 DATA"5F","  ","  ","  ","  ","  "
4170 DATA"  ","81","91","A1","B1","  "
4180 DATA"  ","C1","D1","E1","F1","  "
4190 DATA"11","  ","  ","  ","  ","  "
4200 DATA"  ","  ","  ","63","73","  "
4210 DATA"43","  ","  ","  ","  ","  "
4220 DATA"53","  ","  ","  ","  ","  "
4230 DATA"  ","  ","  ","60","70","  "
4240 DATA"40","  ","  ","  ","  ","  "
4250 DATA"50","  ","  ","  ","  ","  "
4260 DATA"19","  ","  ","  ","  ","  "
4270 DATA"  ","  ","  ","6A","7A","  "
4280 DATA"4A","  ","  ","  ","  ","  "
4290 DATA"5A","  ","  ","  ","  ","  "
4300 DATA"  ","88","98","A8","B8","  "
4310 DATA"  ","C8","D8","E8","F8","  "
4320 DATA"  ","  ","  ","6C","7C","  "
4330 DATA"4C","  ","  ","  ","  ","  "
4340 DATA"5C","  ","  ","  ","  ","  "
4350 DATA"  ","86","96","A6","B6","  "
```

```
4360 DATA"  ","C6","D6","E6","F6","
4370 DATA"  ","8A","9A","AA","BA","
4380 DATA"  ","CA","DA","EA","FA","
4390 DATA"36","  ","  ","  ","  ","
4400 DATA"37","  ","  ","  ","  ","
4410 DATA"32","  ","  ","  ","  ","
4420 DATA"33","  ","  ","  ","  ","
4430 DATA"  ","  ","  ","69","79","
4440 DATA"49","  ","  ","  ","  ","
4450 DATA"59","  ","  ","  ","  ","
4460 DATA"  ","  ","  ","66","76","
4470 DATA"46","  ","  ","  ","  ","
4480 DATA"56","  ","  ","  ","  ","
4490 DATA"  ","  ","  ","68","78","
4500 DATA"48","  ","  ","  ","  ","
4510 DATA"58","  ","  ","  ","  ","
4520 DATA"  ","  ","  ","67","77","
4530 DATA"47","  ","  ","  ","  ","
4540 DATA"57","  ","  ","  ","  ","
4550 DATA"  ","  ","  ","64","74","
4560 DATA"44","  ","  ","  ","  ","
4570 DATA"54","  ","  ","  ","  ","
4580 DATA"  ","  ","97","A7","B7","
4590 DATA"  ","  ","D7","E7","F7","
4600 DATA"  ","80","90","A0","B0","
4610 DATA"  ","C0","D0","E0","F0","
4620 DATA"10","  ","  ","  ","  ","
4630 DATA"  ","82","92","A2","B2","
4640 DATA"  ","C2","D2","E2","F2","
4650 DATA"16","  ","  ","  ","  ","
4660 DATA"17","  ","  ","  ","  ","
4670 DATA"  ","  ","  ","6D","7D","
4680 DATA"4D","  ","  ","  ","  ","
4690 DATA"5D","  ","  ","  ","  ","
4700 DATA"  ","8C","9C","AC","BC","
4710 DATA"09","  ","  ","  ","  ","
4720 DATA"34","  ","  ","  ","  ","
4730 DATA"08","  ","  ","  ","  ","
4740 DATA"31","  ","  ","  ","  ","
4750 DATA"  ","CE","DE","EE","FE","
4760 DATA"  ","8E","9E","AE","BE","
4770 DATA"  ","  ","DF","EF","FF","
4780 DATA"  ","  ","9F","AF","BF","
```

```
4790 DATA"35","  ","  ","  ","  ","  "
4800 DATA"30","  ","  ","  ","  ","  "
4810 DATA"  ","  ","  ","  ","  ","20"
4820 DATA"  ","  ","  ","  ","  ","24"
4830 DATA"  ","  ","  ","  ","  ","25"
4840 DATA"  ","  ","  ","  ","  ","27"
4850 DATA"  ","  ","  ","  ","  ","2C"
4860 DATA"  ","  ","  ","  ","  ","2E"
4870 DATA"  ","  ","  ","  ","  ","22"
4880 DATA"  ","  ","  ","  ","  ","2F"
4890 DATA"  ","  ","  ","  ","  ","23"
4900 DATA"  ","  ","  ","  ","  ","2D"
4910 DATA"  ","  ","  ","  ","  ","28"
4920 DATA"  ","  ","  ","  ","  ","26"
4930 DATA"  ","  ","  ","  ","  ","29"
4940 DATA"  ","  ","  ","  ","  ","2A"
4950 DATA"  ","  ","  ","  ","  ","8D"
4960 DATA"  ","  ","  ","6E","7E","  "
4970 DATA"  ","  ","  ","AD","BD","  "
4980 DATA"  ","  ","  ","  ","  ","  "
4990 DATA"01","  ","  ","  ","  ","  "
5000 DATA"3B","  ","  ","  ","  ","  "
5010 DATA"39","  ","  ","  ","  ","  "
5020 DATA"3F","  ","  ","  ","  ","  "
5030 DATA"3E","  ","  ","  ","  ","  "
5040 DATA"0C","  ","  ","  ","  ","  "
5050 DATA"0E","  ","  ","  ","  ","  "
5060 DATA"0A","  ","  ","  ","  ","  "
5070 DATA"0D","  ","  ","  ","  ","  "
5080 DATA"0F","  ","  ","  ","  ","  "
5090 DATA"0B","  ","  ","  ","  ","  "
5100 DATA"06","  ","  ","  ","  ","  "
5110 CLOSE1
5120 OPEN "liter" FOR INPUT AS#3
5130 LINEINPUT#3,TE$:LINEINPUT#3,S5$
5140 IF EOF(3) THEN   5550
5150 LINEINPUT#3,TE$:LINEINPUT#3,S5$
5160 GOSUB5940
5170 PRINT#4,TE$:PRINT#4,T5:PRINT#4,S$ :CLOSE4
5180 I$="
5190 MID$(I$,24,LEN(TE$))=TE$:Q7=1:F=26
5200 IF MID$(I$,24,2)<> "=C"  THEN  5310
5210 IF Q7<> 1  THEN  5230
```

```
5220 P$=MID$(TE$,3,LEN(TE$)-2):O$=MID$(I$,P,1):P=P+1
5230 IF P<=72 THEN  5270
5240 Z$=I$:GOSUB3910
5250 PRINT#3,"                          invalid delimiter"
5260 GOSUB5860:E=E+1:GOTO5140
5270 MID$(P$,1,1)=""
5280 MID$(P$,2,1)=MID$(I$,P,1)
5290 IF  MID$(P$,2,1) =O$  THEN 5140
5300 GOTO  5320
5310 GOSUB 2610
5320 P=P+1
5330 IF LEN(P$)=0 THEN 5140
5340 N$=P$:GOSUB2840
5350 IF T4<>2 THEN  5390
5360 Z$=I$:GOSUB3910
5370 PRINT#3,"                          invalid litaral"
5380 GOSUB5860:E=E+1:GOTO5140
5390 I=F1:GOSUB5760
5400 IF MID$(TE$,1,2)<>"=D"  THEN   5440
5410 MID$(I$,7,2)=MID$(A$,LEN(A$)-3,2)
5420 MID$(I$,10,2)=MID$(A$,LEN(A$)-1,2)
5430 LC=LC+2:GOTO5480
5440 IF F1<256  THEN 5460
5450 Z$=I$:GOSUB3910:GOSUB5850:E=E+1:GOTO5140
5460 MID$(I$,7,2)=MID$(A$,LEN(A$)-1,2)
5470 LC=LC+1
5480 I=T5:GOSUB5760
5490 MID$(I$,1,4)=MID$(A$,LEN(A$)-3,4)
5500 MID$(I$,5,1)=":"
5510 Z$=I$
5520 IF Q7=1 THEN  5540
5530 Z$=MID$(I$,1,15)
5540 Q7=0:GOSUB3910:T5=LC:GOTO5200
5550 IF YY<>1 THEN 5600
5560 YY=0:CLOSE5
5570 OPEN"liter" FOR OUTPUT AS#5
5580 GOSUB5890
5590 CLOSE5 :GOTO480
5600 CLOSE3,5
5610 GOTO 2020
5620 IF MID$(O$,1,1)<>"=" THEN 1220
5630 M=5
5640 OPEN"liter" FOR INPUT AS#5
```

```
5650 IF EOF(5) THEN 5690
5660 INPUT#5,LP$:LINEINPUT#5,ZZ$
5670 IF LP$<>O$ THEN 5650
5680 CLOSE5:GOTO 5730
5690 CLOSE5
5700 OPEN"liter" FOR APPEND AS #5
5710 PRINT#5,O$:PRINT#5,S$
5720 CLOSE5
5730 A=-1000
5740 GOTO 1690
5750 E=E+1:GOTO480
5760 T$=HEX$(I):A$="000":A$=A$+T$:RETURN
5770 IF EOF(4) THEN 5810
5780 LINEINPUT#4,T$:INPUT#4,TI:LINEINPUT#4,SS$
5790 PRINT#5,T$:PRINT#5,TI:PRINT#5,SS$
5800 GOTO 5770
5810 CLOSE4,5
5820 KILL"symbol"
5830 NAME"ssss"AS"symbol"
5840 RETURN
5850 PRINT#3,"                    value too large"
5860 PRINT#3,T5:PRINT#3,S$:RETURN
5870 PRINT#3,Z$:PRINT#3,T5:PRINT#3,S$:RETURN
5880 LINEINPUT#4,B$:INPUT#4,K:LINEINPUT#4,SS$:RETURN
5890 PRINT#5,"&Z$#":PRINT#5,"zzzz":RETURN
5900 MID$(Z$,1,4)=MID$(A$,LEN(A$)-3,4):MID$(Z$,5,1)=":":RETURN
5910 MID$(Z$,10,2)=MID$(A$,LEN(A$)-3,2)
5920 MID$(Z$,13,2)=MID$(A$,LEN(A$)-1,2):RETURN
5930 OPEN"symbol"FOR INPUT AS#4:RETURN
5940 OPEN"symbol"FOR APPEND AS#4:RETURN
5950 PRINT#3,"                    multiple definition"
5960 GOSUB5860:E=E+1:CLOSE4:RETURN
5970 P1=0
5980 FOR I=1 TO LEN(O$)
5990 IF MID$(O$,I,1)=P$ THEN 6020
6000 NEXTI
6010 P1=0:RETURN
6020 P1=I:RETURN
6030 I=1:P=1:GOSUB 6140:A$=P$:L=ICP
6040 PO$=A$+",":OP$=MID$(N$,J,1):ISP(1)=L
6050 P=P+1
6060 GOSUB 6140:A$=P$:L=ICP:LL$=MID$(N$,J,1)
6070 PO$=PO$+A$+","
```

```
6080 IF ISP(I)>=L THEN 6100
6090 OP$=OP$+LL$:I=I+1:ISP(I)=L:GOTO6050
6100 PO$=PO$+MID$(OP$,LEN(OP$),1)+","
6110 IF LEN(OP$)>1 THEN I=I-1:OP$=MID$(OP$,1,LEN(OP$)-1):GOTO6080
6120 IF L=0 THEN6230
6130 I=1:OP$=LL$:ISP(I)=L:GOTO6050
6140 P$=""
6150 FOR J=P TO LEN(N$)
6160 IF MID$(N$,J,1)="+" THEN 6230
6170 IF MID$(N$,J,1)="-" THEN 6230
6180 IF MID$(N$,J,1)="*" THEN 6240
6190 IF MID$(N$,J,1)="/" THEN 6240
6200 P$=P$+MID$(N$,J,1)
6210 NEXT J
6220 P=J:ICP=0:RETURN
6230 P=J:ICP=1:RETURN
6240 P=J:ICP=2:RETURN
6250 P=1:KK=0:M$=PO$
6260 GOSUB 6380
6270 N$=P$
6280 IF N$=""THEN  F1=J(1):GOTO2260
6290 IF N$="+" THEN X=J(KK-1)+J(KK):GOTO 6370
6300 IF N$="-" THEN X=J(KK-1)-J(KK):GOTO 6370
6310 IF N$="*" THEN X=J(KK-1)*J(KK):GOTO 6370
6320 IF N$="/" THEN X=J(KK-1)/J(KK):GOTO 6370
6330 GOSUB2840
6340 IF T4<>2 THEN 6360
6350 GOTO2230
6360 KK=KK+1:J(KK)=F1:P=P+1:GOTO6260
6370 KK=KK-1:J(KK)=X:P=P+1:GOTO6260
6380 P$=""
6390 FOR J=P TO LEN(M$)
6400     IF MID$(M$,J,1)=","THEN6440
6410     P$=P$+MID$(M$,J,1)
6420 NEXT J
6430 RETURN
6440 P=J:RETURN
```

ภาคผนวก ง.

```
100 PRINT CHR$(12)
110 PRINT"*************************************************************************
120 PRINT"*                                                                       *
130 PRINT"*    PROGRAM FOR GET LISTING FROM ASSEMBLER ASSM6800                    *
140 PRINT"*                                                                       *
150 PRINT"*************************************************************************
160 PRINT
170 INPUT" What is your object file name ";OJ$
180 PRINT" Do you want listing or memory dump ?"
190 INPUT" Entry 1 for listing,2 for memory dump";W$
200 IF W$<>"1" THEN 280
210 OPEN OJ$ FOR INPUT AS#1
220 IF EOF(1) THEN 260
230 LINEINPUT#1,I$
240 LPRINT I$
250 GOTO 220
260 CLOSE1
270 END
280 OPEN OJ$ FOR INPUT AS#1
290 LINEINPUT#1,I$
300 Q$=MID$(I$,1,4)
310 A$=Q$
320 GOSUB 760
330 A=F
340 LC=1
350 Q$=Q$+":  ":GOTO 500
360 IF EOF(1) THEN 630
370 LINEINPUT#1,I$
380 IF MID$(I$,5,1)<>":" THEN 630
390 K$=MID$(I$,1,4)
400 A$=K$
410 GOSUB 760
420  B=F
430 JJ=A+LC
440 X=B-JJ
450 IF X<=0 THEN 500
460 Q$=Q$+"   ":LC=LC+1
470 X=X-1
480 IF LC>16 THEN GOSUB 650
490 IF X>=0THEN 460
500 IF MID$(I$,7,2)="  " THEN 360
```

```
510 J$=MID$(I$,7,2)
520 IF LC>16 THEN  GOSUB 650
530 Q$=Q$+J$+" ":LC=LC+1
540 IF MID$(I$,10,2)="  " THEN 560
550    J$=MID$(I$,10,2)
560    IF LC>16 THEN GOSUB 650
570     Q$=Q$+J$+" ":LC=LC+1
580 IF MID$(I$,13,2)="  " THEN 560
590    J$=MID$(I$,13,2)
600    IF LC>16 THEN  GOSUB 650
610    Q$=Q$+J$+" ":LC=LC+1
620 GOTO 560
630 LPRINT Q$:CLOSE1:END
640 '********  PRINT ROUTINE **************
650 LPRINT Q$
660 Q$=""
670 A=A+16
680 T$="000"
690 B$=HEX$(A)
700 T$=T$+B$
710 Q$=MID$(T$,LEN(T$)-3,4)
720 Q$=Q$+": "
730 LC=1
740 RETURN
750 '********   CONVERT TO DECIMAL  **************
760 F=0:I1=0
770 FOR I2=LEN(A$) TO 1 STEP -1
780 RESTORE 870
790 FOR N=0 TO 15
800 READ F$
810 IF F$= MID$(A$,I2,1) THEN 830
820 NEXT N
830 F=F+N*(16^I1)
840 I1=I1+1
850 NEXT I2
860 RETURN
870 DATA"0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F
```

ประวัติการศึกษา

นาย วิเชียร เปรมชัยสวัสดิ์ เกิดเมื่อวันที่ 2 พฤศจิกายน พ.ศ. 2499 สำเร็จการศึกษาปริญญาบัณฑิต ในสาขาวิศวกรรมไฟฟ้า จากคณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น เมื่อ พ.ศ. 2521 เข้าศึกษาต่อในภาควิชาวิศวกรรมคอมพิวเตอร์ในปีการศึกษา 2524 ปัจจุบันรับราชการในตำแหน่งอาจารย์ ประจำภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น.