BBP

TITLE:

CR80MX INSTRUCTION SET
REFERENCE MANUAL

DOCUMENT NO : CSS/006/RFM/0010

PREPARED BY : Lars Otto Kjær Nielsen

AUTHORIZED BY: Lars Otto Kjær Nielsen

DISTRIBUTION :

APPROVED BY:

AUTHORITY	DATE	SIGNATURE
Dept. Mngr.	831005	NMJ

ISSUE: 1

DATE: 831004

400-810-1

PAGE ISSUE RECORD AND CHANGE LOG.

PAGE	ISSUE 1	2	3	4	5	6	7	8
01								
02								
03								
04								
05								
06								
07								
08								
09								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
32								
33								

PAGE	ISSUE 1	2	3	4	5	6	7	8
34								
35								
36								
37								
38								
39								
40								
41								
42								
43								
44								
45								
46								
47								
48								
49								
50								
51								
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								
62								
63								
64								
65								
66								

PAGE	ISSUE 1	2	3	4	5	6	7	8
67								
68								
69								
70								
71								
72								
73								
74								
75								
76								
77								
78								
79								
80								
81								
82								
83								
84								
85								
86								
87								
88								
89								
90								
91								
92								
93								
94								
95								
96								
97								
98								
99								
100								

ISSUE	DATE	PREPARED BY	APPROVED BY	AUTHORIZED BY
1	831004	LKN	NMJ	LKN

A

CR80MX STANDARD INSTRUCTION SET SUMMARY

Sorted after

MNEMONICS

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
ADD	R3		R3	0 R3 0 R3 1000 1010	008A	12345678	46
ADD	R3	M	B6.X2	X2 B6 0000 1 R3	0008	12345678	46
ADD	R3	M	B0.X3	0 R3 1 X3 1000 1010	088A	12345678	46
ADD	M B6.X2		R3	X2 B6 0011 0 R3	0030	12345678	46
ADD	X3		R3	1 X3 0 R3 1000 1010	808A	12345678	46
ADD	X3	M	B0.X3	1 X3 1 X3 1000 1010	888A	12345678	46
ADDC	M C8		R3	C8 1100 1 R3	00C8	12345678	45
ADDC	M CN8		R3	CN8 1010 1 R3	00A8	12345678	45
ADDU	R3		R3	0 R3 0 R3 1101 0100	00D4	12345678	46
ADDU	R3	M	B0.X3	0 R3 1 X3 1101 0100	08D4	12345678	46
ADDU	X3		R3	1 X3 0 R3 1101 0100	80D4	12345678	46
ADDU	X3	M	B0.X3	1 X3 1 X3 1101 0100	88D4	12345678	46
ALT				1011 0000 1011 1100	B0BC	--345678	87
AND	R3		R3	0 R3 0 R3 1000 1100	008C	12345678	50
AND	R3	M	B0.X3	0 R3 1 X3 1000 1100	088C	12345678	50
AND	X3		R3	1 X3 0 R3 1000 1100	808C	12345678	50
AND	X3	M	B0.X3	1 X3 1 X3 1000 1100	888C	12345678	50
CAD				1000 0001 1011 1110	81BE	---45678	85
CAE				1000 0000 1011 1110	80BE	---45678	85
CIO	X3	(M)	R3	1 R3 1 X3 1101 0110	88D6	------78	70
CIO	R3	(M)	R3	1 R3 1 R3 1001 0110	8896	12345678	70
CLR		M	B6.X2	X2 B6 0010 0111	0027	12345678	51
CLRS	R3		R3	0 R3 0 R3 1000 1000	0088	12345678	66
CLRS	R3	M	B0.X3	0 R3 1 X3 1000 1000	0888	12345678	66
CLRS	X3		R3	1 X3 0 R3 1000 1001	8089	12345678	66
CLRS	X3	M	B0.X3	1 X3 1 X3 1000 1001	8889	12345678	66
CPU	R3			0110 0 R3 1011 1110	60BE	----5-78	82
CPU	M B0.X3			0110 1 X3 1011 1110	68BE	----5-78	82
DDCP	R3			0011 0 R3 1011 1100	30BC	12345678	53
DDCP	M B0.X3			0011 1 X3 1011 1100	38BC	12345678	53
DEC		M	B6.X2	X2 B6 0001 0011	0013	12345678	51
DECD	R3		R3	0 R3 1 R3 1000 1110	088E	12345678	52
DECD	X3	M	B0.X3	1 X3 1 X3 1000 1111	888F	12345678	52
DICP	R3			0010 0 R3 1011 1100	20BC	12345678	53
DICP	M B0.X3			0010 1 X3 1011 1100	28BC	12345678	53
DIV	R3		R3	0 R3 0 R3 1110 1011	00EB	12345678	49
DIV	X3		R3	1 X3 0 R3 1110 1011	80EB	12345678	49
INC		M	B6.X2	X2 B6 0001 0110	0016	12345678	51
INCD	R3		R3	0 R3 0 R3 1000 1110	008E	12345678	52
INCD	X3	M	B0.X3	1 X3 0 X3 1000 1111	808F	12345678	52
INV			R3	1001 0 R3 1011 1100	90BC	12345678	51
INV		M	B0.X3	1001 1 X3 1011 1100	98BC	12345678	51
IOR	R3		R3	0 R3 0 R3 1000 1101	008D	12345678	50
IOR	R3	M	B0.X3	0 R3 1 X3 1000 1101	088D	12345678	50
IOR	X3		R3	1 X3 0 R3 1000 1101	808D	12345678	50
IOR	X3	M	B0.X3	1 X3 1 X3 1000 1101	888D	12345678	50
JMP		M	L10	L8 1101 10L2	00D8	12345678	36
JMP		M	L8	L8 1101 1100	00DC	12345678	36
JMP		M	LN8	LN8 0101 1100	005C	12345678	36
JMP		M	LN10	LN8 0101 10L2	0058	12345678	36
JMP		M	P6.X2	X2 P6 1011 1111	00BF	12345678	37
JMP	S2	M	L8	L8 1101 11S2	00DC	12345678	36
JMP	S2	M	LN8	LN8 0101 11S2	005C	12345678	36
JMPI		M	B8	B8 1110 0111	00E7	12345678	38
JMPI		M	P8	P8 1111 1100	00FC	12345678	38
JMPI		M	B6.X2	X2 B6 0001 0010	0012	12345678	38
JMPI	S2	M	P8	P8 1111 11S2	00FC	12345678	38
JON	R3		L4	1 R3 L4 1111 0000	80F0	12345678	41
JON	R3		LN4	1 R3 LN4 1011 0000	80B0	12345678	41
JON	X3		L4	1 X3 L4 1111 0001	80F1	12345678	41
JON	X3		LN4	1 X3 LN4 1011 0001	80B1	12345678	41
JOZ	R3		L4	1 R3 L4 1111 0010	80F2	12345678	41
JOZ	R3		LN4	1 R3 LN4 1011 0010	80B2	12345678	41
JOZ	X3		L4	1 X3 L4 1111 0011	80F3	12345678	41
JOZ	X3		LN4	1 X3 LN4 1011 0011	80B3	12345678	41
JPZI		M	P8	P8 0001 0111	0017	-----678	39
JVN		M	L4	1111 L4 1011 1110	F0BE	12345678	42
JVN		M	LN4	0111 LN4 1011 1110	70BE	12345678	42
LBR				1011 0001 1011 1100	B1BC	-----678	74
LDL			R3	1000 0 R3 1011 1101	80BD	-----678	75
LDL		M	B0.X3	1000 1 X3 1011 1101	88BD	-----678	75
LDM			C4	1101 C4 1011 1110	D0BE	12345678	72
LDN			R3	0100 0 R3 1011 1100	40BC	1234-678	77
LDN		M	B0.X3	0100 1 X3 1011 1100	48BC	1234-678	77

Mnemonic OP1 OP2 OP3 Binary Hexa CPU-type Page

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
LDP			R3	0000 0 R3 1011 1100	00BC	1234-678	77
LDP		M	B0.X3	0000 1 X3 1011 1100	08BC	1234-678	77
LDS			R3	1110 0 R3 1011 1100	E0BC	12345678	72
LDS		M	B0.X3	1110 1 X3 1011 1100	E8BC	12345678	72
LDT			R3	0101 0 R3 1011 1100	50BC	12345678	72
LDT		M	B0.X3	0101 1 X3 1011 1100	58BC	12345678	72
MMP	M P0.X3		R3	1 X3 0 R3 0001 0001	8011	-----678	29
MOD	M B8			B8 1001 0100	0094	12345678	22
MOD	M P8			P8 1001 0101	0095	12345678	22
MOD	M R3			0111 0 R3 1011 1101	70BD	12345678	22
MOD	M B6.X2			X2 B6 0101 0100	0054	12345678	22
MOD	M X3			0111 1 X3 1011 1101	78BD	12345678	22
MOD4	M C8			C8 1110 0101	00E5	12345678	21
MOD4	M CN8			CN8 1010 0101	00A5	12345678	21
MOD8	M C8			C8 0101 0110	0056	12345678	21
MODC	M C8			C8 1110 0100	00E4	12345678	20
MODC	M CN8			CN8 1010 0100	00A4	12345678	20
MODN	M B8			B8 0001 0100	0014	12345678	22
MODN	M P8			P8 0001 0101	0015	12345678	22
MODN	M R3			0011 0 R3 1011 1101	30BD	12345678	22
MODN	M B6.X2			X2 B6 0101 0101	0055	12345678	22
MODN	M X3			0011 1 X3 1011 1101	38BD	12345678	22
MON		C8		C8 1010 0110	00A6	-----678	80
MOV	M B8		R3	B8 0110 1 R3	0068	12345678	25
MOV	M P8		R3	P8 1001 1 R3	0098	12345678	25
MOV	R3	M B8		B8 0111 0 R3	0070	12345678	25
MOV	R3		R3	0 R3 1 R3 1011 1011	08BB	12345678	25
MOV	R3	M B6.X2		X2 B6 1000 0 R3	0080	12345678	25
MOV	R3	M B0.X3		1 R3 1 X3 1011 1011	88BB	12345678	25
MOV	M B6.X2		R3	X2 B6 0001 1 R3	0018	12345678	25
MOV	X3		R3	0 X3 1 R3 1111 1011	08FB	12345678	25
MOV	X3	M B0.X3		1 X3 1 X3 1111 1011	88FB	12345678	25
MOVB	R2	M BB6.X2		X2 B6 1001 00R2	0090	12345678	26
MOVB	M BB6.X2		R2	X2 B6 0111 10R2	0078	12345678	26
MOVC	M C4		X3	0 X3 C4 0101 0000	0050	12345678	24
MOVC	M C8		R3	C8 0100 1 R3	0048	12345678	24
MOVC	M CN4		X3	0 X3 CN4 0001 0000	0010	12345678	24
MOVC	M CN8		R3	CN8 0010 1 R3	0028	12345678	24
MOVL	R2	M B6.X2		X2 B6 1101 00R2	00D0	12345678	27
MOVL	B6.X2		R2	X2 B6 0110 00R2	0060	12345678	27
MOVM	(M) X3		X3	0 X3 1 X3 1110 1001	08E9	12345678	30
MUL	R3	M B0.X3		0 R3 1 X3 1110 1010	08EA	12345678	48
MUL	X3	M B0.X3		1 X3 1 X3 1110 1010	88EA	12345678	48
MVP	M P6.X2		R2	X2 P6 0010 00R2	0020	12345678	28
NEG			R3	1111 0 R3 1011 1100	F0BC	12345678	51
NEG		M	B0.X3	1111 1 X3 1011 1100	F8BC	12345678	51
NMI				1011 0010 1011 1100	B2BC	-------8	83
PUT	(M) R3		X3	1 R3 1 X3 1110 1001	88E9	12345678	31
PUT	(M) X3		X3	1 X3 1 X3 1110 1000	88E8	12345678	31
RELS	X3	C4		0 X3 C4 0001 0001	0011	12345678	68
RESS	X3	C4		0 X3 C4 0101 0001	0051	12345678	69
RIO	R3 (M)		R3	0 R3 1 R3 1001 0111	0897	12345678	70
RIO	X3 (M)		R3	0 R3 1 X3 1101 0111	08D7	12345678	70
RPZ	M	P6.X2		X2 P6 1010 0111	00A7	-----678	40
RTM	M	P6.X2		X2 P6 1110 0110	00E6	-----678	81
RTMI	M	B6		10 B6 1110 0011	80E3	-----678	81
SBN	R3 (M)	C4		0 R3 C4 1111 0000	00F0	12345678	57
SBN	X3 (M)	C4		0 X3 C4 1111 0001	00F1	12345678	57
SBNP	R3 (M)	C4		0 R3 C4 1011 0000	00B0	12345678	57
SBNP	X3 (M)	C4		0 X3 C4 1011 0001	00B1	12345678	57
SBZ	R3 (M)	C4		0 R3 C4 1111 0010	00F2	12345678	57
SBZ	X3 (M)	C4		0 X3 C4 1111 0011	00F3	12345678	57
SBZP	R3 (M)	C4		0 R3 C4 1011 0010	00B2	12345678	57
SBZP	X3 (M)	C4		0 X3 C4 1011 0011	00B3	12345678	57
SEQ	R3		R3	0 R3 0 R3 1111 1010	00FA	12345678	59
SEQ	R3	M B0.X3		0 R3 1 X3 1111 1010	08FA	12345678	59
SEQ	X3		R3	1 X3 0 R3 1111 1010	80FA	12345678	59
SEQ	X3	M B0.X3		1 X3 1 X3 1111 1010	88FA	12345678	59
SEQ	R3	M C4		1 R3 C4 0111 1110	807E	12745678	59
SEQ	R3	M CN4		0 R3 CN4 0111 1110	007E	12745678	59
SEQ	X3	M C4		1 X3 C4 0111 1111	807F	12745678	59
SEQ	X3	M CN4		0 X3 CN4 0111 1111	007F	12745678	59
SEQP	R3	M C4		1 R3 C4 0011 1110	803E	12345678	59
SEQP	R3	M CN4		0 R3 CN4 0011 1110	003E	12345678	59

Mnemonic OP1 OP2 OP3 Binary Hexa CPU-type Page

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
SEQP	R3		R3	0 R3 0 R3 1011 1010	00BA	12345678	59
SEQP	R3	M	BO.X3	0 R3 1 X3 1011 1010	08BA	12345678	59
SEQP	X3	M	C4	1 X3 C4 0011 1111	803F	12345678	59
SEQP	X3	M	CN4	0 X3 CN4 0011 1111	003F	12345678	59
SEQP	X3		R3	1 X3 0 R3 1011 1010	80BA	12345678	59
SEQP	X3	M	BO.X3	1 X3 1 X3 1011 1010	88BA	12345678	59
SETS	R3		C4	0 R3 C4 1010 0011	00A3	12345678	67
SETS	R3		R3	0 R3 0 R3 0101 0010	0052	12345678	67
SETS	R3	M	BO.X3	0 R3 1 X3 0101 0010	0852	12345678	67
SETS	X3		C4	0 X3 C4 1110 0011	00E3	12345678	67
SETS	X3		R3	1 X3 0 R3 0101 0011	8053	12345678	67
SETS	X3	M	BO.X3	1 X3 1 X3 0101 0011	8853	12345678	67
SGE	R3	M	C4	1 R3 C4 0100 0010	8042	12345678	59
SGE	R3		R3	0 R3 0 R3 1111 0110	00F6	12345678	59
SGE	R3	M	BO.X3	0 R3 1 X3 1111 0110	08F6	12345678	59
SGE	X3	M	C4	1 X3 C4 0100 0011	8043	12345678	59
SGE	X3		R3	1 X3 0 R3 1111 0110	80F6	12345678	59
SGE	X3	M	BO.X3	1 X3 1 X3 1111 0110	88F6	12345678	59
SGEP	R3	M	C4	1 R3 C4 0000 0010	8002	12345678	59
SGEP	R3		R3	0 R3 0 R3 1011 0110	00B6	12345678	59
SGEP	R3	M	BO.X3	0 R3 1 X3 1011 0110	08B6	12345678	59
SGEP	X3	M	C4	1 X3 C4 0000 0011	8003	12345678	59
SGEP	X3		R3	1 X3 0 R3 1011 0110	80B6	12345678	59
SGEP	X3	M	BO.X3	1 X3 1 X3 1011 0110	88B6	12345678	59
SHS	R3	M	C4	1 R3 C4 0100 0110	8046	12345678	59
SHS	R3		R3	0 R3 0 R3 1111 0111	00F7	12345678	59
SHS	R3	M	BO.X3	0 R3 1 X3 1111 0111	08F7	12345678	59
SHS	X3	M	C4	1 X3 C4 0100 0111	8047	12345678	59
SHS	X3		R3	1 X3 0 R3 1111 0111	80F7	12345678	59
SHS	X3	M	BO.X3	1 X3 1 X3 1111 0111	88F7	12345678	59
SHSP	R3	M	C4	1 R3 C4 0000 0110	8006	12345678	59
SHSP	R3		R3	0 R3 0 R3 1011 0111	00B7	12345678	59
SHSP	R3	M	BO.X3	0 R3 1 X3 1011 0111	08B7	12345678	59
SHSP	X3	M	C4	1 X3 C4 0000 0111	8007	12345678	59
SHSP	X3		R3	1 X3 0 R3 1011 0111	80B7	12345678	59
SHSP	X3	M	BO.X3	1 X3 1 X3 1011 0111	88B7	12345678	59
SIO	X3	(M)	R3	0 R3 1 X3 1101 0110	08D6	------78	70
SIO	R3	(M)	R3	0 R3 1 R3 1001 0110	0896	12345678	70
SLC	R3	(M)	C4	1 R3 C4 1010 0010	80A2	12345678	63
SLC	X3	(M)	C4	1 X3 C4 1110 0010	80E2	12345678	63
SLL	R3	(M)	C4	1 R3 C4 1010 0000	80A0	12345678	63
SLL	X3	(M)	C4	1 X3 C4 1110 0000	80E0	12345678	63
SLLL	R3	(M)	C4	0 R3 C4 1010 0001	00A1	12345678	64
SLLL	X3	(M)	C4	0 X3 C4 1110 0001	00E1	12345678	64
SLO	R3	M	C4	1 R3 C4 0100 0100	8044	12345678	59
SLO	R3		R3	0 R3 0 R3 1111 0101	00F5	12345678	59
SLO	R3	M	BO.X3	0 R3 1 X3 1111 0101	08F5	12345678	59
SLO	X3	M	C4	1 X3 C4 0100 0101	8045	12345678	59
SLO	X3		R3	1 X3 0 R3 1111 0101	80F5	12345678	59
SLO	X3	M	BO.X3	1 X3 1 X3 1111 0101	88F5	12345678	59
SLOP	R3	M	C4	1 R3 C4 0000 0100	8004	12345678	59
SLOP	R3		R3	0 R3 0 R3 1011 0101	00B5	12345678	59
SLOP	R3	M	BO.X3	0 R3 1 X3 1011 0101	08B5	12345678	59
SLOP	X3	M	C4	1 X3 C4 0000 0101	8005	12345678	59
SLOP	X3		R3	1 X3 0 R3 1011 0101	80B5	12345678	59
SLOP	X3	M	BO.X3	1 X3 1 X3 1011 0101	88B5	12345678	59
SLS			R3	0000 0 R3 1011 1101	00BD	-----678	72
SLS		M	BO.X3	0000 1 X3 1011 1101	08BD	-----678	72
SLT	R3	M	C4	1 R3 C4 0100 0000	8040	12345678	59
SLT	R3		R3	0 R3 0 R3 1111 0100	00F4	12345678	59
SLT	R3	M	BO.X3	0 R3 1 X3 1111 0100	08F4	12345678	59
SLT	X3	M	C4	1 X3 C4 0100 0001	8041	12345678	59
SLT	X3		R3	1 X3 0 R3 1111 0100	80F4	12345678	59
SLT	X3	M	BO.X3	1 X3 1 X3 1111 0100	88F4	12345678	59
SLTP	R3	M	C4	1 R3 C4 0000 0000	8000	12345678	59
SLTP	R3		R3	0 R3 0 R3 1011 0100	00B4	12345678	59
SLTP	R3	M	BO.X3	0 R3 1 X3 1011 0100	08B4	12345678	59
SLTP	X3	M	C4	1 X3 C4 0000 0001	8001	12345678	59
SLTP	X3		R3	1 X3 0 R3 1011 0100	80B4	12345678	59
SLTP	X3	M	BO.X3	1 X3 1 X3 1011 0100	88B4	12345678	59
SNE	R3		R3	0 R3 0 R3 1111 1000	00F8	12345678	59
SNE	R3	M	BO.X3	0 R3 1 X3 1111 1000	08F8	12345678	59
SNE	X3		R3	1 X3 0 R3 1111 1000	80F8	12345678	59
SNE	X3	M	BO.X3	1 X3 1 X3 1111 1000	88F8	12345678	59

Mnemonic OP1 OP2 OP3 Binary Hexa CPU-type Page

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
SNE		R3	M C4	1 R3 C4 0111 1100	807C	12745678	59
SNE		R3	M CN4	0 R3 CN4 0111 1100	007C	12745678	59
SNE		X3	M C4	1 X3 C4 0111 1101	807D	12745678	59
SNE		X3	M CN4	0 X3 CN4 0111 1101	007D	12745678	59
SNEP		R3	M C4	1 R3 C4 0011 1100	803C	12345678	59
SNEP		R3	M CN4	0 R3 CN4 0011 1100	003C	12345678	59
SNEP		R3	R3	0 R3 0 R3 1011 1000	00B8	12345678	59
SNEP		R3	M B0.X3	0 R3 1 X3 1011 1000	08B8	12345678	59
SNEP		X3	M C4	1 X3 C4 0011 1101	803D	12345678	59
SNEP		X3	M CN4	0 X3 CN4 0011 1101	003D	12345678	59
SNEP		X3	R3	1 X3 0 R3 1011 1000	80B8	12345678	59
SNEP		X3	M B0.X3	1 X3 1 X3 1011 1000	88B8	12345678	59
SOB		R3	M LN8	LN8 1100 0 R3	00C0	12345678	43
SON	M B6.X2			X2 B6 0011 1011	003B	12345678	58
SONP	M B6.X2			X2 B6 0011 1001	0039	12345678	58
SOZ	M B6.X2			X2 B6 0011 1010	003A	12345678	58
SOZP	M B6.X2			X2 B6 0011 1000	0038	12345678	58
SRA		R3	(M) C4	1 R3 C4 0010 0110	8026	12345678	63
SRA		X3	(M) C4	1 X3 C4 0110 0110	8066	12345678	63
SRL		R3	(M) C4	1 R3 C4 0010 0100	8024	12345678	63
SRL		X3	(M) C4	1 X3 C4 0110 0100	8064	12345678	63
SRLL		R3	(M) C4	0 R3 C4 0010 0101	0025	12345678	64
SRLL		X3	(M) C4	0 X3 C4 0110 0101	0065	12345678	64
SSS			R3	0001 0 R3 1011 1100	10BC	-----678	72
SSS			M B0.X3	0001 1 X3 1011 1100	18BC	-----678	72
STC			C4	1110 C4 1011 1110	E0BE	12345678	33
STC		R3	M B0.X3	0 R3 0 X3 1011 1001	00B9	12345678	33
STC		X3	M B0.X3	0 X3 0 X3 1111 1001	00F9	12345678	33
SUB		R3	R3	0 R3 0 R3 1000 1011	008B	12345678	47
SUB		R3	M B0.X3	0 R3 1 X3 1000 1011	088B	12345678	47
SUB		X3	R3	1 X3 0 R3 1000 1011	808B	12345678	47
SUB		X3	M B0.X3	1 X3 1 X3 1000 1011	888B	12345678	47
SUBU		R3	R3	0 R3 0 R3 1101 0101	00D5	12345678	47
SUBU		R3	M B0.X3	0 R3 1 X3 1101 0101	08D5	12345678	47
SUBU		X3	R3	1 X3 0 R3 1101 0101	80D5	12345678	47
SUBU		X3	M B0.X3	1 X3 1 X3 1101 0101	88D5	12345678	47
SVL			R3	0101 0 R3 1011 1101	50BD	-----678	75
SVL			M B0.X3	0101 1 X3 1011 1101	58BD	-----678	75
SVP			L4	0010 L4 1011 1110	20BE	1234-678	76
SVS			R3	1101 0 R3 1011 1100	D0BC	12345678	72
SVS			M B0.X3	1101 1 X3 1011 1100	D8BC	12345678	72
SVT			R3	1100 0 R3 1011 1100	C0BC	12345678	72
SVT			M B0.X3	1100 1 X3 1011 1100	C8BC	12345678	72
SWP			R3	0110 0 R3 1011 1100	60BC	12345678	51
SWP			M B0.X3	0110 1 X3 1011 1100	68BC	12345678	51
SXT		X3	(M) C4	0 X3 C4 0110 0111	0067	12345678	54
TRP			C4	0000 C4 1011 1110	00BE	12345678	84
TST				1000 0011 1011 1110	83BE	---45678	86
UNS			C4	1010 C4 1011 1110	A0BE	12345678	34
UNS		R3	M B0.X3	0 R3 1 X3 1011 1001	08B9	12345678	34
UNS		X3	M B0.X3	0 X3 1 X3 1111 1001	08F9	12345678	34
WIO		R3	(M) R3	1 R3 1 R3 1001 0111	8897	12345678	70
WIO		X3	(M) R3	1 R3 1 X3 1101 0111	88D7	12345678	70
XCH		R3	R3	0 R3 0 R3 1110 1110	00EE	12345678	32
XCH		R3	M B0.X3	0 R3 1 X3 1110 1110	08EE	12345678	32
XCH		X3	R3	1 X3 0 R3 1110 1111	80EF	12345678	32
XCH		X3	M B0.X3	1 X3 1 X3 1110 1111	88EF	12345678	32
XCU			X3	1000 0 X3 1011 1100	80BC	12345678	79
XCUI			M B0.X3	1000 1 X3 1011 1100	88BC	12345678	79
XOR		R3	R3	0 R3 0 R3 1110 1000	00E8	12345678	50
XOR		X3	R3	1 X3 0 R3 1110 1000	80E8	12345678	50
XTR		R3	C4	0 R3 C4 1110 1100	00EC	12345678	55
XTR		X3	C4	1 X3 C4 1110 1101	80ED	12345678	55

Mnemonic OP1 OP2 OP3 Binary Hexa CPU-type Page

CR80MX STANDARD INSTRUCTION SET SUMMARY

Sorted after

OPCODES

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
SLTP	R3	M	C4	1 R3 C4 0000 0000	8000	12345678	59
SLTP	X3	M	C4	1 X3 C4 0000 0001	8001	12345678	59
SGEP	R3	M	C4	1 R3 C4 0000 0010	8002	12345678	59
SGEP	X3	M	C4	1 X3 C4 0000 0011	8003	12345678	59
SLOP	R3	M	C4	1 R3 C4 0000 0100	8004	12345678	59
SLOP	X3	M	C4	1 X3 C4 0000 0101	8005	12345678	59
SHSP	R3	M	C4	1 R3 C4 0000 0110	8006	12345678	59
SHSP	X3	M	C4	1 X3 C4 0000 0111	8007	12345678	59
ADD	R3	M	B6.X2	X2 B6 0000 1 R3	0008	12345678	46
MOVC	M CN4		X3	0 X3 CN4 0001 0000	0010	12345678	24
RELS	X3		C4	0 X3 C4 0001 0001	0011	12345678	68
MMP	M PO.X3		R3	1 X3 0 R3 0001 0001	8011	-----678	29
JMPI		M	B6.X2	X2 B6 0001 0010	0012	12345678	38
DEC		M	B6.X2	X2 B6 0001 0011	0013	12345678	51
MODN	M B8			B8 0001 0100	0014	12345678	22
MODN	M P8			P8 0001 0101	0015	12345678	22
INC		M	B6.X2	X2 B6 0001 0110	0016	12345678	51
JPZI		M	P8	P8 0001 0111	0017	-----678	39
MOV	M B6.X2		R3	X2 B6 0001 1 R3	0018	12345678	25
MVP	M P6.X2		R2	X2 P6 0010 00R2	0020	12345678	28
SRL	R3 (M)	C4		1 R3 C4 0010 0100	8024	12345678	63
SRLL	R3 (M)	C4		0 R3 C4 0010 0101	0025	12345678	64
SRA	R3 (M)	C4		1 R3 C4 0010 0110	8026	12345678	63
CLR		M	B6.X2	X2 B6 0010 0111	0027	12345678	51
MOVC	M CN8		R3	CN8 0010 1 R3	0028	12345678	24
ADD	M B6.X2		R3	X2 B6 0011 0 R3	0030	12345678	46
SOZP	M B6.X2			X2 B6 0011 1000	0038	12345678	58
SONP	M B6.X2			X2 B6 0011 1001	0039	12345678	58
SOZ	M B6.X2			X2 B6 0011 1010	003A	12345678	58
SON	M B6.X2			X2 B6 0011 1011	003B	12345678	58
SNEP	R3	M	CN4	0 R3 CN4 0011 1100	003C	12345678	59
SNEP	R3	M	C4	1 R3 C4 0011 1100	803C	12345678	59
SNEP	X3	M	CN4	0 X3 CN4 0011 1101	003D	12345678	59
SNEP	X3	M	C4	1 X3 C4 0011 1101	803D	12345678	59
SEQP	R3	M	CN4	0 R3 CN4 0011 1110	003E	12345678	59
SEQP	R3	M	C4	1 R3 C4 0011 1110	803E	12345678	59
SEQP	X3	M	CN4	0 X3 CN4 0011 1111	003F	12345678	59
SEQP	X3	M	C4	1 X3 C4 0011 1111	803F	12345678	59
SLT	R3	M	C4	1 R3 C4 0100 0000	8040	12345678	59
SLT	X3	M	C4	1 X3 C4 0100 0001	8041	12345678	59
SGE	R3	M	C4	1 R3 C4 0100 0010	8042	12345678	59
SGE	X3	M	C4	1 X3 C4 0100 0011	8043	12345678	59
SLO	R3	M	C4	1 R3 C4 0100 0100	8044	12345678	59
SLO	X3	M	C4	1 X3 C4 0100 0101	8045	12345678	59
SHS	R3	M	C4	1 R3 C4 0100 0110	8046	12345678	59
SHS	X3	M	C4	1 X3 C4 0100 0111	8047	12345678	59
MOVC	M C8		R3	C8 0100 1 R3	0048	12345678	24
MOVC	M C4		X3	0 X3 C4 0101 0000	0050	12345678	24
RESS	X3		C4	0 X3 C4 0101 0001	0051	12345678	69
SETS	R3		R3	0 R3 0 R3 0101 0010	0052	12345678	67
SETS	R3	M	BO.X3	0 R3 1 X3 0101 0010	0852	12345678	67
SETS	X3		R3	1 X3 0 R3 0101 0011	8053	12345678	67
SETS	X3	M	BO.X3	1 X3 1 X3 0101 0011	8853	12345678	67
MOD	M B6.X2			X2 B6 0101 0100	0054	12345678	22
MODN	M B6.X2			X2 B6 0101 0101	0055	12345678	22
MOD8	M C8			C8 0101 0110	0056	12345678	21
JMP		M	LN10	LN8 0101 10L2	0058	12345678	36
JMP		M	LN8	LN8 0101 1100	005C	12345678	36
JMP	S2	M	LN8	LN8 0101 11S2	005C	12345678	36
MOVL	B6.X2		R2	X2 B6 0110 00R2	0060	12345678	27
SRL	X3 (M)	C4		1 X3 C4 0110 0100	8064	12345678	63
SRLL	X3 (M)	C4		0 X3 C4 0110 0101	0065	12345678	64
SRA	X3 (M)	C4		1 X3 C4 0110 0110	8066	12345678	63
SXT	X3 (M)	C4		0 X3 C4 0110 0111	0067	12345678	54
MOV	M B8		R3	B8 0110 1 R3	0068	12345678	25
MOV	R3	M	B8	B8 0111 0 R3	0070	12345678	25
MOVB	M BB6.X2		R2	X2 B6 0111 10R2	0078	12345678	26
SNE	R3	M	CN4.	0 R3 CN4 0111 1100	007C	12745678	59
SNE	R3	M	C4	1 R3 C4 0111 1100	807C	12745678	59
SNE	X3	M	CN4	0 X3 CN4 0111 1101	007D	12745678	59
SNE	X3	M	C4	1 X3 C4 0111 1101	807D	12745678	59
SEQ	R3	M	CN4	0 R3 CN4 0111 1110	007E	12745678	59
SEQ	R3	M	C4	1 R3 C4 0111 1110	807E	12745678	59
SEQ	X3	M	CN4	0 X3 CN4 0111 1111	007F	12745678	59

Mnemonic OP1 OP2 OP3 Binary Hexa CPU-type Page

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
SEQ		X3 M	C4	1 X3 C4 0111 1111	807F	12745678	59
MOV		R3 M	B6.X2	X2 B6 1000 0 R3	0080	12345678	25
CLRS		R3	R3	0 R3 0 R3 1000 1000	0088	12345678	66
CLRS		R3 M	B0.X3	0 R3 1 X3 1000 1000	0888	12345678	66
CLRS		X3	R3	1 X3 0 R3 1000 1001	8089	12345678	66
CLRS		X3 M	B0.X3	1 X3 1 X3 1000 1001	8889	12345678	66
ADD		R3	R3	0 R3 0 R3 1000 1010	008A	12345678	46
ADD		R3 M	B0.X3	0 R3 1 X3 1000 1010	088A	12345678	46
ADD		X3	R3	1 X3 0 R3 1000 1010	808A	12345678	46
ADD		X3 M	B0.X3	1 X3 1 X3 1000 1010	888A	12345678	46
SUB		R3	R3	0 R3 0 R3 1000 1011	008B	12345678	47
SUB		R3 M	B0.X3	0 R3 1 X3 1000 1011	088B	12345678	47
SUB		X3	R3	1 X3 0 R3 1000 1011	808B	12345678	47
SUB		X3 M	B0.X3	1 X3 1 X3 1000 1011	888B	12345678	47
AND		R3	R3	0 R3 0 R3 1000 1100	008C	12345678	50
AND		R3 M	B0.X3	0 R3 1 X3 1000 1100	088C	12345678	50
AND		X3	R3	1 X3 0 R3 1000 1100	808C	12345678	50
AND		X3 M	B0.X3	1 X3 1 X3 1000 1100	888C	12345678	50
IOR		R3	R3	0 R3 0 R3 1000 1101	008D	12345678	50
IOR		R3 M	B0.X3	0 R3 1 X3 1000 1101	088D	12345678	50
IOR		X3	R3	1 X3 0 R3 1000 1101	808D	12345678	50
IOR		X3 M	B0.X3	1 X3 1 X3 1000 1101	888D	12345678	50
INCD		R3	R3	0 R3 0 R3 1000 1110	008E	12345678	52
DECD		R3	R3	0 R3 1 R3 1000 1110	088E	12345678	52
INCD		X3 M	B0.X3	1 X3 0 X3 1000 1111	808F	12345678	52
DECD		X3 M	B0.X3	1 X3 1 X3 1000 1111	888F	12345678	52
MOVB		R2 M	BB6.X2	X2 B6 1001 00R2	0090	12345678	26
MOD	M B8			B8 1001 0100	0094	12345678	22
MOD	M P8			P8 1001 0101	0095	12345678	22
SIO		R3 (M)	R3	0 R3 1 R3 1001 0110	0896	12345678	70
CIO		R3 (M)	R3	1 R3 1 R3 1001 0110	8896	12345678	70
RIO		R3 (M)	R3	0 R3 1 R3 1001 0111	0897	12345678	70
WIO		R3 (M)	R3	1 R3 1 R3 1001 0111	8897	12345678	70
MOV	M P8		R3	P8 1001 1 R3	0098	12345678	25
SLL		R3 (M)	C4	1 R3 C4 1010 0000	80A0	12345678	63
SLLL		R3 (M)	C4	0 R3 C4 1010 0001	00A1	12345678	64
SLC		R3 (M)	C4	1 R3 C4 1010 0010	80A2	12345678	63
SETS		R3	C4	0 R3 C4 1010 0011	00A3	12345678	67
MODC	M CN8			CN8 1010 0100	00A4	12345678	20
MOD4	M CN8			CN8 1010 0101	00A5	12345678	21
MON			C8	C8 1010 0110	00A6	-----678	80
RPZ		M	P6.X2	X2 P6 1010 0111	00A7	-----678	40
ADDC	M CN8		R3	CN8 1010 1 R3	00A8	12345678	45
SBNP		R3 (M)	C4	0 R3 C4 1011 0000	00B0	12345678	57
JON		R3	LN4	1 R3 LN4 1011 0000	80B0	12345678	41
SBNP		X3 (M)	C4	0 X3 C4 1011 0001	00B1	12345678	57
JON		X3	LN4	1 X3 LN4 1011 0001	80B1	12345678	41
SBZP		R3 (M)	C4	0 R3 C4 1011 0010	00B2	12345678	57
JOZ		R3	LN4	1 R3 LN4 1011 0010	80B2	12345678	41
SBZP		X3 (M)	C4	0 X3 C4 1011 0011	00B3	12345678	57
JOZ		X3	LN4	1 X3 LN4 1011 0011	80B3	12345678	41
SLTP		R3	R3	0 R3 0 R3 1011 0100	00B4	12345678	59
SLTP		R3 M	B0.X3	0 R3 1 X3 1011 0100	08B4	12345678	59
SLTP		X3	R3	1 X3 0 R3 1011 0100	80B4	12345678	59
SLTP		X3 M	B0.X3	1 X3 1 X3 1011 0100	88B4	12345678	59
SLOP		R3	R3	0 R3 0 R3 1011 0101	00B5	12345678	59
SLOP		R3 M	B0.X3	0 R3 1 X3 1011 0101	08B5	12345678	59
SLOP		X3	R3	1 X3 0 R3 1011 0101	80B5	12345678	59
SLOP		X3 M	B0.X3	1 X3 1 X3 1011 0101	88B5	12345678	59
SGEP		R3	R3	0 R3 0 R3 1011 0110	00B6	12345678	59
SGEP		R3 M	B0.X3	0 R3 1 X3 1011 0110	08B6	12345678	59
SGEP		X3	R3	1 X3 0 R3 1011 0110	80B6	12345678	59
SGEP		X3 M	B0.X3	1 X3 1 X3 1011 0110	88B6	12345678	59
SHSP		R3	R3	0 R3 0 R3 1011 0111	00B7	12345678	59
SHSP		R3 M	B0.X3	0 R3 1 X3 1011 0111	08B7	12345678	59
SHSP		X3	R3	1 X3 0 R3 1011 0111	80B7	12345678	59
SHSP		X3 M	B0.X3	1 X3 1 X3 1011 0111	88B7	12345678	59
SNEP		R3	R3	0 R3 0 R3 1011 1000	00B8	12345678	59
SNEP		R3 M	B0.X3	0 R3 1 X3 1011 1000	08B8	12345678	59
SNEP		X3	R3	1 X3 0 R3 1011 1000	80B8	12345678	59
SNEP		X3 M	B0.X3	1 X3 1 X3 1011 1000	88B8	12345678	59
STC		R3 M	B0.X3	0 R3 0 X3 1011 1001	00B9	12345678	33
UNS		R3 M	B0.X3	0 R3 1 X3 1011 1001	08B9	12345678	34
SEQP		R3	R3	0 R3 0 R3 1011 1010	00BA	12345678	59

Mnemonic OP1 OP2 OP3 Binary Hexa CPU-type Page

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
SEQP	R3	M	B0.X3	0 R3 1 X3 1011 1010	08BA	12345678	59
SEQP	X3		R3	1 X3 0 R3 1011 1010	80BA	12345678	59
SEQP	X3	M	B0.X3	1 X3 1 X3 1011 1010	88BA	12345678	59
MOV	R3		R3	0 R3 1 R3 1011 1011	08BB	12345678	25
MOV	R3	M	B0.X3	1 R3 1 X3 1011 1011	88BB	12345678	25
LDP			R3	0000 0 R3 1011 1100	00BC	1234-678	77
LDP		M	B0.X3	0000 1 X3 1011 1100	08BC	1234-678	77
SSS			R3	0001 0 R3 1011 1100	10BC	-----678	72
SSS		M	B0.X3	0001 1 X3 1011 1100	18BC	-----678	72
DICP	R3			0010 0 R3 1011 1100	20BC	12345678	53
DICP	M B0.X3			0010 1 X3 1011 1100	28BC	12345678	53
DDCP	R3			0011 0 R3 1011 1100	30BC	12345678	53
DDCP	M B0.X3			0011 1 X3 1011 1100	38BC	12345678	53
LDN			R3	0100 0 R3 1011 1100	40BC	1234-678	77
LDN		M	B0.X3	0100 1 X3 1011 1100	48BC	1234-678	77
LDT			R3	0101 0 R3 1011 1100	50BC	12345678	72
LDT		M	B0.X3	0101 1 X3 1011 1100	58BC	12345678	72
SWP			R3	0110 0 R3 1011 1100	60BC	12345678	51
SWP		M	B0.X3	0110 1 X3 1011 1100	68BC	12345678	51
XCU			X3	1000 0 X3 1011 1100	80BC	12345678	79
XCUI		M	B0.X3	1000 1 X3 1011 1100	88BC	12345678	79
INV			R3	1001 0 R3 1011 1100	90BC	12345678	51
INV		M	B0.X3	1001 1 X3 1011 1100	98BC	12345678	51
ALT				1011 0000 1011 1100	B0BC	--345678	87
LBR				1011 0001 1011 1100	B1BC	-----678	74
NMI				1011 0010 1011 1100	B2BC	-------8	83
SVT			R3	1100 0 R3 1011 1100	C0BC	12345678	72
SVT		M	B0.X3	1100 1 X3 1011 1100	C8BC	12345678	72
SVS			R3	1101 0 R3 1011 1100	D0BC	12345678	72
SVS		M	B0.X3	1101 1 X3 1011 1100	D8BC	12345678	72
LDS			R3	1110 0 R3 1011 1100	E0BC	12345678	72
LDS		M	B0.X3	1110 1 X3 1011 1100	E8BC	12345678	72
NEG			R3	1111 0 R3 1011 1100	F0BC	12345678	51
NEG		M	B0.X3	1111 1 X3 1011 1100	F8BC	12345678	51
SLS			R3	0000 0 R3 1011 1101	00BD	-----678	72
SLS		M	B0.X3	0000 1 X3 1011 1101	08BD	-----678	72
MODN	M	R3		0011 0 R3 1011 1101	30BD	12345678	22
MODN	M	X3		0011 1 X3 1011 1101	38BD	12345678	22
SVL			R3	0101 0 R3 1011 1101	50BD	-----678	75
SVL		M	B0.X3	0101 1 X3 1011 1101	58BD	-----678	75
MOD	M	R3		0111 0 R3 1011 1101	70BD	12345678	22
MOD	M	X3		0111 1 X3 1011 1101	78BD	12345678	22
LDL			R3	1000 0 R3 1011 1101	80BD	-----678	75
LDL		M	B0.X3	1000 1 X3 1011 1101	88BD	-----678	75
TRP			C4	0000 C4 1011 1110	00BE	12345678	84
SVP			L4	0010 L4 1011 1110	20BE	1234-678	76
CPU	R3			0110 0 R3 1011 1110	60BE	----5-78	82
CPU	M B0.X3			0110 1 X3 1011 1110	68BE	----5-78	82
JVN		M	LN4	0111 LN4 1011 1110	70BE	12345678	42
CAE				1000 0000 1011 1110	80BE	---45678	85
CAD				1000 0001 1011 1110	81BE	---45678	85
TST				1000 0011 1011 1110	83BE	---45678	86
UNS			C4	1010 C4 1011 1110	A0BE	12345678	34
LDM			C4	1101 C4 1011 1110	D0BE	12345678	72
STC			C4	1110 C4 1011 1110	E0BE	12345678	33
JVN		M	L4	1111 L4 1011 1110	F0BE	12345678	42
JMP		M	P6.X2	X2 P6 1011 1111	00BF	12345678	37
SOB	R3	M	LN8	LN8 1100 0 R3	00C0	12345678	43
ADDC	M C8		R3	C8 1100 1 R3	00C8	12345678	45
MOVL	R2	M	B6.X2	X2 B6 1101 00R2	00D0	12345678	27
ADDU	R3		R3	0 R3 0 R3 1101 0100	00D4	12345678	46
ADDU	R3	M	B0.X3	0 R3 1 X3 1101 0100	08D4	12345678	46
ADDU	X3		R3	1 X3 0 R3 1101 0100	80D4	12345678	46
ADDU	X3	M	B0.X3	1 X3 1 X3 1101 0100	88D4	12345678	46
SUBU	R3		R3	0 R3 0 R3 1101 0101	00D5	12345678	47
SUBU	R3	M	B0.X3	0 R3 1 X3 1101 0101	08D5	12345678	47
SUBU	X3		R3	1 X3 0 R3 1101 0101	80D5	12345678	47
SUBU	X3	M	B0.X3	1 X3 1 X3 1101 0101	88D5	12345678	47
SIO	X3	(M)	R3	0 R3 1 X3 1101 0110	08D6	------78	70
CIO	X3	(M)	R3	1 R3 1 X3 1101 0110	88D6	------78	70
RIO	X3	(M)	R3	0 R3 1 X3 1101 0111	08D7	12345678	70
WIO	X3	(M)	R3	1 R3 1 X3 1101 0111	88D7	12345678	70
JMP		M	L10	L8 1101 10L2	00D8	12345678	36
JMP		M	L8	L8 1101 1100	00DC	12345678	36

Mnemonic OP1 OP2 OP3 Binary Hexa CPU-type Page

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
JMP		S2 M	L8	L8 1101 11S2	00DC	12345678	36
SLL		X3 (M)	C4	1 X3 C4 1110 0000	80E0	12345678	63
SLLL		X3 (M)	C4	0 X3 C4 1110 0001	00E1	12345678	64
SLC		X3 (M)	C4	1 X3 C4 1110 0010	80E2	12345678	63
SETS		X3	C4	0 X3 C4 1110 0011	00E3	12345678	67
RTMI		M	B6	10 B6 1110 0011	80E3	-----678	81
MODC	M C8			C8 1110 0100	00E4	12345678	20
MOD4	M C8			C8 1110 0101	00E5	12345678	21
RTM		M	P6.X2	X2 P6 1110 0110	00E6	-----678	81
JMPI		M	B8	B8 1110 0111	00E7	12345678	38
XOR		R3	R3	0 R3 0 R3 1110 1000	00E8	12345678	50
XOR		X3	R3	1 X3 0 R3 1110 1000	80E8	12345678	50
PUT	(M)	X3	X3	1 X3 1 X3 1110 1000	88E8	12345678	31
MOVM	(M)	X3	X3	0 X3 1 X3 1110 1001	08E9	12345678	30
PUT	(M)	R3	X3	1 R3 1 X3 1110 1001	88E9	12345678	31
MUL		R3 M	B0.X3	0 R3 1 X3 1110 1010	08EA	12345678	48
MUL		X3 M	B0.X3	1 X3 1 X3 1110 1010	88EA	12345678	48
DIV		R3	R3	0 R3 0 R3 1110 1011	00EB	12345678	49
DIV		X3	R3	1 X3 0 R3 1110 1011	80EB	12345678	49
XTR		R3	C4	0 R3 C4 1110 1100	00EC	12345678	55
XTR		X3	C4	1 X3 C4 1110 1101	80ED	12345678	55
XCH		R3	R3	0 R3 0 R3 1110 1110	00EE	12345678	32
XCH		R3 M	B0.X3	0 R3 1 X3 1110 1110	08EE	12345678	32
XCH		X3	R3	1 X3 0 R3 1110 1111	80EF	12345678	32
XCH		X3 M	B0.X3	1 X3 1 X3 1110 1111	88EF	12345678	32
SBN		R3 (M)	C4	0 R3 C4 1111 0000	00F0	12345678	57
JON		R3	L4	1 R3 L4 1111 0000	80F0	12345678	41
SBN		X3 (M)	C4	0 X3 C4 1111 0001	00F1	12345678	57
JON		X3	L4	1 X3 L4 1111 0001	80F1	12345678	41
SBZ		R3 (M)	C4	0 R3 C4 1111 0010	00F2	12345678	57
JOZ		R3	L4	1 R3 L4 1111 0010	80F2	12345678	41
SBZ		X3 (M)	C4	0 X3 C4 1111 0011	00F3	12345678	57
JOZ		X3	L4	1 X3 L4 1111 0011	80F3	12345678	41
SLT		R3	R3	0 R3 0 R3 1111 0100	00F4	12345678	59
SLT		R3 M	B0.X3	0 R3 1 X3 1111 0100	08F4	12345678	59
SLT		X3	R3	1 X3 0 R3 1111 0100	80F4	12345678	59
SLT		X3 M	B0.X3	1 X3 1 X3 1111 0100	88F4	12345678	59
SLO		R3	R3	0 R3 0 R3 1111 0101	00F5	12345678	59
SLO		R3 M	B0.X3	0 R3 1 X3 1111 0101	08F5	12345678	59
SLO		X3	R3	1 X3 0 R3 1111 0101	80F5	12345678	59
SLO		X3 M	B0.X3	1 X3 1 X3 1111 0101	88F5	12345678	59
SGE		R3	R3	0 R3 0 R3 1111 0110	00F6	12345678	59
SGE		R3 M	B0.X3	0 R3 1 X3 1111 0110	08F6	12345678	59
SGE		X3	R3	1 X3 0 R3 1111 0110	80F6	12345678	59
SGE		X3 M	B0.X3	1 X3 1 X3 1111 0110	88F6	12345678	59
SHS		R3	R3	0 R3 0 R3 1111 0111	00F7	12345678	59
SHS		R3 M	B0.X3	0 R3 1 X3 1111 0111	08F7	12345678	59
SHS		X3	R3	1 X3 0 R3 1111 0111	80F7	12345678	59
SHS		X3 M	B0.X3	1 X3 1 X3 1111 0111	88F7	12345678	59
SNE		R3	R3	0 R3 0 R3 1111 1000	00F8	12345678	59
SNE		R3 M	B0.X3	0 R3 1 X3 1111 1000	08F8	12345678	59
SNE		X3	R3	1 X3 0 R3 1111 1000	80F8	12345678	59
SNE		X3 M	B0.X3	1 X3 1 X3 1111 1000	88F8	12345678	59
STC		X3 M	B0.X3	0 X3 0 X3 1111 1001	00F9	12345678	33
UNS		X3 M	B0.X3	0 X3 1 X3 1111 1001	08F9	12345678	34
SEQ		R3	R3	0 R3 0 R3 1111 1010	00FA	12345678	59
SEQ		R3 M	B0.X3	0 R3 1 X3 1111 1010	08FA	12345678	59
SEQ		X3	R3	1 X3 0 R3 1111 1010	80FA	12345678	59
SEQ		X3 M	B0.X3	1 X3 1 X3 1111 1010	88FA	12345678	59
MOV		X3	R3	0 X3 1 R3 1111 1011	08FB	12345678	25
MOV		X3 M	B0.X3	1 X3 1 X3 1111 1011	88FB	12345678	25
JMPI		M	P8	P8 1111 1100	00FC	12345678	38
JMPI		S2 M	P8	P8 1111 11S2	00FC	12345678	38

Mnemonic OP1 OP2 OP3 Binary Hexa CPU-type Page

CR80MX ALTERNATIVE INSTRUCTION SET SUMMARY

Sorted after
MNEMONICS and OPCODES

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
AAMOB	ABX33	ABX33	R3	0AX3 0AX3 1000 1 R3	0088	------78	91
AAMOW	AX33	AX33	R3	0AX3 0AX3 1000 0 R3	0080	------78	90
ALMOB	ABX33	BX33	R3	0AX3 1 X3 1000 1 R3	0888	------78	91
ALMOW	AX33	X3	R3	0AX3 1 X3 1000 0 R3	0880	------78	90
AMOB	R3	ABX33		1AX3 0 R3 1001 0001	8091	------78	93
AMOB	ABX33	R3		1AX3 1 R3 1001 0001	8891	------78	93
AMOW	R3	AX33		0AX3 0 R3 1001 0001	0091	------78	92
AMOW	AX33	R3		0AX3 1 R3 1001 0001	0891	------78	92
ARED	AX33	R3		1AX3 0 R3 0101 0011	8053	-------8	
ARELS	AX33	C4		1AX3 C4 1001 0010	8092	------78	95
ARESS	AX33	C4		0AX3 C4 1001 0010	0092	------78	94
ARSC	AX33	R3		0AX3 0 R3 0101 0011	0053	-------8	99
DRD	R3 (M)	R3		0 R3 0 R3 1001 0011	0093	------78	96
DRD	X3 (M)	R3		0 X3 1 R3 1001 0011	0893	------78	96
DWR	R3 (M)	R3		1 R3 0 R3 1001 0011	8093	------78	96
DWR	X3 (M)	R3		1 X3 1 R3 1001 0011	8893	------78	96
LAMOB	BX33	ABX33	R3	1 X3 0AX3 1000 1 R3	8088	------78	91
LAMOW	X3	AX33	R3	1 X3 0AX3 1000 0 R3	8080	------78	90
MRD	R3 (M)	R3		0 R3 0 R3 0101 0001	0051	------7-	97
MRD	X3 (M)	R3		0 X3 1 R3 0101 0001	0851	------7-	97
MWR	R3 (M)	R3		1 R3 0 R3 0101 0001	8051	------7-	97
MWR	X3 (M)	R3		1 X3 1 R3 0101 0001	8851	------7-	97
RSP	R3			0 R3 0000 0101 0010	0052	------78	98
RSP	X3			0 X3 1000 0101 0010	0852	------78	98
WSP	R3			1 R3 0000 0101 0010	8052	------78	98
WSP	X3			1 X3 1000 0101 0010	8852	------78	98

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page
MRD	R3 (M)	R3		0 R3 0 R3 0101 0001	0051	------7-	97
MRD	X3 (M)	R3		0 X3 1 R3 0101 0001	0851	------7-	97
MWR	R3 (M)	R3		1 R3 0 R3 0101 0001	8051	------7-	97
MWR	X3 (M)	R3		1 X3 1 R3 0101 0001	8851	------7-	97
RSP	R3			0 R3 0000 0101 0010	0052	------78	98
RSP	X3			0 X3 1000 0101 0010	0852	------78	98
WSP	R3			1 R3 0000 0101 0010	8052	------78	98
WSP	X3			1 X3 1000 0101 0010	8852	------78	98
ARSC	AX33	R3		0AX3 0 R3 0101 0011	0053	-------8	99
ARED	AX33	R3		1AX3 0 R3 0101 0011	8053	-------8	
AAMOW	AX33	AX33	R3	0AX3 0AX3 1000 0 R3	0080	------78	90
ALMOW	AX33	X3	R3	0AX3 1 X3 1000 0 R3	0880	------78	90
LAMOW	X3	AX33	R3	1 X3 0AX3 1000 0 R3	8080	------78	90
AAMOB	ABX33	ABX33	R3	0AX3 0AX3 1000 1 R3	0088	------78	91
ALMOB	ABX33	BX33	R3	0AX3 1 X3 1000 1 R3	0888	------78	91
LAMOB	BX33	ABX33	R3	1 X3 0AX3 1000 1 R3	8088	------78	91
AMOW	R3	AX33		0AX3 0 R3 1001 0001	0091	------78	92
AMOW	AX33	R3		0AX3 1 R3 1001 0001	0891	------78	92
AMOB	R3	ABX33		1AX3 0 R3 1001 0001	8091	------78	93
AMOB	ABX33	R3		1AX3 1 R3 1001 0001	8891	------78	93
ARESS	AX33	C4		0AX3 C4 1001 0010	0092	------78	94
ARELS	AX33	C4		1AX3 C4 1001 0010	8092	------78	95
DRD	R3 (M)	R3		0 R3 0 R3 1001 0011	0093	------78	96
DRD	X3 (M)	R3		0 X3 1 R3 1001 0011	0893	------78	96
DWR	R3 (M)	R3		1 R3 0 R3 1001 0011	8093	------78	96
DWR	X3 (M)	R3		1 X3 1 R3 1001 0011	8893	------78	96

Mnemonic	OP1	OP2	OP3	Binary	Hexa	CPU-type	Page

SAMPLE INSTRUCTION DESCRIPTION

```
****************     SHIFT SINGLE     *************** 1 2 3 4 5 6
***

INSTRUCTION:
  SHIFT LEFT CYCLICALLY                                SLC
  SHIFT RIGHT ARITHMETICALLY                           SRA
  SHIFT LEFT LOGICALLY                                 SLL
  SHIFT RIGHT LOGICALLY                                SRL
SYNONYME:
  SHIFT LEFT ARITHMETICALLY                            SLA     (SLL)

FORMAT:
                  OP1       OP2                NORM     MOD      BUS
  <INSTR>           R3 (M)C4                   4+S      6+S      1
  <INSTR>           X3 (M)C4                   4+S      6+S      3
                                               (S=SHIFTS)

FUNCTION:
  SHIFTS:= IF MODIFIED THEN   MOD MODULO 16
      ELSE IF C4=0 THEN                    16
      ELSE                                 C4;
  FOR I:= 1 STEP 1 UNTIL SHIFTS DO
    CASE INSTR OF
      BEGIN
        SLC: OP1:= OP1.[15:1]&OP1[15:14:15];
        SRA: OP1:=         OP1&OP1[14:15:15];
        SLL: OP1:=           0&OP1[15:14:15];
        SRL: OP1:=           0&OP1[14:15:15];
      END;
  CLEAR MODIFICATION;

DESCRIPTION:
  OP1 IS SHIFTED AS MANY BIT POSITIONS AS SPECIFIED BY OP2 OR
  THE MODIFY REGISTER. IF UNMODIFIED AND ZERO SHIFTS ARE
  SPECIFIED
  16 SHIFTS ARE PERFORMED. THE SHIFTS ARE PERFORMED AS INDICATED
  IN THE INSTRUCTION:
    SLC: SHIFT LEFT CYCLICALLY. THE 16 BIT OPERAND IS ROTATED
         LEFT. NO CARRY OR OTHER EXTERNAL BITS ARE INCORPORATED.
    SRA: SHIFT RIGHT ARITHMETICALLY. THE 16 BIT OPERAND IS
         SHIFTED RIGHT WITH SIGN EXTENSION IN THE UPPER BITS.
    SLL: SHIFT LEFT LOGICALLY. THE 16 BIT OPERAND IS SHIFTED
         LEFT WITH ZERO EXTENSION IN THE LOWER BITS.
    SRL: SHIFT RIGHT LOGICALLY. THE 16 BIT OPERAND IS SHIFTED
         RIGHT WITH ZERO EXTENSION IN THE UPPER BITS.

NOTICE:
  THE CONDITION CODES ARE UNCHANGED.

CODE:
                  OP1       OP2     BINARY                    HEXA
  SLC               R3      C4                                80A2
  SLC               X3      C4                                80E2
  SRA               R3      C4                                8026
  SRA               X3      C4                                8066
```

SLL	(SLA)	R3	C4	80A0
SLL	(SLA)	X3	C4	80E0
SRL		R3	C4	8024
SRL		X3	C4	8064

1. SUBGROUP HEADING

This heading specifies the type of instruction being described in the subgroup. If only one instruction is included, the heading is identical to the instruction name.

The heading also defines the type of CPU for which this instruction is implemented:

1 CR8001S with or without micro program RAM but without subbus
2 CR8001S with or without micro program RAM but with subbus
3 CR80101S or the CR8002D SCM_CPU
4 CR8003D/030P-/00 unmapped cache CPU
5 CR8003D/040PC/00 mapped cache CPU
6 CR80101S or CR8002D SCM_CPU for XAMOS
7 CR8003M/050PC/00 unmapped cache CPU for MX
8 CR8501 CMR

2. INSTRUCTION NAME

The name is a short form text for the instruction function. This column includes all the different instructions being described under this heading.

3. MNEMONIC

The instructions are identified by a 3 letter abbreviation for the instruction function, e.g.

MOD for modify
SEQ for skip if equal

In order to ease the understanding and the commenting, some of the instructions have two names which are synonomous. For example, for the skip instructions it is possible to use one of the two terms:

skip if condition is fulfilled
execute if condition is not fulfilled,

which are identical. E.g.

SEQ for skip if equal
INE for execute if not equal

Some mnemonics have a fourth character. The meaning of this character is:

B byte
C constant
D double, i.e. two individual operands
I indirect
L long, i.e. two consecutive operands
M multiple number of words
N negative
P pair, i.e. two consecutive operands
U update with carry or borrow
4 left shifted 4
8 left shifted 8

If the character is surrounded by square brackets, it is an optional feature which may be omitted.

4. FORMAT

This column shows the instruction in the different formats which are provided. If the mnemonic is replaced by

<instr>

it means that all the possibilities from the mnemonic fields may be inserted instead.

5. OPERANDS

An instruction may have either none, one, or two operands. The OP1 and OP2 columns define operand 1 which normally is the source operand, and operand 2 which normally is the destination operand. The columns are empty if no operands are used.

An operand is designated by one or two letters and a number. The first letter indicates the addressing mode:

- C constant
- B base relative
- P program relative
- L location relative
- R register
- X index register

The optional second letter is either

- B for byte address or
- N for negative (two's complement)

The number (n) defines the bit width of the field in the binary instruction code e.g. C8 means a constant contained in 8 bits, that is a constant from 0 to 255.

Rn: The operand is the contents of a register. This number of bits to indicate which register.

Bn: The operand is the contents of the memory location, the address of which is calculated as follows:
Contents of the n-bit field of the instruction plus contents of the base register.

Pn: The operand is the contents of the memory location, the address of which is calculated as follows:
Contents of the n-bit field of the instruction plus contents of the program register (PROG).

Ln: The operand is the contents of the n-bit field of the instruction. Used for location-relative jumps.

Cn: The operand is the contents of the n-bit field of the instruction.

Xn: Means that the n bits are used to indicate one of the registers as index. Thus:
B6.X2 means an operand which is:
The contents of the memory location, the address of which is calculated as follows:
Contents of the 6-bit field of the instruction plus the contents of the indicated (index) register plus the contents of the BASE register of the CPU.
Similarly X3 means an operand which is:
The contents of the memory location, the address of which is calculated as follows:
Contents of the indicated (index) register plus the

contents of the BASE register of the CPU.

The following list contains all operands as they are used in the instruction description:

C8 : an 8-bit constant
CN8 : an 8-bit constant that is negated before use (two's complement)
C4 : a 4-bit constant
CN4 : a 4-bit constant that is negated before use (two's complement).
B0 : a 0-bit process base relative constant. The notation is used to indicate that modification will be taken as a process base relative.
B6 : a 6-bit base relative constant
B8 : an 8-bit process base relative constant
BB6 : a 6 bit process base relative constant taken as a byte address
P6 : a 6-bit program base relative constant
P8 : an 8-bit program base relative constant
L4 : a 4-bit location relative constant
LN4 : a 4-bit location relative constant that is negated (two's complement) before use
L8 : an 8-bit location relative constant
LN8 : an 8-bit location relative constant that is negated (two's complement) before use
L10 : a 10-bit location relative constant
LN10: a 10-bit location relative constant that is negated (two's complement) before use
R2 : one of the registers: R0, R1, R2, or R3 used as an operand
R3 : one of the registers: R0, R1, R2, R3, R4, R5, R6, or R7 used as an operand.
X2 : one of the registers: X4, X5, X6, or X7 used as index
X3 : one of the registers: X0, X1, X2, X3, X4, X5, X6, or X7 used as index
S2 : one of the registers: S4, S5, or S6 used as return link

Please note that the left column above includes the number of bits in the instruction fields used to identify the register, while the description to the right includes the register number as it is used in the assembler coding. Please also note that the registers Ri, Xi, and Si are the same physical registers but the use is different.

An M in front of any oppart denotes that the instruction may be modified and that this oppart will be affected by the modifier. E.g. M P6.X2 will set up the absolute address:

Memory address :=
contents of program base register
+ a 6-bit displacement field from the instruction code
+ contents of the index register
+ contents of the modify register

The assembler will automatically insert one or two modify instructions, if the address field of the instruction cannot hold the required constant, and provided that the instruction can be modified.

The expansion of the displacement field is seen from the following table:

without modify	with 1 modify	with 2 modifies
DP0	DP8, DPN8	DP16
DP4, DP5	DP12, DPN12	DP16
DP6	DP12, DPN12	DP16
DP8, DPN8	DP16	-

where DP stands for C, B, P, or L.

A modify indication in parenthesis specifies that a modify will have a special meaning with the instruction and the assembler does never modify this instruction automatically.

6. TIMING SPECIFICATIONS

These three columns specify the timing consumption of each CPU for the type 6 CPUs.

All the numbers are in cycles.
NORM is the total number of cycles for the instruction
MOD is the total number of cycles for the instruction if it is modified
BUS is the number of cycles which are buscycles.

The CPU cycles are 250 ns each. The memory cycles vary with the memory type.

7. FUNCTION

This field shows the function of the instruction in an Algol-like manner. Below is given a list of terms:

Assignment is denoted by :=

The successor of an operand is:
.register: the register with number 1 higher (modulo 8)
.memory: the location with address 1 higher (modulo 64*1024)

The notation: OP1.[x:y] means a y-bit field of OP1 with the leftmost bit being bit x.

The notation: OP1 & OP2 [x:y:z] means that a z-bit field from OP2 with the leftmost bit being the bit y is inserted into OP1 with the leftmost bit inserted as bit x. The 16-z remaining bits of OP1 are left unchanged.

The notation: OP1 con OP2 means OP1 concatenated with OP2, i.e. a 32 bit operand where OP1 is the 16 most significant bits (31-16) and OP2 is the 16 least significant bits (15-0).

8. DESCRIPTION

This field describes the function of the instruction in plain English. Further, more special features about a certain instruction are explained.

The description field may be followed by a NOTICE and by a REMARK, which is fully instruction dependant.

9. INSTRUCTION CODES

In this column all possible formats of a single instruction are shown.

10. BINARY CODE

In this column the instruction bit lay out is specified and the placement of the variable parameter fields is shown.

The code may contain don't care bits (0) which indicate that both a zero and a one is valid. The HEXA value specifies which value the assembler puts into don't care bits.

11. HEXADECIMAL CODE

This column defines the fixed part of the instruction code. The variable fields specified in the BINARY code have to be added to the hexadecimal value before use.

****
********** MODIFY INSTRUCTIONS **********
****

THE MODIFY INSTRUCTIONS ALL HAVE ONE OPERAND, WHICH IS ADDED TO THE MODIFY REGISTER (MOD). FURTHER THE BOOLEAN, MODIFIED, IS SET TO TRUE.

A MODIFY INSTRUCTION CANNOT ITSELF BE MODIFIED. IF TWO OR MORE MODIFY INSTRUCTIONS FOLLOW EACH OTHER, THEIR FUNCTIONS ARE ACCUMULATED IN THE MODIFY REGISTER (MOD). HOWEVER WHEN TWO MODC INSTRUCTIONS FOLLOW EACH OTHER, THE RESULT IS THE SAME AS IF THE FIRST ONE MODIFIED THE SECOND ONE. NO INTERRUPT, EXCEPT TIME-OUT AND PARITY, IS SERVED BY THE CPU AS LONG AS THE BOOLEAN MODIFIED IS TRUE.

********** MODIFY WITH CONSTANT ******* 1 2 3 4 5 6 7 8

INSTRUCTION:
MODIFY WITH CONSTANT MODC

FORMAT:

		OP1	OP2	NORM	MOD	BUS
MODC	M	C8		3	3	1
MODC	M	CN8		3	3	1

FUNCTION:
```
MOD:= MOD+OP1;
MODIFIED:= TRUE;
```

DESCRIPTION:
ADDS THE OPERAND TO THE MODIFY REGISTER (MOD) AND SETS THE BOOLEAN, MODIFIED, TO TRUE.

NOTICE:
IF THE INSTRUCTION IS PRECEDED BY ANOTHER MODIFY INSTRUCTION, THIS HAS EFFECT ON THE MODIFY REGISTER (MOD), BUT NOT ON THE PRESENT INSTRUCTION. HOWEVER THIS MAKES NO DIFFERENCE BECAUSE THE INSTRUCTION OPERAND IS A CONSTANT. THE FINAL RESULT OF THIS CAN THUS BE THE SPLITTING OF ONE MODC INSTRUCTION INTO TWO, WHICH TOGETHER COVER THE FULL ADDRESSING ROOM.

CODE:

	OP1	OP2	BINARY	HEXA
MODC	C8			00E4
MODC	CN8			00A4

********** MODIFY WITH SHIFTED CONSTANT 1 2 3 4 5 6 7 8

INSTRUCTION:
MODIFY WITH CONSTANT, SHIFTED 4 MOD4
MODIFY WITH CONSTANT, SHIFTED 8 MOD8

FORMAT:

	OP1	OP2	NORM	MOD	BUS
MOD4	C8		5	5	1
MOD4	CN8		5	5	1
MOD8	C8		3	3	1

FUNCTION:
```
MOD:= MOD+OP1*2**SHIFTS;
MODIFIED:= TRUE;
```

DESCRIPTION:
ADDS THE LEFT SHIFTED OPERAND TO THE MODIFY REGISTER (MOD) AND SETS THE BOOLEAN, MODIFIED, TO TRUE.

CODE:

	OP1	OP2	BINARY	HEXA
MOD4	C8			00E5
MOD4	CN8			00A5
MOD8	C8			0056

********** MODIFY DIRECT ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
MODIFY [,NEGATIVE] MOD[N]

FORMAT:

	OP1	OP2	NORM	MOD	BUS
MOD[N]	B6.X2		3	3	2
MOD[N]	B8		3	3	2
MOD[N]	P8		3	3	2
MOD[N]	R3		3	3	1
MOD[N]	X3		4	4	2

FUNCTION:
```
OP:= OP1;
IF NEGATIVE THEN OP:= -OP;
MOD:= MOD+OP;
MODIFIED:= TRUE;
```

DESCRIPTION:
IF NEGATIVE MARKED, THE OPERAND IS NEGATED BEFORE USE. ADDS THE OPERAND TO THE MODIFY REGISTER (MOD) AND SETS THE BOOLEAN, MODIFIED, TO TRUE.

NOTICE:
IF THE INSTRUCTION IS PRECEDED BY ANOTHER MODIFY INSTRUCTION, THIS HAS EFFECT ON THE MODIFY REGISTER (MOD), BUT NOT ON THE PRESENT INSTRUCTION.

CODE:

	OP1	OP2	BINARY	HEXA
MOD	B6.X2			0054
MODN	B6.X2			0055
MOD	B8			0094
MODN	B8			0014
MOD	P8			0095
MODN	P8			0015
MOD	R3			70BD
MODN	R3			30BD
MOD	X3			78BD
MODN	X3			38BD

****
********** MOVE INSTRUCTIONS **********
****

THE MOVE INSTRUCTIONS PROVIDE THE USER WITH THE POSIBILITY OF MOVING OPERANDS SIZED FROM ONE BIT, BYTE, WORD, DOUBLE WORD, AND UP TO 16 WORDS IN ONE STEP. THE MOVE INSTRUCTION HAS TWO OPERANDS SPECIFIING THE MOVE DIRECTION:

- IMMEDIATE CONSTANT TO REGISTER
- IMMEDIATE CONSTANT TO MEMORY
- FROM REGISTER TO REGISTER
- FROM REGISTER TO MEMORY
- FROM MEMORY TO REGISTER
- FROM MEMORY TO MEMORY

********** MOVE CONSTANT ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
MOVE CONSTANT MOVC

FORMAT:

	OP1	OP2	NORM	MOD	BUS
MOVC	M C8	R3	3	3	1
MOVC	M CN8	R3	3	3	1
MOVC	M C4	X3	4	4	2
MOVC	M CN4	X3	4	4	2

FUNCTION:
OP2:= OP1;
CLEAR MODIFICATION;

DESCRIPTION:
MOVES A CONSTANT INTO OP2.

CODE:

	OP1	OP2	BINARY	HEXA
MOVC	C8	R3		0048
MOVC	CN8	R3		0028
MOVC	C4	X3		0050
MOVC	CN4	X3		0010

********** MOVE ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
MOVE MOV

FORMAT:

	OP1	OP2	NORM	MOD	BUS
MOV	M B8	R3	3	3	2
MOV	R3	M B8	5	5	2
MOV	R3	R3	2	2	1
MOV	R3	M B0.X3	3	3	2
MOV	X3	R3	2	2	2
MOV	X3	M B0.X3	5	5	3
MOV	M B6.X2	R3	3	4	2
MOV	R3	M B6.X2	4	5	2
MOV	M P8	R3	3	3	2

FUNCTION:
OP2:= OP1;
CLEAR MODIFICATION;

DESCRIPTION:
OP1 IS MOVED INTO OP2.

CODE:

	OP1	OP2	BINARY	HEXA
MOV	B8	R3		0068
MOV	R3	B8		0070
MOV	R3	R3		08BB
MOV	R3	B0.X3		88BB
MOV	X3	R3		08FB
MOV	X3	B0.X3		88FB
MOV	B6.X2	R3		0018
MOV	R3	B6.X2		0080
MOV	P8	R3		0098

********** MOVE BYTE ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
MOVE BYTE MOVB

FORMAT:

	OP1	OP2
MOVB *)	M BB6.X2	R2
MOVB **)	R2	M BB6.X2

NORM	MOD	BUS
5	5	2
5+L	6+L	2

(RIGHT BYTE: L=0)
(LEFT BYTE: L=1)

*) TYPE: LOAD
**) TYPE: STORE

FUNCTION:

```
CASE TYPE OF
  BEGIN
    LOAD:  OP2:= BYTEMEM[2*BASE+REG[X2]+BB6+2*MOD]&0[15:7:8];
    STORE: BYTEMEM[2*BASE+REG[X2]+BB6+2*MOD]:= OP1.[7:8];
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:
MOVES THE FIRST OPERAND INTO THE SECOND OPERAND. THE OPERANDS ARE BYTE VALUES. WHEN A BYTE IS MOVED TO A REGISTER, ITS VALUE IS PLACED IN THE LOWER BYTE, AND THE UPPER BYTE IS CLEARED. WHEN A BYTE IS MOVED FROM A REGISTER, THE LOWER BYTE IS USED, AND THE REGISTER IS UNCHANGED. IN MEMORY THE BYTE ADDRESSES ARE TAKEN RELATIVE TO THE BASE. THE CONTENT OF THE MODIFY REGISTER IS ADDED TO THE BASE REGISTER BEFORE THE ADDRESS IS CONVERTED TO A BYTEADDRESS. LOWER BYTES IN MEMORY HAVE EVEN BYTE ADDRESSES, UPPER BYTES HAVE ODD ADDRESSES ONE HIGHER THAN THE LOWER BYTE IN THE SAME MEMORY WORD.

CODE:

	OP1	OP2	BINARY	HEXA
MOVB	BB6.X2	R2		0078
MOVB	R2	BB6.X2		0090

********** MOVE LONG ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
MOVE LONG MOVL

FORMAT:

	OP1		OP2	NORM	MOD	BUS
MOVL	B6.X2		R2	4	4	3
MOVL	R2	M	B6.X2	6	7	3

FUNCTION:
OP2:= OP1;
SUCCESSOR(OP2):= SUCCESSOR(OP1);
CLEAR MODIFICATION;

DESCRIPTION:
MOVES OP1 AND ITS SUCCESSOR INTO OP2 AND ITS SUCCESSOR.

CODE:

	OP1	OP2	BINARY	HEXA
MOVL	B6.X2	R2		0060
MOVL	R2	B6.X2		00D0

********** MOVE PARAMETER ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
MOVE PARAMETER MVP

FORMAT:

		OP1	OP2	NORM	MOD	BUS
MVP	M	P6.X2	R2	3	4	2

FUNCTION:
OP2:= OP1;
CLEAR MODIFICATION;

DESCRIPTION:
MOVES OP1 INTO OP2.

CODE:

	OP1	OP2	BINARY	HEXA
MVP	P6.X2	R2		0020

********** MOVE MONITOR PARAMETER ***** * * * * * 6 7 8

INSTRUCTION:
MOVE MONITOR PARAMETER MMP

FORMAT:

	OP1	OP2	NORM	MOD	BUS
MMP	M P0.X3	R3	5+N (LEVEL= 1: N=1) (LEVEL<>1: N=0)	5+N	2

FUNCTION:
IF LEVEL=1 OR LEVEL=#8000 THEN
OP2:=OP1, FETCH FROM USER PROGRAM PAGE
ELSE
OP2:=OP1, FETCH FROM SYSTEM PROGRAM PAGE;
CLEAR MODIFICATION;

DESCRIPTION:
MOVES OP1 INTO OP2. IF LEVEL=1 OR LEVEL=#8000 THEN THE CURRENT MONITOR ROUTINE WAS CALLED BY AN INSTRUCTION IN THE USER PROGRAM PAGE, AND OP1 IS FETCHED FROM THAT PAGE.
IF LEVEL <> 0,1,#8000 THEN THE CURRENT MONITOR ROUTINE WAS CALLED BY AN INSTRUCTION IN SYSTEM PROGRAM PAGE, AND OP1 IS FETCHED FROM THAT PAGE.

CODE:

	OP1	OP2	BINARY	HEXA
MMP	M P0.X3	R3	|1|X3|0|R3|00010001|	8011

********** MOVE MULTIPLE ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
MOVE MULTIPLE MOVM

FORMAT:

	OP1	OP2	NORM	MOD	BUS
MOVM (M)	X3	X3	8+2*N	9+2*N	1+2*N
			(N=WORDS MOVED)		
			(UNM.: 4+2*16=36)		

FUNCTION:

```
N:= 1+(MOD-1) MODULO 16;
FOR I:= 1 STEP 1 UNTIL N DO
  BEGIN
    OP2:= OP1;
    INCR(INDEX1);
    INCR(INDEX2)
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:
MOVES A SEQUENCE OF 1 TO 16 WORDS IN MEMORY. THE TWO POINTERS, SPECIFIED AS INDEX REGISTERS, ARE INCREMENTED CORRESPONDINGLY, THUS AT THE END OF THE INSTRUCTION POINTING AT THE WORDS FOLLOWING THE LAST WORDS MOVED. THE NUMBER OF WORDS MOVED IS DEFINED BY THE MODIFY REGISTER (MOD) MODULO 16. HOWEVER A ZERO VALUE SPECIFIES MOVING OF 16 WORDS. THEREBY THE INSTRUCTION WHEN UNMODIFIED MOVES 16 WORDS.

NOTICE:
THE WORDS ARE MOVED ONE AT A TIME, SO OVERLAPPING MAY CAUSE REPETITION OF ONE OR MORE WORDS INSTEAD OF MOVING ALL WORDS.

CODE:

	OP1	OP2	BINARY	HEXA
MOVM	X3	X3		08E9

********** PUT MASKED ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
PUT MASKED PUT

FORMAT:

	OP1	OP2	NORM	MOD	BUS
PUT (M)	R3	X3	6	6	3
PUT (M)	X3	X3	7	7	4

FUNCTION:
OP2:= (OP2 AND (NOT MOD)) OR (OP1 AND MOD);
CLEAR MODIFICATION;

DESCRIPTION:
THE INSTRUCTION MOVES SOME BITS IN OP1 INTO THE CORRESPONDING BITS IN OP2. THE ACTIVE BITS ARE SELECTED BY ONES IN THE MODIFY REGISTER (MOD). THE REMAINING BITS IN OP2 ARE UNCHANGED.

CODE:

	OP1	OP2	BINARY	HEXA
PUT	R3	X3		88E9
PUT	X3	X3		88E8

********** EXCHANGE OPERANDS ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
EXCHANGE XCH

FORMAT:

	OP1		OP2	NORM	MOD	BUS
XCH	R3		R3	4	4	1
XCH	R3	M	B0.X3	5	5	3
XCH	X3		R3	4	4	3
XCH	X3	M	B0.X3	9	9	5

FUNCTION:
```
WORK:= OP2;
OP2:= OP1;
OP1:= WORK;
CLEAR MODIFICATION;
```

DESCRIPTION:
EXCHANGES THE TWO OPERANDS.

CODE:

	OP1	OP2	BINARY	HEXA
XCH	R3	R3		00EE
XCH	R3	B0.X3		08EE
XCH	X3	R3		80EF
XCH	X3	B0.X3		88EF

********** STACK INSTRUCTIONS ********* 1 2 3 4 5 6 7 8

INSTRUCTION:

STACK STC

FORMAT:

	OP1	OP2		NORM	MOD	BUS
STC *)		C4	(M)	11+N	11+N	1+N
STC **)	R3	M B0.X3		4	4	2
STC **)	X3	M B0.X3		5	5	3

(N=REGISTERS STACKED)

*) STACK REGISTERS
**) STACK VALUE

FUNCTION:

```
CASE INSTR OF
  BEGIN
  STACK REGISTERS:
    FOR I:= (15-C4) STEP -1 UNTIL 0 DO
      BEGIN
        REG[7]:= REG[7]-1;
        MEM[BASE+REG[7]+MOD]:= REG[I]
      END;
  STACK VALUE:
    BEGIN
      REG[X3(OP2)]:= REG[X3(OP2)]+1;
      MEM[REG[X3(OP2)]+BASE+MOD]:= OP1
    END;
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

STACKS ONE OR MORE REGISTERS IN A STACK. THE STACK POINTER ALWAYS POINTS AT THE STACK TOP, CONTAINING THE LAST STACKED VALUE. A POSSIBLE MODIFICATION WORKS AS A DISPLACEMENT TO THE STACK POINTER. THE FUNCTIONS ARE:

STACK REGISTERS: STACKS THE REGISTERS WITH THE NUMBERS 15-C4 THRU 0
INTO A BACKWARD STACK, I.E. THE STACK POINTER IS DECREMENTED BY THE STACK OPERATION. R7 IS USED AS STACK POINTER.

STACK VALUE: STACKS OP1 INTO A FORWARD STACK, I.E. THE STACK POINTER IS INCREMENTED BY THE STACK OPERATION.

CODE:

	OP1	OP2	BINARY	HEXA
STC		C4		E0BE
STC	R3	B0.X3		00B9
STC	X3	B0.X3		00F9

********** UNSTACK INSTRUCTIONS ******* 1 2 3 4 5 6 7 8

INSTRUCTION:

UNSTACK UNS

FORMAT:

	OP1	OP2		NORM	MOD	BUS
UNS *)		C4	(M)	6+N	6+N	1+N
UNS **)	R3	M B0.X3		4	4	2
UNS **)	X3	M B0.X3		5	5	3

(N=REGISTERS UNSTACKED)

*) UNSTACK REGISTERS
**) UNSTACK VALUE

FUNCTION:

```
CASE INSTR OF
  BEGIN
  UNSTACK REGISTERS:
  BEGIN
    CNT:= 15-C4;
    PTR:= REG[7];
    REG[7]:= REG[7]+CNT;
    FOR I:=0 STEP 1 UNTIL CNT DO
      REG[I]:= MEM[BASE+PTR+MOD+I];
  END;
  UNSTACK VALUE:
    BEGIN
      OP1:= MEM[REG[X3(OP2)]+BASE+MOD];
      REG[X3(OP2)]:= REG[X3(OP2)]-1
    END;
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

UNSTACKS ONE OR MORE REGISTERS FROM A STACK. THE STACK POINTER ALWAYS POINTS AT THE STACK TOP, CONTAINING THE NEXT VALUE TO BE UNSTACKED. A POSSIBLE MODIFICATION WORKS AS A DISPLACEMENT TO THE STACK POINTER. THE FUNCTIONS ARE:

UNSTACK REGISTERS: UNSTACKS THE REGISTERS WITH THE NUMBERS 0 THRU 15-C4 FROM A BACKWARD STACK, I.E. THE STACK POINTER IS INCREMENTED BY THE UNSTACK OPERATION. R7 IS USED AS STACK POINTER.

UNSTACK VALUE: UNSTACKS A VALUE INTO OP1 FROM A FORWARD STACK, I.E. THE STACK POINTER IS DECREMENTED BY THE UNSTACK OPERATION.

CODE:

	OP1	OP2	BINARY	HEXA
UNS		C4		A0BE
UNS	R3	B0.X3		08B9
UNS	X3	B0.X3		08F9

****
********** JUMP INSTRUCTIONS **********
****

JUMP INSTRUCTIONS ARE MAINLY UNCONDITIONAL. CONDITIONAL JUMPS SHOULD BE PERFORMED BY COMBINATIONS OF SKIP INSTRUCTIONS AND UNCONDITIONAL JUMPS. HOWEVER A FEW CONDITIONAL JUMP INSTRUCTIONS ARE INCLUDED IN ORDER TO SPEED UP THE MOST COMMON CASES.

********** JUMP RELATIVE ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
JUMP RELATIVE JMP

FORMAT:

	OP1	OP2	NORM	MOD	BUS
JMP		M L10	3	3	2
JMP		M LN10	3	3	2
JMP		M L8	3	4	2
JMP		M LN8	3	4	2
JMP	S2	M L8	3	4	2
JMP	S2	M LN8	3	4	2

FUNCTION:
IF RETURN THEN S2:= LOC+1-PROG;
LOC:= LOC+1+OP2+MOD;
CLEAR MODIFICATION;

DESCRIPTION:
JUMPS UNCONDITIONALLY TO THE LOCATION SPECIFIED BY OP2. IF A RETURN REGISTER IS SPECIFIED, THIS WILL POINT AT THE NEXT INSTRUCTION.

NOTICE:
THE MODIFY REGISTER IS ALWAYS ADDED TO LOC, ALSO IF THE OPERAND IS NEGATED BEFORE USE.

CODE:

	OP1	OP2	BINARY	HEXA
JMP		L10		00D8
JMP		LN10		0058
JMP		L8		00DC
JMP		LN8		005C
JMP	S2	L8		00DC
JMP	S2	LN8		005C

********** JUMP INDEXED ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
JUMP INDEXED JMP

FORMAT:

	OP1	OP2	NORM	MOD	BUS
JMP		M P6.X2	3	4	2

FUNCTION:
LOC:= REG[X2]+P6+MOD+PROG;
CLEAR MODIFICATION;

DESCRIPTION:
JUMPS UNCONDITIONALLY TO THE LOCATION SPECIFIED BY OP2.

REMARK:
THE INSTRUCTION IS MAINLY INTENDED FOR RETURN FROM A SUBROUTINE.

CODE:

	OP1	OP2	BINARY	HEXA
JMP		P6.X2		00BF

********** JUMP INDIRECT ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
JUMP INDIRECT JMPI

FORMAT:

	OP1	OP2	NORM	MOD	BUS
JMPI		M B6.X2	4	5	3
JMPI		M B8	5	5	3
JMPI		M P8	4	4	3
JMPI	S2	M P8	5	5	3

FUNCTION:
```
IF RETURN THEN S2:= LOC+1-PROG;
LOC:= OP2+PROG;
CLEAR MODIFICATION;
```

DESCRIPTION:
JUMPS UNCONDITIONALLY TO THE LOCATION SPECIFIED BY OP2. IF A RETURN REGISTER IS SPECIFIED, THIS WILL POINT AT THE NEXT INSTRUCTION.

CODE:

	OP1	OP2	BINARY	HEXA
JMPI		B6.X2		0012
JMPI		B8		00E7
JMPI		P8		00FC
JMPI	S2	P8		00FC

********** JUMP INDIRECT PAGE ZERO **** * * * * * 6 7 8

INSTRUCTION:
JUMP INDIRECT PAGE ZERO JPZI

FORMAT:

	OP1	OP2	NORM	MOD	BUS
JPZI	M	P8	8	8	3

FUNCTION:

```
LEVEL:= LEVEL&1[15:0:1];    % SET BIT 15
R4:=LOC+1-PROG;             % R4 FIXED RETURN REGISTER
PSW:=PSW&(0)[0:0:1];        % CLEAR BIT 0
LOC:= OP2+PROG:             % FETCH FROM USER PROGRAM PAGE
CLEAR MODIFICATION;
```

DESCRIPTION:
JUMPS UNCONDITIONALLY TO THE LOCATION IN PAGE ZERO SPECIFIED BY OP2. THE RETURN REGISTER (R4) POINTS TO THE NEXT INSTRUCTION.

CODE:

	OP1	OP2	BINARY		HEXA
JPZI	M	P8	P8	00010111	0017

********** RETURN FROM PAGE ZERO ****** * * * * * 6 7 8

INSTRUCTION:
RETURN FROM PAGE ZERO RPZ

FORMAT:

	OP1	OP2	NORM	MOD	BUS
RPZ	M	P6.X2	6	7	2

```
FUNCTION:
  LEVEL:= LEVEL&(0)[15:0:1];          % CLEAR BIT 15
  IF LEVEL=0 THEN
       PSW:= USER PROGRAM PAGE
       ELSE
       PSW:= SYSTEM PROGRAM PAGE;
  PSW:= PSW&(1)[0:0:1];               % SET BIT 0
  LOC:= REG[X2]+P6+MOD+PROG;
  CLEAR MODIFICATION;
```

DESCRIPTION:
IF LEVEL=0 A JUMP TO THE LOCATION IN THE USER PROGRAM PAGE SPECIFIED BY OP2 IS PERFORMED.IF LEVEL <> 0 THE JUMP IS EXCECUTED IN SYSTEM PROGRAM PAGE.

CODE:

	OP1	OP2	BINARY			HEXA
RPZ	M	P6.X2	|X2|	P6	|10100111|	00A7

********** JUMP ON OPERAND ********** 1 2 3 4 5 6 7 8

INSTRUCTION:

JUMP IF OPERAND NONZERO	JON
JUMP IF OPERAND ZERO	JOZ

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<INSTR>	R3	L4	4+2*J	4+2*J	1+J
<INSTR>	R3	LN4	4+2*J	4+2*J	1+J
<INSTR>	X3	L4	4+2*J	4+2*J	2+J
<INSTR>	X3	LN4	4+2*J	4+2*J	2+J

FUNCTION:

```
IF CASE INSTR OF
    (JON: OP1<>0,
     JOZ: OP1= 0
    ) THEN
  LOC:= LOC+1+OP2
ELSE
  LOC:= LOC+1;
COMMENT: LOC IS UPDATED ACCORDING TO LOCATIONS OCCUPIED;
```

DESCRIPTION:

IF THE INSTRUCTION CONDITION IS FULFILLED, A JUMP IS PERFORMED TO THE LOCATION SPECIFIED BY OP2. THE CONDITIONS ARE:

- JON: THE OPERAND IS NONZERO.
- JOZ: THE OPERAND IS ZERO.

CODE:

	OP1	OP2	BINARY	HEXA
JON	R3	L4		80F0
JON	R3	LN4		80B0
JON	X3	L4		80F1
JON	X3	LN4		80B1
JOZ	R3	L4		80F2
JOZ	R3	LN4		80B2
JOZ	X3	L4		80F3
JOZ	X3	LN4		80B3

********** JUMP ON OVERFLOW ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
JUMP IF OVERFLOW JVN

FORMAT:

	OP1	OP2	NORM	MOD	BUS
JVN		M L4	5+2*J	5+2*J	1+2*J
JVN		M LN4	4+3*J	4+3*J	1+2*J
			(NO JUMP: J=0)		
			(JUMP: J=1)		

FUNCTION:
```
IF PSW.[V:1]=0 THEN   LOC:= LOC+1+OP2
ELSE                  LOC:= LOC+1;
CLEAR MODIFICATION;
```

DESCRIPTION:
IF THE OVERFLOW BIT IN PSW IS RESET, A JUMP IS PERFORMED TO THE LOCATION SPECIFIED BY OP2. THAT IS, THE JUMP IS PERFORMED WHEN AN OVERFLOW CONDITION EXISTS.

CODE:

	OP1	OP2	BINARY	HEXA
JVN		L4		F0BE
JVN		LN4		70BE

********** LOOP CONTROL ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
SUBTRACT ONE, BRANCH IF NONZERO SOB

FORMAT:

	OP1	OP2	NORM	MOD	BUS
SOB	R3	M LN8	4+N	4+N	1+N

(ZERO: N=0)
(NONZERO: N=1)

FUNCTION:
```
OP1:= OP1-1;
IF OP1<>0 THEN   LOC:= LOC+1+OP2
ELSE             LOC:= LOC+1;
CLEAR MODIFICATION;
```

DESCRIPTION:
SUBTRACTS ONE FROM OP1. IF OP1 THEN IS NONZERO, A JUMP IS PERFORMED TO THE LOCATION SPECIFIED BY OP2.

CODE:

	OP1	OP2	BINARY	HEXA
SOB	R3	LN8		00C0

****
********** ARITHMETIC AND LOGIC INSTRUCTIONS **********
****

THE ARITHMETIC AND LOGIC INSTRUCTIONS COVER THE FULL RANGE OF ARITHMETIC AND LOGIC OPERATIONS, INCLUDING MULTIPLE WORD OPERATIONS. IN GENERAL THE OPERATIONS MAY BE PERFORMED EITHER ON REGISTERS OR DIRECTLY IN MEMORY USING AN INDEX REGISTER.

********** ADD CONSTANT ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
ADD CONSTANT ADDC

FORMAT:

	OP1	OP2	NORM	MOD	BUS
ADDC	M C8	R3	3	3	1
ADDC	M CN8	R3	3	3	1

FUNCTION:
OP1:= OP1+MOD (2'S COMPLEMENT)
OP2:= OP2+OP1;
SET CONDITION CODES;
CLEAR MODIFICATION;

DESCRIPTION:
ADDS AN IMMEDIATE CONSTANT TO A REGISTER.

CODE:

	OP1	OP2	BINARY	HEXA
ADDC	C8	R3		00C8
ADDC	CN8	R3		00A8

********** ADD ********** 1 2 3 4 5 6 7 8

INSTRUCTION:

ADD [,UPDATE WITH CARRY] ADD[U]

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<INSTR>	R3	R3	3	3	1
<INSTR>	R3	M B0.X3	5	5	3
<INSTR>	X3	R3	4	4	2
<INSTR>	X3	M B0.X3	6	6	4
ADD	M B6.X2	R3	4	5	2
ADD	R3	M B6.X2	5	6	3

FUNCTION:

```
CASE INSTR OF
  BEGIN
    ADD:  OP2:= OP2+OP1;
    ADDU: OP2:= OP2+OP1+PSW.[C:1];
  END;
CLEAR MODIFICATION;
SET CONDITION CODES;
```

DESCRIPTION:

THE ARITHMETIC OPERATION IS PERFORMED ON THE TWO OPERANDS. THE RESULT IS DELIVERED IN THE SECOND OPERAND. THE OPERATIONS ARE USING TWO'S COMPLEMENT ARITHMETIC. THEY ARE:

ADD: OP1 IS ADDED TO OP2.

ADDU: OP1 AND THE CARRY ARE ADDED TO OP2.

INSTRUCTIONS UPDATING WITH CARRY WORK AS REQUIRED FOR MULTI-LENGTH CALCULATIONS.

CODE:

	OP1	OP2	BINARY	HEXA
ADD	R3	R3		008A
ADD	R3	B0.X3		088A
ADD	X3	R3		808A
ADD	X3	B0.X3		888A
ADD	B6.X2	R3		0030
ADD	R3	B6.X2		0008
ADDU	R3	R3		00D4
ADDU	R3	B0.X3		08D4
ADDU	X3	R3		80D4
ADDU	X3	B0.X3		88D4

********** SUBTRACT ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
SUBTRACT [,UPDATE WITH CARRY] SUB[U]

FORMAT:

	OP1		OP2	NORM	MOD	BUS
<INSTR>	R3		R3	3	3	1
<INSTR>	R3	M	B0.X3	5	5	3
<INSTR>	X3		R3	4	4	2
<INSTR>	X3	M	B0.X3	7	7	4

FUNCTION:

```
CASE INSTR OF
  BEGIN
    SUB:  OP2:= OP2-OP1;
    SUBU: OP2:= OP2-OP1+PSW.[C:1]-1;
  END;
CLEAR MODIFICATION;
SET CONDITION CODES;
```

DESCRIPTION:
THE ARITHMETIC OPERATION IS PERFORMED ON THE TWO OPERANDS. THE RESULT IS DELIVERED IN THE SECOND OPERAND. THE OPERATIONS ARE USING TWO'S COMPLEMENT ARITHMETIC. THEY ARE:
SUB: OP1 IS SUBTRACTED FROM OP2.
SUBU: OP1 AND THE BORROW ARE SUBTRACTED FROM OP2.
INSTRUCTIONS UPDATING WITH BORROW WORK AS REQUIRED FOR MULTI-LENGTH CALCULATIONS.

CODE:

	OP1	OP2	BINARY	HEXA
SUB	R3	R3		008B
SUB	R3	B0.X3		088B
SUB	X3	R3		808B
SUB	X3	B0.X3		888B
SUBU	R3	R3		00D5
SUBU	R3	B0.X3		08D5
SUBU	X3	R3		80D5
SUBU	X3	B0.X3		88D5

********** MULTIPLY ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
MULTIPLY MUL

FORMAT:

	OP1	OP2	NORM	MOD	BUS
MUL	R3	M B0.X3	27	28	4
MUL	X3	M B0.X3	27	28	5

FUNCTION:
DEFINE OP3::= SUCCESSOR(OP2) CON OP2;
OP3:= OP1*OP2;
CLEAR MODIFICATION;

DESCRIPTION:
OP1 AND OP2 ARE MULTIPLIED. THE RESULT IS STORED AS A THIRTYTWO-BIT TWO'S COMPLEMENT NUMBER INTO THE CONCATENATION OF THE SUCCESSOR OF OP2 AND OP2 ITSELF.

CODE:

	OP1	OP2	BINARY	HEXA
MUL	R3	B0.X3		08EA
MUL	X3	B0.X3		88EA

********** DIVIDE ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
DIVIDE DIV

FORMAT:

	OP1	OP2	NORM	MOD	BUS
DIV	R3	R3	41	41	1
DIV	X3	R3	41	41	2

FUNCTION:
```
DEFINE OP3::= SUCCESSOR(OP2) CON OP2;
OP2:= OP3 DIV OP1;
SUCCESSOR(OP2):= OP3 REM OP1;
CLEAR MODIFICATION;
```

DESCRIPTION:
THE SUCCESSOR OF OP2 CONCATENATED WITH OP2 ITSELF IS TAKEN AS AN UNSIGNED THIRTYTWO-BIT NUMBER THAT IS DIVIDED BY OP1. THE SIXTEEN LEAST SIGNIFICANT BITS OF THE RESULT ARE STORED INTO OP2, AND THE REMAINDER OF THE DIVISION IS STORED INTO THE SUCCESSOR OF OP2.

NOTE:
NO INDICATION IS GIVEN IF AN OVERFLOW CONDITION OCCURS.

CODE:

	OP1	OP2	BINARY	HEXA
DIV	R3	R3		00EB
DIV	X3	R3		80EB

********** LOGICAL DYADIC INSTRUCTIONS 1 2 3 4 5 6 7 8

INSTRUCTION:

LOGICAL AND	AND
LOGICAL INCLUSIVE OR	IOR
LOGICAL EXCLUSIVE OR	XOR

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<INSTR>	R3	R3	3	3	1
<INSTR> *)	R3	M B0.X3	5	5	3
<INSTR>	X3	R3	3	3	2
<INSTR> *)	X3	M B0.X3	6	6	4

*) ONLY AND, IOR.

FUNCTION:

```
CASE INSTR OF
  BEGIN
    AND: OP2:= OP2 AND OP1;
    IOR: OP2:= OP2 OR OP1;
    XOR: OP2:= (OP2 OR OP1)-(OP2 AND OP1);
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

THE LOGICAL DYADIC OPERATION IS PERFORMED ON THE TWO OPERANDS. THE RESULT IS DELIVERED IN THE SECOND OPERAND. THE TRUTH-TABLES ARE:

AND	IOR	XOR
0,0: 0	0,0: 0	0,0: 0
0,1: 0	0,1: 1	0,1: 1
1,0: 0	1,0: 1	1,0: 1
1,1: 1	1,1: 1	1,1: 0

CODE:

	OP1	OP2	BINARY	HEXA
AND	R3	R3		008C
AND	R3	B0.X3		088C
AND	X3	R3		808C
AND	X3	B0.X3		888C
IOR	R3	R3		008D
IOR	R3	B0.X3		088D
IOR	X3	R3		808D
IOR	X3	B0.X3		888D
XOR	R3	R3		00E8
XOR	X3	R3		80E8

********** MONADIC INSTRUCTIONS ******* 1 2 3 4 5 6 7 8

INSTRUCTION:

CLEAR	CLR
DECREMENT	DEC
INCREMENT	INC
INVERT	INV
NEGATE	NEG
SWAP	SWP

FORMAT:

	OP1	OP2	NORM	MOD	BUS
CLR		M B6.X2	4	5	3
DEC		M B6.X2	4	5	3
INC		M B6.X2	4	5	3
INV		R3	3	4	1
INV		M B0.X3	6	7	3
NEG		R3	3	4	1
NEG		M B0.X3	6	7	3
SWP		R3	3	3	1
SWP		M B0.X3	5	5	3

FUNCTION:

```
CASE INSTR OF
  BEGIN
    CLR: OP2:= 0;
    DEC: OP2:= OP2-1;
    INC: OP2:= OP2+1;
    INV: OP2:= -1-OP2;
    NEG: OP2:= -OP2;
    SWP: OP2:= OP2.[15:8]&OP2[15:7:8];
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

THE OPERAND IS ASSIGNED A VALUE DEPENDING ON THE INSTRUCTION:

- CLR: ZERO.
- DEC: THE OPERAND DECREMENTED BY ONE.
- INC: THE OPERAND INCREMENTED BY ONE.
- INV: ONE'S COMPLEMENT OF THE OPERAND.
- NEG: TWO'S COMPLEMENT OF THE OPERAND.
- SWP: THE TWO BYTES IN THE OPERAND EXCHANGED.

CODE:

	OP1	OP2	BINARY	HEXA
CLR		B6.X2		0027
DEC		B6.X2		0013
INC		B6.X2		0016
INV		R3		90BC
INV		B0.X3		98BC
NEG		R3		F0BC
NEG		B0.X3		F8BC
SWP		R3		60BC
SWP		B0.X3		68BC

********** DOUBLE UPDATE INSTRUCTIONS * 1 2 3 4 5 6 7 8

INSTRUCTION:

DECREMENT DOUBLE	DECD
INCREMENT DOUBLE	INCD

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<INSTR>	R3	R3	4	4	1
<INSTR>	X3	M B0.X3	8	8	5

FUNCTION:

```
WORK1:= OP1; WORK2:= OP2;
CASE INSTR OF
  BEGIN
    DECD: BEGIN OP1:= WORK1-1; OP2:= WORK2-1; END;
    INCD: BEGIN OP1:= WORK1+1; OP2:= WORK2+1; END;
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

DECREMENTS OR INCREMENTS THE TWO OPERANDS OF THE INSTRUCTION.

NOTICE:

IF THE TWO OPERANDS ARE THE SAME, IT IS ONLY UPDATED ONCE.

CODE:

	OP1	OP2	BINARY	HEXA
DECD	R3	R3		088E
DECD	X3	B0.X3		888F
INCD	R3	R3		008E
INCD	X3	B0.X3		808F

********** PAIR UPDATE INSTRUCTIONS *** 1 2 3 4 5 6 7 8

INSTRUCTION:

DOUBLE DECREMENT, PAIR	DDCP
DOUBLE INCREMENT, PAIR	DICP

FORMAT:

	OP1	OP2	NORM	MOD	BUS
DDCP	R3		5	5	1
DDCP	M B0.X3		8	8	5
DICP	R3		5	5	1
DICP	M B0.X3		8	8	5

FUNCTION:

```
DEFINE OP2::= SUCCESSOR(OP1);
CASE INSTR OF
  BEGIN
    DDCP: BEGIN OP1:= OP1-2; OP2:= OP2-2; END;
    DICP: BEGIN OP1:= OP1+2; OP2:= OP2+2; END;
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

DECREMENTS OR INCREMENTS THE OPERAND AND ITS SUCCESSOR TWICE.

CODE:

	OP1	OP2	BINARY	HEXA
DDCP	R3			30BC
DDCP	B0.X3			38BC
DICP	R3			20BC
DICP	B0.X3			28BC

********** SIGN EXTENSION ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
SIGN EXTENSION SXT

FORMAT:

	OP1	OP2	NORM	MOD	BUS
SXT	X3	(M)C4	7	8	3

FUNCTION:

```
BIT:= IF MODIFIED THEN   MOD MODULO 16
    ELSE                          OP2;
OP1:= OP1.[BIT:BIT+1];
IF OP1.[BIT:1]=1 THEN OP1:= OP1&(-1)[15:15:15-BIT];
CLEAR MODIFICATION;
```

DESCRIPTION:
EXTENDS A BIT IN OP1 AS A SIGN BIT. THE BIT NUMBER IS DEFINED BY OP2 OR, IF MODIFIED, BY THE MODIFY REGISTER (MOD) MODULO 16.

CODE:

	OP1	OP2	BINARY	HEXA
SXT	X3	C4		0067

********** EXTRACT BIT GROUP ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
EXTRACT XTR

FORMAT:

	OP1	OP2	NORM	MOD	BUS
XTR	R3	C4	3	3	1
XTR	X3	C4	5	5	3

FUNCTION:
```
IF OP2=0 THEN OP1:= 0
ELSE          OP1:= OP1.[BITS-1:BITS];
CLEAR MODIFICATION;
```

DESCRIPTION:
EXTRACTS A GROUP OF LEAST SIGNIFICANT BITS IN OP1. THE NUMBER OF BITS IS DEFINED BY OP2.

REMARK:
IF THE SECOND OPERAND SPECIFIES ZERO BITS, THE FIRST OPERAND IS CLEARED.

CODE:

	OP1	OP2	BINARY	HEXA
XTR	R3	C4		00EC
XTR	X3	C4		80ED

****
********** SKIP INSTRUCTIONS **********
****

THE SKIP INSTRUCTIONS ARE DIVIDED INTO THE GROUPS:
- SKIP ON BIT
- SKIP ON OPERAND
- SKIP ON CONDITION

THE INSTRUCTIONS HAVE SOME COMMON VARIATIONS. GENERALLY THERE ARE ONLY 5 INSTRUCTIONS: SBN, SON, SNE, SGE, SHS. EACH OF THESE 5 INSTRUCTIONS FURTHER EXISTS IN A VERSION SKIPPING ON THE COMPLEMENTARY CONDITION. EACH OF THESE 10 INSTRUCTIONS FURTHER EXISTS IN A VERSION SKIPPING TWO LOCATIONS INSTEAD OF ONE.

SOME COMBINATIONS OF SKIP AND JUMP INSTRUCTIONS CAN BE REPLACED BY A CONDITIONAL JUMP INSTRUCTION.

********** SKIP ON BIT ********** 1 2 3 4 5 6 7 8

INSTRUCTION:

SKIP IF BIT NONZERO [,PAIR]	SBN[P]
SKIP IF BIT ZERO [,PAIR]	SBZ[P]

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<SKIP>	R3	(M)C4	4+S	5+S	1+S
<SKIP>	X3	(M)C4	5+S	6+S	2+S
<SKIPPAIR>	R3	(M)C4	4+2*S	6+2*S	1+2*S
<SKIPPAIR>	X3	(M)C4	5+2*S	6+2*S	2+2*S
			(NO SKIP: S=0)		
			(SKIP: S=1)		

FUNCTION:

```
IF MODIFIED THEN DEFINE OP2::= MOD MODULO 16;
IF CASE INSTR OF
    (SBN[P]: OP1.[OP2:1]=1,
     SBZ[P]: OP1.[OP2:1]=0
    ) THEN
  IF PAIR THEN LOC:= LOC+3    % SKIP 2
  ELSE         LOC:= LOC+2    % SKIP 1
ELSE           LOC:= LOC+1;   % SKIP 0
CLEAR MODIFICATION;
```

DESCRIPTION:

INSPECTS A BIT IN OP1. THE BIT NUMBER IS SPECIFIED BY OP2 OR, IF MODIFIED, BY THE MODIFY REGISTER (MOD) MODULO 16. IF THE CONDITION OF THE INSTRUCTION IS FULFILLED, A SKIP IS PERFORMED. IF PAIR IS FALSE, ONE MEMORY LOCATION IS SKIPPED, IF PAIR IS TRUE, TWO MEMORY LOCATIONS ARE SKIPPED. THE CONDITIONS ARE:

SBN[P]: THE BIT IS NONZERO.
SBZ[P]: THE BIT IS ZERO.

CODE:

	OP1	OP2	BINARY	HEXA
SBN	R3	C4		00F0
SBN	X3	C4		00F1
SBZ	R3	C4		00F2
SBZ	X3	C4		00F3
SBNP	R3	C4		00B0
SBNP	X3	C4		00B1
SBZP	R3	C4		00B2
SBZP	X3	C4		00B3

********** SKIP ON OPERAND ********** 1 2 3 4 5 6 7 8

INSTRUCTION:

SKIP IF OPERAND NONZERO [,PAIR]	SON[P]
SKIP IF OPERAND ZERO [,PAIR]	SOZ[P]

FORMAT:

		OP1	OP2	NORM	MOD	BUS
<SKIP>	M	B6.X2		4+S	5+S	2+S
<SKIPPAIR>	M	B6.X2		4+2*S	5+2*S	2+S
				(NO SKIP: S=0)		
				(SKIP: S=1)		

FUNCTION:

```
IF CASE INSTR OF
    (SON[P]: OP1<>0,
     SOZ[P]: OP1=0
    ) THEN
  IF PAIR THEN LOC:= LOC+3    % SKIP 2
  ELSE         LOC:= LOC+2    % SKIP 1
ELSE           LOC:= LOC+1;   % SKIP 0
CLEAR MODIFICATION;
```

DESCRIPTION:

IF THE CONDITION OF THE INSTRUCTION IS FULFILLED, A SKIP IS PERFORMED. IF PAIR IS FALSE, ONE MEMORY LOCATION IS SKIPPED, IF PAIR IS TRUE, TWO MEMORY LOCATIONS ARE SKIPPED.
THE CONDITIONS ARE:

- SON[P]: THE OPERAND IS NONZERO.
- SOZ[P]: THE OPERAND IS ZERO.

CODE:

	OP1	OP2	BINARY	HEXA
SON	B6.X2			003B
SOZ	B6.X2			003A
SONP	B6.X2			0039
SOZP	B6.X2			0038

********** SKIP ON CONDITION ********** 1 2 3 4 5 6 7 8

INSTRUCTION:

SKIP IF EQUAL [,PAIR]	SEQ[P]
SKIP IF NOT EQUAL [,PAIR]	SNE[P]
SKIP IF GREATER OR EQUAL [,PAIR]	SGE[P]
SKIP IF LESS THAN [,PAIR]	SLT[P]
SKIP IF HIGHER OR SAME [,PAIR]	SHS[P]
SKIP IF LOWER [,PAIR]	SLO[P]

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<INSTR>	R3	M C4	3+2*S	4+2*S	1+S
<INSTR> *)	R3	M CN4	3+2*S	4+2*S	1+S
<INSTR>	X3	M C4	4+2*S	5+2*S	2+S
<INSTR> *)	X3	M CN4	4+2*S	5+2*S	2+S
<SKIP>	R3	R3	3+S	3+S	1+S
<SKIP>	R3	M B0.X3	5+S	5+S	2+S
<SKIP>	X3	R3	3+S	3+S	2+S
<SKIP>	X3	M B0.X3	6+S	6+S	3+S
<SKIPPAIR>	R3	R3	3+2*S	3+2*S	1+S
<SKIPPAIR>	R3	M B0.X3	5+2*S	5+2*S	2+S
<SKIPPAIR>	X3	R3	3+2*S	3+2*S	2+S
<SKIPPAIR>	X3	M B0.X3	6+2*S	6+2*S	3+S
			(NO SKIP: S=0)		
*) ONLY SEQ[P], SNE[P].			(SKIP: S=1)		

FUNCTION:

```
IF CASE INSTR OF
    (SEQ[P]: OP1= OP2,
     SNE[P]: OP1<>OP2,
     SGE[P]: OP1>=OP2, % SIGNED
     SLT[P]: OP1< OP2, % SIGNED
     SHS[P]: OP1>=OP2, % UNSIGNED
     SLO[P]: OP1< OP2  % UNSIGNED
    ) THEN
  IF PAIR THEN LOC:= LOC+3    % SKIP 2
  ELSE         LOC:= LOC+2    % SKIP 1
ELSE           LOC:= LOC+1;   % SKIP 0
CLEAR MODIFICATION;
```

DESCRIPTION:

CALCULATES THE CONDITION SPECIFIED BY THE INSTRUCTION. IF FULFILLED, A SKIP IS PERFORMED. IF PAIR IS FALSE, ONE MEMORY LOCATION IS SKIPPED, IF PAIR IS TRUE, TWO MEMORY LOCATIONS ARE SKIPPED. THE CONDITIONS ARE:

- SEQ[P]: THE TWO OPERANDS ARE EQUAL.
- SNE[P]: THE TWO OPERANDS ARE NOT EQUAL.
- SGE[P]: OP1 IS GREATER THAN OR EQUAL TO OP2, SIGNED.
- SLT[P]: OP1 IS LESS THAN OP2, SIGNED.
- SHS[P]: OP1 IS GREATER THAN OR EQUAL TO OP2, UNSIGNED.
- SLO[P]: OP1 IS LESS THAN OP2, UNSIGNED.

REMARK:
SOME COMBINATIONS OF SKIP AND JUMP INSTRUCTIONS CAN BE BETTER DONE BY MEANS OF THE CONDITIONAL JUMP INSTRUCTIONS.

CODE:

	OP1	OP2	BINARY	HEXA
SEQ	R3	C4		807E
SEQ	R3	CN4		007E
SEQ	X3	C4		807F
SEQ	X3	CN4		007F
SEQ	R3	R3		00FA
SEQ	R3	BO.X3		08FA
SEQ	X3	R3		80FA
SEQ	X3	BO.X3		88FA
SNE	R3	C4		807C
SNE	R3	CN4		007C
SNE	X3	C4		807D
SNE	X3	CN4		007D
SNE	R3	R3		00F8
SNE	R3	BO.X3		08F8
SNE	X3	R3		80F8
SNE	X3	BO.X3		88F8
SGE	R3	C4		8042
SGE	X3	C4		8043
SGE	R3	R3		00F6
SGE	X3	R3		80F6
SGE	R3	X3		08F6
SGE	X3	X3		88F6
SLT	R3	C4		8040
SLT	X3	C4		8041
SLT	R3	R3		00F4
SLT	X3	R3		80F4
SLT	R3	X3		08F4
SLT	X3	X3		88F4
SHS	R3	C4		8046
SHS	X3	C4		8047
SHS	R3	R3		00F7
SHS	X3	R3		80F7
SHS	R3	X3		08F7
SHS	X3	X3		88F7
SLO	R3	C4		8044
SLO	X3	C4		8045
SLO	R3	R3		00F5
SLO	X3	R3		80F5
SLO	R3	X3		08F5
SLO	X3	X3		88F5

SEQP	R3	C4	803E
SEQP	R3	CN4	003E
SEQP	X3	C4	803F
SEQP	X3	CN4	003F
SEQP	R3	R3	00BA
SEQP	R3	BO.X3	08BA
SEQP	X3	R3	80BA
SEQP	X3	BO.X3	88BA
SNEP	R3	C4	803C
SNEP	R3	CN4	003C
SNEP	X3	C4	803D
SNEP	X3	CN4	003D
SNEP	R3	R3	00B8
SNEP	R3	BO.X3	08B8
SNEP	X3	R3	80B8
SNEP	X3	BO.X3	88B8
SGEP	R3	C4	8002
SGEP	X3	C4	8003
SGEP	R3	R3	00B6
SGEP	X3	R3	80B6
SGEP	R3	X3	08B6
SGEP	X3	X3	88B6
SLTP	R3	C4	8000
SLTP	X3	C4	8001
SLTP	R3	R3	00B4
SLTP	X3	R3	80B4
SLTP	R3	X3	08B4
SLTP	X3	X3	88B4
SHSP	R3	C4	8006
SHSP	X3	C4	8007
SHSP	R3	R3	00B7
SHSP	X3	R3	80B7
SHSP	R3	X3	08B7
SHSP	X3	X3	88B7
SLOP	R3	C4	8004
SLOP	X3	C4	8005
SLOP	R3	R3	00B5
SLOP	X3	R3	80B5
SLOP	R3	X3	08B5
SLOP	X3	X3	88B5

****
********** SHIFT INSTRUCTIONS **********
****

THE SHIFT INSTRUCTIONS ARE DIVIDED INTO THE GROUPS:
- SHIFT SINGLE
- SHIFT LONG

THE TWO GROUPS ARE WORKING ON A SINGLE WORD OPERAND AND A DOUBLE WORD OPERAND RESPECTIVELY. A DOUBLE WORD OPERAND IS THE SUCCESSOR OF THE OPERAND CONCATENATED WITH THE OPERAND. THE SUCCESSOR BEING THE NEXT REGISTER OR THE NEXT MEMORY LOCATION.

LEFT SHIFTS AND RIGHT SHIFTS ARE PERFORMED BY DIFFERENT INSTRUCTIONS. THIS HAS THE CONSEQUENCE THAT A SHIFT INSTRUCTION CANNOT BE MODIFIED TO CHANGE SHIFT DIRECTION.

IF A SHIFT INSTRUCTION IS PRECEDED BY A MODIFY INSTRUCTION THE NUMBER OF BIT POSITIONS SHIFTED IS DEFINED BY THE MODIFY REGISTER (MOD) MODULO 16,THUS IGNORING THE SECOND OPERAND OF THE SHIFT INSTRUCTION. IF UNMODIFIED AND THE SECOND OPERAND SPECIFIES ZERO SHIFTS, 16 SHIFTS ARE PERFORMED.

********** SHIFT SINGLE ********** 1 2 3 4 5 6 7 8

INSTRUCTION:

SHIFT LEFT CYCLICALLY	SLC
SHIFT RIGHT ARITHMETICALLY	SRA
SHIFT LEFT LOGICALLY	SLL
SHIFT RIGHT LOGICALLY	SRL

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<INSTR>	R3	(M)C4	4+S	6+S	1
<INSTR>	X3	(M)C4	5+S	7+S	3
			(S=SHIFTS)		

FUNCTION:

```
SHIFTS:= IF MODIFIED THEN   MOD MODULO 16
    ELSE IF C4=0 THEN                  16
    ELSE                               C4;
FOR I:= 1 STEP 1 UNTIL SHIFTS DO
  CASE INSTR OF
    BEGIN
      SLC: OP1:= OP1.[15:1]&OP1[15:14:15];
      SRA: OP1:=        OP1&OP1[14:15:15];
      SLL: OP1:=          0&OP1[15:14:15];
      SRL: OP1:=          0&OP1[14:15:15];
    END;
CLEAR MODIFICATION;
```

DESCRIPTION:

OP1 IS SHIFTED AS MANY BIT POSITIONS AS SPECIFIED BY OP2 OR THE MODIFY REGISTER. IF UNMODIFIED AND ZERO SHIFTS ARE SPECIFIED 16 SHIFTS ARE PERFORMED. THE SHIFTS ARE PERFORMED AS INDICATED IN THE INSTRUCTION:

- SLC: SHIFT LEFT CYCLICALLY. THE 16 BIT OPERAND IS ROTATED LEFT. NO CARRY OR OTHER EXTERNAL BITS ARE INCORPORATED.
- SRA: SHIFT RIGHT ARITHMETICALLY. THE 16 BIT OPERAND IS SHIFTED RIGHT WITH SIGN EXTENSION IN THE UPPER BITS.
- SLL: SHIFT LEFT LOGICALLY. THE 16 BIT OPERAND IS SHIFTED LEFT WITH ZERO EXTENSION IN THE LOWER BITS.
- SRL: SHIFT RIGHT LOGICALLY. THE 16 BIT OPERAND IS SHIFTED RIGHT WITH ZERO EXTENSION IN THE UPPER BITS.

NOTICE:

THE CONDITION CODES ARE UNCHANGED.

CODE:

	OP1	OP2	BINARY	HEXA
SLC	R3	C4		80A2
SLC	X3	C4		80E2
SRA	R3	C4		8026
SRA	X3	C4		8066
SLL	R3	C4		80A0
SLL	X3	C4		80E0
SRL	R3	C4		8024
SRL	X3	C4		8064

********** SHIFT LONG ********** 1 2 3 4 5 6 7 8

INSTRUCTION:

SHIFT LEFT LOGICALLY, LONG	SLLL
SHIFT RIGHT LOGICALLY, LONG	SRLL

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<INSTR>	R3	(M)C4	7+S	9+S	1
<INSTR>	X3	(M)C4	9+S	11+S	5
			(S=SHIFTS)		

FUNCTION:

```
SHIFTS:= IF MODIFIED THEN   MOD MODULO 16
    ELSE IF C4=0 THEN                  16
    ELSE                               C4;
DEFINE OP1::= SUCCESSOR(OP1) CON OP1;
FOR I:= 1 STEP 1 UNTIL SHIFTS DO
  CASE INSTR OF
    BEGIN
      SLLL: OP1:= 0&OP1[31:30:31];
      SRLL: OP1:= 0&OP1[30:31:31];
    END;
CLEAR MODIFICATION;
```

DESCRIPTION:

THE SUCCESSOR OF OP1 CONCATENATED WITH OP1 IS THE OPERAND SHIFTED. THE NUMBER OF BIT POSITIONS SHIFTED IS DEFINED BY OP2 OR THE MODIFY REGISTER. IF UNMODIFIED AND ZERO SHIFTS ARE SPECIFIED 16 SHIFTS ARE PERFORMED. THE SHIFTS ARE PERFORMED AS INDICATED IN THE INSTRUCTION:

- SLLL: SHIFT LEFT LOGICALLY. THE 32 BIT OPERAND IS SHIFTED LEFT WITH ZERO EXTENTION IN THE LOWER BITS.
- SRLL: SHIFT RIGHT LOGICALLY. THE 32 BIT OPERAND IS SHIFTED RIGHT WITH ZERO EXTENSION IN THE UPPER BITS.

CODE:

	OP1	OP2	BINARY	HEXA
SLLL	R3	C4		00A1
SLLL	X3	C4		00E1
SRLL	R3	C4		0025
SRLL	X3	C4		0065

****
********** SINGLE BIT OPERATIONS **********
****

THESE INSTRUCTIONS WORK ON TWO OPERANDS. THE FIRST DEFINES A WORD, THE SECOND DEFINES A BIT NUMBER IN THE WORD. THE SPECIFIED BIT CAN BE CLEARED OR SET DIRECTLY, OR IT CAN BE CLEARED OR SET UNDER SEMAPHORE PROTECTION. THE SEMAPHORE PROTECTION GUARANTEES THAT THE ENTIRE INSTRUCTION IS EXECUTED WITHOUT INTERVENTION FROM ANY OTHER DATA TRANSFER ON THE MAIN BUS.

********** CLEAR SINGLE BIT ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
CLEAR SINGLE BIT CLRS

FORMAT:

	OP1		OP2	NORM	MOD	BUS
CLRS	R3		R3	4	4	1
CLRS	R3	M	B0.X3	5	5	2
CLRS	X3		R3	6	6	3
CLRS	X3	M	B0.X3	8	8	4

FUNCTION:
OP1:= OP1&0[OP2 MODULO 16:0:1];
CLEAR MODIFICATION;

DESCRIPTION:
CLEARS A BIT IN OP1. THE BIT NUMBER IS DEFINED BY OP2 MODULO 16.

CODE:

	OP1	OP2	BINARY	HEXA
CLRS	R3	R3		0088
CLRS	R3	B0.X3		0888
CLRS	X3	R3		8089
CLRS	X3	B0.X3		8889

********** SET SINGLE BIT ********** 1 2 3 4 5 6 7 8

INSTRUCTION:
SET SINGLE BIT SETS

FORMAT:

	OP1	OP2	NORM	MOD	BUS
SETS	R3	R3	4	4	1
SETS	R3	M B0.X3	5	5	2
SETS	X3	R3	6	6	3
SETS	X3	M B0.X3	8	8	4
SETS	R3	C4	3	3	1
SETS	X3	C4	5	5	3

FUNCTION:
OP1:= OP1&1[OP2 MODULO 16:0:1];
CLEAR MODIFICATION;

DESCRIPTION:
SETS A BIT IN OP1. THE BIT NUMBER IS DEFINED BY OP2 MODULO 16.

CODE:

	OP1	OP2	BINARY	HEXA
SETS	R3	R3		0052
SETS	R3	B0.X3		0852
SETS	X3	R3		8053
SETS	X3	B0.X3		8853
SETS	R3	C4		00A3
SETS	X3	C4		00E3

********** RELEASE SINGLE BIT ********* 1 2 3 4 5 6 7 8

INSTRUCTION:
RELEASE SINGLE BIT RELS

FORMAT:

	OP1	OP2	NORM	MOD	BUS
RELS	X3	C4	5	6	4

FUNCTION:
SET SEMAPHORE BLOCKING;
OP1:= OP1&0[OP2:0:1];
CLEAR SEMAPHORE BLOCKING;

DESCRIPTION:
CLEARS A BIT IN OP1. THE BIT NUMBER IS DEFINED BY OP2. THE INSTRUCTION IS PERFORMED UNDER SEMAPHORE PROTECTION.

NOTICE:
MODIFICATION HAS NO INFLUENCE ON THE INSTRUCTION. MODIFICATION IS NOT CLEARED.

CODE:

	OP1	OP2	BINARY	HEXA
RELS	X3	C4		0011

********** RESERVE SINGLE BIT ********* 1 2 3 4 5 6 7 8

INSTRUCTION:
RESERVE SINGLE BIT RESS

FORMAT:

	OP1	OP2	NORM	MOD	BUS
RESS	X3	C4	7	8	3+R

(EXECUTE NEXT: R=0)
(RESERVE AND SKIP: R=1)

FUNCTION:

```
SET SEMAPHORE BLOCKING;
IF OP1.[OP2:1]=1 THEN
  BEGIN
    OP1:= OP1&1[OP2:0:1];
    LOC:= LOC+2
  END
ELSE LOC:= LOC+1;
CLEAR SEMAPHORE BLOCKING;
```

DESCRIPTION:
TESTS A BIT IN OP1. IF ONE, NO ACTION IS TAKEN. IF ZERO, THE BIT IS SET, AND THE NEXT INSTRUCTION IS SKIPPED. THE BIT NUMBER IS DEFINED BY OP2. THE INSTRUCTION IS PERFORMED UNDER SEMAPHORE PROTECTION.

NOTICE:
MODIFICATION HAS NO INFLUENCE ON THE INSTRUCTION. MODIFICATION IS NOT CLEARED.

CODE:

	OP1	OP2	BINARY	HEXA
RESS	X3	C4		0051

****
********** INPUT/OUTPUT INSTRUCTIONS **********
****

********** I/O DEVICE INSTRUCTIONS **** * * * * * * 7 8

INSTRUCTION:

SENSE I/O	SIO
CONTROL I/O	CIO
READ I/O	RIO
WRITE I/O	WIO

FORMAT:

	OP1	OP2	NORM	MOD	BUS
<INSTR>	R3	(M)R3	3	3	2
<INSTR>	X3	(M)R3	4+B (WIO,CIO: B=0) (RIO,SIO: B=1)	4+B	3

FUNCTION:

```
DEV:= OP2+MOD;
CASE INSTR OF
  BEGIN
    SIO, RIO: OP1:= DEVICE MEMORY(DEV);
    CIO, WIO: DEVICE MEMORY(DEV):= OP1;
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

THESE INSTRUCTIONS ARE USED TO ACCESS CONTROLLER MODULES. THE SIX LEAST SIGNIFICANT BITS OF OP2 PLUS MOD ARE THE DEVICE ADDRESS AND THE FOUR MOST SIGNIFICANT BITS ARE THE CRATE ADDRESS WHILE THE REMAINING BITS [11:6] MAY BE USED AS COMMAND WITH A DEVICE DEPENDENT MEANING. THE DATA ARE DEFINED BY OP1. THE OPERATIONS ARE:

SIO, RIO: READ A WORD FROM THE DEVICE.
CIO, WIO: WRITE A WORD TO THE DEVICE.

CODE:

	OP1	OP2	BINARY	HEXA
SIO	R3	R3		0896
SIO	X3	R3		08D6
CIO	R3	R3		8896
CIO	X3	R3		88D6
RIO	R3	R3		0897
RIO	X3	R3		08D7
WIO	R3	R3		8897
WIO	X3	R3		88D7

****
********** SPECIAL INSTRUCTIONS **********
****

THE SPECIAL INSTRUCTIONS COVERS THE FOLLOWING AREAS:
- LOAD AND SAVE OF THE PROCESS STATUS WORD
- LOAD AND SAVE OF THE TIMER
- LOAD OF THE BOUND REGISTER
- LOAD AND SAVE OF THE LEVEL REGISTER
- LOAD AND SAVE PROCESS
- MONITOR CALL
- RETURN FROM MONITOR ROUTINE
- TRAP
- CPU INTERRUPT AND SIMULATED I/O INTERRUPT
- SWITCH TO ALTERNATIVE INSTRUCTION SET
- ACCESS OF MAP MEMORY

********** STATUS AND TIMER HANDLING ** * * * * * * 7 8

INSTRUCTION:

LOAD STATUS	LDS
SYSTEM LOAD STATUS	SLS
SAVE STATUS	SVS
SYSTEM SAVE STATUS	SSS
LOAD TIMER	LDT
SAVE TIMER	SVT
LOAD MASK	LDM

FORMAT:

	OP1	OP2	NORM	MOD	BUS
LDS		R3	5	6	1
LDS		M B0.X3	5	6	2
SLS		R3	8	8	1
SLS		M B0.X3	10	10	2
SVS		R3	3	4	1
SVS		M B0.X3	8	9	2
SSS		R3	12	12	1
SSS		M B0.X3	14	14	2
LDT		R3	3	3	1
LDT		M B0.X3	3	3	2
SVT		R3	3	4	1
SVT		M B0.X3	6	7	2
LDM		C4	11	11	1

FUNCTION:

```
CASE INSTR OF
  BEGIN
    LDS,
    SLS: BEGIN
           IF PSW[13] = 1 THEN
             DISABLE CACHE;
           PSW := OP2;
           UMAP := UMAP & PSW[1:3:2];
           IF PSW[13] = 1 THEN
             ENABLE CACHE;
         END;
    SVS,
    SSS: OP2:= PSW;
    LDT: TIMER:= OP2;
    SVT: OP2:= TIMER;
    LDM: PSW:= PSW&OP2[15:7:2];
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

LDS, SLS: LOADS THE PROCESS STATUS WORD.
SVS, SSS: SAVES THE PROCESS STATUS WORD.
LDT: LOADS THE TIMER REGISTER.
SVT: SAVES THE TIMER REGISTER.
LDM: LOADS THE INTERRUPT MASK BITS INTO PSW.

NOTICE:

THE INSTRUCTIONS SVS, SSS AND LDS, SLS ARE SYNONYMS FOR THE SAME INSTRUCTION.
THE CACHE OPERATIONS ARE ONLY EXECUTED FOR THE CPU TYPE 7.

CODE:

	OP1	OP2	BINARY	HEXA
LDS		R3	|1110|0|R3|10111100|	E0BC
LDS		B0.X3	|1101|1|X3|10111100|	E8BC
SLS		R3	|0000|0|R3|10111101|	00BD
SLS		B0.X3	|0000|1|X3|10111101|	08BD
SVS		R3	|1101|0|R3|10111100|	D0BC
SVS		B0.X3	|1101|1|X3|10111100|	D8BC
SSS		R3	|0001|0|R3|10111100|	10BC
SSS		B0.X3	|0001|1|X3|10111100|	18BC
LDT		R3	|0101|0|R3|10111100|	50BC
LDT		B0.X3	|0101|1|X3|10111100|	58BC
SVT		R3	|1100|0|R3|10111100|	C0BC
SVT		B0.X3	|1100|1|X3|10111100|	C8BC
LDM		C4	|1101| C4 |10111110|	D0BE

********** LOAD BOUND REGISTER ******** * * * * * 6 7 8

INSTRUCTION:
LOAD BOUND REGISTER LBR

FORMAT:

	OP1	OP2	NORM	MOD	BUS
LBR			7	8	2

FUNCTION:
BOUND:= MEM[BASE-1];
CLEAR MODIFICATION;

DESCRIPTION:
LOADS THE BOUND REGISTER FROM MEM[BASE-1]

CODE:

	OP1	OP2	BINARY	HEXA
LBR			¦1011¦0001¦10111100¦	B1BC

********** LEVEL HANDLING ********** * * * * * 6 7 8

INSTRUCTION:

SAVE LEVEL	SVL
LOAD LEVEL	LDL

FORMAT:

	OP1	OP2	NORM	MOD	BUS
SVL		R3	3	3	1
SVL		M B0.X3	5	5	2
LDL		R3	3	3	1
LDL		M B0.X3	4	4	2

FUNCTION:

```
CASE INSTR OF
  BEGIN
    SVL: OP2:= LEVEL;
    LDL: LEVEL:= OP2;
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

SVL: SAVES THE LEVEL REGISTER
LDL: LOADES THE LEVEL REGISTER

CODE:

	OP1	OP2	BINARY	HEXA
SVL		R3	¦0101¦0¦R3¦10111101¦	50BD
SVL		M B0.X3	¦0101¦1¦X3¦10111101¦	58BD
LDL		R3	¦1000¦0¦R3¦10111101¦	80BD
LDL		M B0.X3	¦1000¦1¦X3¦10111101¦	88BD

********** SAVE PROCESS ********** * * * * * * 7 8

INSTRUCTION:
SAVE PROCESS SVP

FORMAT:

	OP1	OP2	NORM	MOD	BUS
SVP		L4	24	24	16

FUNCTION:

```
FOR I:= 0 STEP 1 UNTIL 7 DO MEM[BASE+I]:= REG[I];
MEM[BASE+8]:=  BASE;
MEM[BASE+9]:=  UMAP;
MEM[BASE+10]:= PROG;
MEM[BASE+11]:= OP2;
MEM[BASE+12]:= TIMER;
MEM[BASE+13]:= PSW;
MEM[BASE-2]:= LEVEL;
IF LEVEL[14:15] = 0 THEN MEM[BASE - 1] := BOUND;
PSW:= PSW&3[15:1:2];
CLEAR MODIFICATION;
```

DESCRIPTION:
SAVES THE PROCESS AT ITS BASE IN MEMORY. THE OPERAND DEFINES THE LOCATION IN WHICH THE PROCESS WILL CONTINUE AFTER RELOADING. THE PSW, AFTER BEING SAVED, IS CHANGED TO DISABLE ALL INTERRUPTS.

REMARK:
THE INSTRUCTION IS FOLLOWED BY THE NEXT INSTRUCTION, I.E. THE SAVED PROCESS CONTINUES EXECUTION.

CODE:

	OP1	OP2	BINARY	HEXA
SVP		L4	|0010| C4 |10111110|	20BE

********** LOAD PROCESS ********** * * * * * * 7 8

INSTRUCTON:

LOAD PROCESS NONLINKED	LDN
LOAD PROCESS	LDP

FORMAT:

	OP1	OP2	NORM	MOD	BUS
LDN		R3	30+N	30+N	17+M
LDN		M B0.X3	30+N	30+N	18+M
LDP		R3	32+N	32+N	19+M
LDP		M B0.X3	32+N	32+N	20+M

(LEVEL= 0: N=2, M=1)
(LEVEL<>0: N=0, M=0)

FUNCTION:

```
DEFINE OP3 = SUCCESSOR(OP2);
WORK1 := BASE;
WORK2 := UMAP;
UMAP := UMAP & OP3[7:7:8];
UPDATE USER DATA MAP;
IF PSW[13] = 1 THEN
  DISABLE CACHE;
LEVEL := MEM[OP2 - 2];
IF LEVEL[14:15] = 0 THEN
  BOUND := MEM[OP2 - 1]
ELSE
  BOUND := -1;
FOR I:= 0 STEP 1 UNTIL 7 DO REG[I]:= MEM[OP2+I];
BASE:=    MEM[OP2+8];
UMAP:=    MEM[OP2+9];
PROG:=    MEM[OP2+10];
LOC:=     MEM[OP2+11];
TIMER:=   MEM[OP2+12];
PSW:=     MEM[OP2+13];
UPDATE MAP;
IF PSW[13] = 1 THEN
  ENABLE CACHE;
IF INSTR = LDP THEN
  BEGIN
    MEM[BASE+14] := WORK1;
    MEM[BASE+15] := WORK2;
  END;
CLEAR MODIFICATION;
```

DESCRIPTION:

A PROCESS DESCRIPTOR IS LOADED FROM AN ABSOLUTE ADDRESS (AS DEFINED IN THE DESCRIPTION OF THE ALTERNATIVE INSTRUCTIONS) DEFINED BY OP2.
IN CASE OF A LINKED LOAD, THE ABSOLUTE ADDRESS OF THE CURRENT PROCESS DESCRIPTOR (DATA PART OF UMAP AND BASE) IS STORED IN THE NEW PROCESS DESCRIPTOR.

NOTICE:

THE CACHE OPERATIONS ARE ONLY EXECUTED FOR THE CPU TYPE 7.

CODE:

	OP1	OP2	BINARY	HEXA
LDN		R3	¦0100¦0¦R3¦10111100¦	40BC
LDN		B0.X3	¦0100¦1¦X3¦10111100¦	48BC
LDP		R3	¦0000¦0¦R3¦10111100¦	00BC
LDP		B0.X3	¦0000¦1¦X3¦10111100¦	08BC

********** EXECUTE INSTRUCTION ******** 1 2 3 4 5 6 7 8

INSTRUCTION:

EXECUTE	XCU
EXECUTE INDIRECT	XCUI

FORMAT:

	OP1	OP2	NORM	MOD	BUS
XCU		X3	4	5	2
XCUI	M	B0.X3	5	6	3

FUNCTION:

```
INSTR:= OP2;
IF INDIRECT THEN INSTR:= MEM[BASE+INSTR];
CLEAR MODIFICATION;
GOTO DECODE INSTRUCTION;
```

DESCRIPTION:

IF INDIRECT MARKED, THE OPERAND POINTS IN MEMORY AT THE RESULTING OPERAND. THE OPERAND IS EXECUTED AS AN INSTRUCTION PLACED AT THE LOCATION OF THE EXECUTE INSTRUCTION.

NOTICE:

THE LOCATION COUNTER IS NOT INCREMENTED AS THIS WILL BE DONE IN THE DECODED INSTRUCTION.

CODE:

	OP1	OP2	BINARY	HEXA
XCU		X3		80BC
XCUI		B0.X3		88BC

********** MONITOR CALL ********** * * * * * 6 7 8

INSTRUCTION:
MONITOR CALL MON

FORMAT:

	OP1	OP2	NORM	MOD	BUS
MON		C8	7+N	7+N	2+M
			(SPECIAL: N=1,M=0)		
			(INDIRECT: N=2,M=1)		

FUNCTION:
```
REG[7]:= LOC+1-PROG;
LEVEL:= LEVEL+1;
PSW:= PSW&(0)[0:0:1];                % CLEAR BIT 0
BOUND:= -1;
LOC:=    IF OP2<64 THEN    63        % SPECIAL
    ELSE IF OP2<256 THEN   MEM[OP2]; % INDIRECT
CLEAR MODIFICATION;
```

DESCRIPTION:
JUMPS WITH RETURN ADDRESS IN R7 TO AN ABSOLUTE MEMORY LOCATION DEPENDING ON OP2:

OP2<64: JUMPS TO LOCATION 63.
63<OP2<256: JUMPS INDIRECT VIA THE LOCATION SPECIFIED BY OP2.

CODE:

	OP1	OP2	BINARY	HEXA
MON		C8		00A6

********** RETURN MONITOR CALL ******** * * * * * 6 7 8

INSTRUCTION:

RETURN MONITOR CALL	RTM
RETURN MONITOR INDIRECT	RTMI

FORMAT:

	OP1	OP2	NORM	MOD	BUS
RTM	M	P6.X2	7+N	8+N	3+M
RTMI	M	B6	8+N	9+N	3+M
			(LEVEL= 0: N=6, M=1)		
			(LEVEL<>0: N=0, M=0)		

FUNCTION:

```
LEVEL:= LEVEL-1;
IF LEVEL=0 THEN PSW:= PSW&(1)[0:0:1];        % SET BIT 0
IF LEVEL[14:15]=0 THEN BOUND:=MEM[BASE-1];
LOC:= OP2+PROG;
CLEAR MODIFICATION;
```

DESCRIPTION:

JUMPS UNCONDITIONALLY TO THE LOCATION SPECIFIED BY OP2. THE LEVEL REGISTER IS DECREMENTED, AND IF IT BECOMES ZERO, A SWITCH TO THE ORIGINAL PROGRAMPAGE IS PERFORMED. IF THE LEVEL COUNTER (LEVEL,BIT 14-0) BECOMES ZERO, A SWITCH TO THE ORIGINAL BOUND-VALUE IS PERFORMED.

CODE:

	OP1	OP2	BINARY			HEXA
RTM		P6.X2	¦X2¦	P6	¦11100110¦	00E6
RTMI		B6	¦10¦	B6	¦11100011¦	80E3

********** CPU INTERRUPT ********** * * * * * * 7 8

INSTRUCTION:
CPU INTERRUPT CPU

FORMAT:

	OP1	OP2	NORM	MOD	BUS
CPU		R3			
CPU	M	B0.X3			

FUNCTION:
NOTIFY THE INTERRUPT VECTOR SPECIFIED BY OP2;
CLEAR MODIFICATION;

DESCRIPTION:
AN INTERRUPT IS ISSUED TO THE INTERRUPT VECTOR SPECIFIED BY OP2.

CODE:

	OP1	OP2	BINARY	HEXA
CPU		R3	|0110|0|R3|10111110|	60BE
CPU	M	B0.X3	|0110|1|X3|10111110|	68BE

********** NON-MASKABLE INTERRUPT ******** * * * * * * * 8

INSTRUCTION:
NON-MASKABLE INTERRUPT NMI

FORMAT:

	OP1	OP2	NORM	MOD	BUS
NMI			*	*	*

FUNCTION:
CAUSE ISSUING OF NON-MASKABLE INTERRUPT;
CLEAR MODIFICATION;

DESCRIPTION:
COMMANDS THE BUS CONTROLLING DEVICE TO ISSUE A NON-MASKABLE INTERRUPT, AWAITS AND IGNORES IT ITSELF.
FETCHES A PROCESS DESCRIPTOR REFERENCE FROM ABSOLUTE ADDRESS:
#FFFFE0 + (2 * CPU#)
SAVES CONTEXT AT THIS ADDRESS.
THIS CONTEXT HOLDS CAUSE CODE = 8 IN WORD 17 AND THE SYSTEM PROGRAM MAP REGISTER IN WORD 16.
AWAITS NMI NOTIFICATION.
WHEN NOTIFIED, READS ITS NMI NOTIFICATION DESCRIPTOR (FROM THE BUS CONTROLLING DEVICE).
CONTINUES EXECUTION FROM THE POINT DEFINED BY DESCRIPTOR.

CODE:

	OP1	OP2	BINARY	HEXA
NMI			¦1011¦0010¦10111100¦	B2BC

********** TRAP INSTRUCTION ********** 1 2 3 4 5 6 7 8

INSTRUCTION:

TRAP	TRP

FORMAT:

	OP1	OP2	NORM	MOD	BUS
TRP		C4	*	*	*

FUNCTION:
CLEAR MODIFICATION;
ILLEGAL INSTRUCTION;

DESCRIPTION:
THE TRAP INSTRUCTION IS HANDLED AS AN ILLEGAL INSTRUCTION. AN INTERNAL INTERRUPT IS GENERATED WITH CAUSE=1.

CODE:

	OP1	OP2	BINARY	HEXA
TRP		C4		00BE

********** CACHE INSTRUCTIONS ******** * * * 4 5 6 7 *

INSTRUCTION:

CACHE ENABLE	CAE
CACHE DISABLE	CAD

FORMAT:

	OP1	OP2	NORM	MOD	BUS
CAE			*	*	*
CAD			*	*	*

```
FUNCTION:
  CASE INSTR OF
       CAE:    ENABLE THE CACHE FUNCTION;
       CAD:    DISABLE THE CACHE FUNCTION;
  END;
CLEAR MODIFICATION;
```

NOTICE:
IN CPU TYPE 8 (CMR) THESE INSTRUCTIONS ARE INTERPRETED AS NOP's.

CODE:

	OP1	OP2	BINARY	HEXA
CAE			1000 0000 1011 1110	80BE
CAD			1000 0001 1011 1110	81BE

********** TEST CPU ********** * * * 4 5 6 7 8

INSTRUCTION:

TEST CPU	TST

FORMAT:

	OP1	OP2	NORM	MOD	BUS
TST	p		*	*	*

```
FUNCTION:
  BEGIN
       TEST CPU
       IF ERROR THEN
          R0:=ERROR CODE
       ELSE
          R0:=0000(H);
       CLEAR MODIFICATION;
  END
```

DESCRIPTION:

AN INTERNAL FIRMWARE TEST OF THE CPU HARDWARE IS PERFORMED, AND REGISTER 0 IS LOADED WITH A RESULT DEPENDING ERRORCODE. THE TEST DESTROYS CONTENTS OF REGISTER 0-3 AND ARITHMETIC FLAGS IN PSW (BIT(7:4)).

ERRORCODES:

```
       R0=0      ;NO ERRORS DETECTED
       R0=1      ;ALS TEST FAILED
       R0=2      ;AMS TEST FAILED
       R0=3      ;OPREG TEST FAILED
       R0=4      ;PSW TEST FAILED
       R0=5      ;CACHE TEST FAILED
```

DETAILED INFORMATION ABOUT ERRORCODES REF. TO HARDWARE MANUAL "FUNCTIONAL DESCRIPTIONS". UNDER ANY CIRCUMSTANCES R0 <> 0 MEANS AN ERROR WHICH MIGHT CAUSE SOFTWARE RUN AWAY.

CODE:

	OP1	OP2	BINARY	HEXA
TST	p		1000 0011 1011 1110	83BE

********** ALTERNATIVE INSTRUCTION SET ** * * * * * 6 7 8

INSTRUCTION:
SWITCH TO ALTERNATIVE INSTRUCTION SET ALT

FORMAT:

	OP1	OP2	NORM	MOD	BUS
ALT			4	5	1

FUNCTION:

```
PSW:= PSW&(1)[12:0:1];      % SET BIT 12
IF CPU_TYPE = 6 THEN CLEAR MODIFICATION;
```

DESCRIPTION:
SETS BIT 12 IN PSW, CAUSING AN ALTERNATIVE INSTRUCTION DECODING.

CODE:

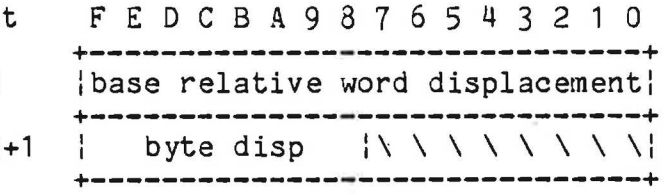
	OP1	OP2	BINARY	HEXA
ALT			¦1011¦0000¦10111100¦	B0BC

****
********** ALTERNATIVE INSTRUCTIONS **********
****

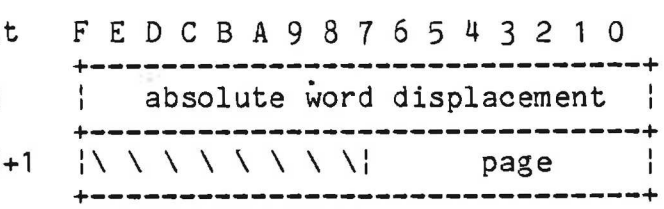
The set of alternative instructions is implemented to support memory access outside the 64K data area of a process, to reserve semaphore bits and to access I/O devices with controlled action for timeout and parity errors.

The following definitions concern absolute addresses, which are used in the alternative instructions.

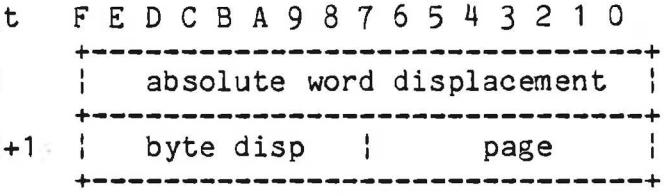
BX33 is a base relative byte address, on own data page, calculated as
byte_addr = 2*reg[X3] + reg[X3+1] [15:8]



AX33 is an absolute word address, on page 0..255, calculated as
page= reg[X3+1] [7:8]
word_addr= reg[X3]



ABX33 is an absolute byte address, on page 0.255, calculated as
page= reg[X3+1] [7:8]
byte_addr= 2*reg[X3] + reg[X3+1] [15:8]



The set of alternative instructions consists of the following:

- Absolute move instructions.
- Absolute reserve and release single bit.
- Alternative IO instructions.

Most of the alternative instructions, as defined herein, have in common a double exit possibility, where the first exit location is an error exit, which can be reached if a transfer error (time-out or parity error) occurs. In this case, the CPU stores the address of the instruction in location 16 relative to base, stores the cause (2 for parity error, 3 for time-out) in location 17 relative to base, disables interrupts and exits to the error exit.

This means, that the programmer must ensure re-enabling of interrupts if the instructions exits to the error exit (possibly by a LDM instruction).

********** ABSOLUTE MOVE WORDS ******* * * * * * * 7 8

Format:

	op1	op2	op3	binary	hexa
AAMOW	AX33	AX33	R3	|0AX3|0AX3|1000|0 R3|	0080
LAMOW	X3	AX33	R3	|1 X3|0AX3|1000|0 R3|	8080
ALMOW	AX33	X3	R3	|0AX3|1 X3|1000|0 R3|	0880

Function:

```
WHILE OP3>0 DO
  BEGIN
    OP2 := OP1;  COMMENT IF STEP1_CPU THEN
                    BYPASS CACHE;
    INCR (INDEX1);
    INCR (INDEX2);
    DECR (OP3);
  END;
  IF PSW[13] = 1 THEN
    CLEAR CACHE;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

Moves a memory portion. Source is referenced by OP1. Destination is referenced by OP2. The length in words is determined by OP3. Index registers and length count are updated during the instruction execution.

Notice:

The cache operations are only executed for the CPU type 7.

The instruction has 2 exit points:

loc+1 if time-out or parity error occured in a memory reference

loc+2 if terminated normally

The instruction may be interrupted for each 16 words moved.

********** ABSOLUTE MOVE BYTES ******* * * * * * * 7 8

Format:

	op1	op2	op3	binary	hexa
AAMOB	ABX33	ABX33	R3	¦0AX3¦0AX3¦1000¦1 R3¦	0088
LAMOB	BX33	ABX33	R3	¦1 X3¦0AX3¦1000¦1 R3¦	8088
ALMOB	ABX33	BX33	R3	¦0AX3¦1 X3¦1000¦1 R3¦	0888

Function:

```
WHILE OP3>0 DO
  BEGIN
    OP2 := OP1;  COMMENT IF STEP1_CPU THEN
                    BYPASS CACHE;
    INCR (INDEX1);
    INCR (INDEX2);
    DECR (OP3);
  END;
  IF PSW[13] = 1 THEN
    CLEAR CACHE;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

Moves a memory portion. Source is referenced by OP1. Destination is referenced by OP2. The length in bytes is determined by OP3. Index registers and length count are updated during the instruction execution.

Notice:

The cache operations are only executed for the CPU type 7.

The instruction has 2 exit points:

loc+1 if time-out or parity error occured in a memory reference

loc+2 if terminated normally

The instruction may be interrupted for each 32 bytes moved.

********** ABSOLUTE MOVE WORD ******** * * * * * * 7 8

Format:

	op1	op2	op3	binary	hexa
AMOW	AX33	R3		¦0AX3¦1 R3¦1001¦0001¦	0891
AMOW	R3	AX33		¦0AX3¦0 R3¦1001¦0001¦	0091

Function:

```
IF PSW[13] = 1 THEN
  DISABLE CACHE;
OP2 := OP1;
IF PSW[13] = 1 THEN
  ENABLE CACHE;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

Moves a word between absolute memory and a register.

Notice:

The cache operations are only executed for the CPU type 7.

The instruction has 2 exit points:

loc+1 if time-out or parity error occured in a memory reference

loc+2 if terminated normally

********** ABSOLUTE MOVE BYTE ******** * * * * * * 7 8

Format:

	op1	op2	op3	binary	hexa
AMOB	ABX33	R3		¦1AX3¦1 R3¦1001¦0001¦	8891
AMOB	R3	ABX33		¦1AX3¦0 R3¦1001¦0001¦	8091

Function:

```
IF PSW[13] = 1 THEN
  DISABLE CACHE;
OP2 := OP1;
IF PSW[13] = 1 THEN
  ENABLE CACHE;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

Moves a byte between absolute memory and a register.

Notice:

The cache operations are only executed for the CPU type 7.

The instruction has 2 exit points:

loc+1 if time-out or parity error occured in a memory reference

loc+2 if terminated normally

********** ABSOLUTE RESERVE BIT ****** * * * * * * 7 8

Format:

	op1	op2	op3	binary	hexa
ARESS	AX33	C4		|0AX3| C4 |1001|0010|	0092

Function:

```
IF PSW[13] = 1 THEN
  DISABLE CACHE;
SET SEMAPHORE BLOCKING;
IF OP1.[OP2:1]=1 THEN
  BEGIN
    OP1 := OP1 & 1[OP2:0:1];
    LOC := LOC + 3;
  END
ELSE
  LOC := LOC + 2;
CLEAR SEMAPHORE BLOCKING;
IF PSW[13] = 1 THEN
  ENABLE CACHE;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

Tests a bit in OP1, If one, one instruction is skipped. If zero, the bit is set, and two instructions are skipped. The bit number is defined by OP2. The instruction is performed under semaphore protection.

Notice:

The cache operations are only executed for the CPU type 7.

The instruction has 3 exit points:

- loc+1 if time-out or parity error occured in a memory reference
- loc+2 if the bit was initially one
- loc+3 if the bit was initially zero

********** ABSOLUTE RELEASE BIT ****** * * * * * * 7 8

Format:

	op1	op2	op3	binary	hexa
ARELS	AX33	C4		¦1AX3¦ C4 ¦1001¦0010¦	8092

Function:

```
IF PSW[13] = 1 THEN
  DISABLE CACHE;
SET SEMAPHORE BLOCKING;
OP1 := OP1 & 0[OP2:0:1];
CLEAR SEMAPHORE BLOCKING;
IF PSW[13] = 1 THEN
  ENABLE CACHE;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

Clears a bit in OP1. The bit number is defined by OP2. The instruction is performed under semaphore protection.

Notice:

The cache operations are only executed for the CPU type 7.

The instruction has 2 exit points:

loc+1 if time-out or parity error occured in a memory reference
loc+2 if terminated normally

********** INPUT/OUTPUT INSTRUCTIONS ***** * * * * * * 7 8

Format:

	op1	op2	op3	binary	hexa
DRD	R3	M	R3	|0 R3|0 R3|1001|0011|	0093
DRD	X3	M	R3	|0 X3|1 R3|1001|0011|	0893
DWR	R3	M	R3	|1 R3|0 R3|1001|0011|	8093
DWR	X3	M	R3	|1 X3|1 R3|1001|0011|	8893

Function:

```
CASE INSTR OF
  BEGIN
    DRD:  OP1 := DEVICE REGISTER (OP2);
    DWR:  DEVICE REGISTER (OP2) := OP1;
  END;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

The four most significant bits of OP2 designates the CU containing the device - zero designates the PU.
The six least significant bits of OPO2 designates the device within the CU or PU.
The remaining bits of OP2, OP2[11:6], may be used as a command with a device dependant meaning.

The I/O read command reads a device register into OP1. The I/O write command writes OP1 into a device register.

Notice:

The instruction has 2 exit points:
loc+1 if time-out or parity error occured in a I/O reference
loc+2 if terminated normally

********** MAP ACCESS INSTRUCTIONS ***** * * * * * * 7 *

Format:

	op1	op2	op3	binary	hexa
MRD	R3	M R3		¦0 R3¦0 R3¦0101¦0001¦	0051
MRD	X3	M R3		¦0 X3¦1 R3¦0101¦0001¦	0851
MWR	R3	M R3		¦1 R3¦0 R3¦0101¦0001¦	8051
MWR	X3	M R3		¦1 X3¦1 R3¦0101¦0001¦	8851

Function:

```
CASE INSTR OF
  BEGIN
    MRD:  OP1 := BUS CONTROLLING DEVICE LOCATION (OP2);
    MWR:  BUS CONTROLLING DEVICE LOCATION (OP2) := OP1;
  END;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

These instructions are used to read and write the internal control memory of the mapping module.

Notice:

Available in CPU type 7 (MX step 1). Illegal instruction in CPU type 8 (MX step 2).

********** SYSTEM MAP INSTRUCTIONS ***** * * * * * * 7 8

Format:

	op1	op2	op3	binary	hexa
RSP	R3			|0 R3|0000|0101|0010|	0052
RSP	X3			|0 X3|1000|0101|0010|	0852
WSP	R3			|1 R3|0000|0101|0010|	8052
WSP	X3			|1 X3|1000|0101|0010|	8852

Function:

```
CASE INSTR OF
  BEGIN
    RSP:  OP1 := SYSTEM PROGRAM MAP REGISTER & 0[15:7:8];
    WSP:  SYSTEM PROGRAM MAP REGISTER := OP1[7:8];
  END;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

These instructions are used to read and write into the system program map register.

** ABSOLUTE READ SYNDROME BITS AND CORRECT WORD * * * * * * * 8

Format:

	op1	op2	op3	binary	hexa
ARSC	AX33	R3		¦0 AX33¦0 R3¦0101¦0011¦	0053

POSSIBLE PARITY ERROR

Function:

```
RESERVE MEMORY;
WORK:= OP1 CORRECTED BY EDC LOGIC;
OP2:= SYNDROME BITS (OP1);
OP1:= WORK;
RELEASE MEMORY;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;
```

Description:

Indivisible read-modify-write instruction, which is used for correction of single-bit errors in EDC RAM. The syndrome bits indicate the bit-in-error, if any.

Notice

Available in CPU type 8 (MX step 2) only.

* ABSOLUTE READ WORD WITH ERROR DETECTION DISABLED * * * * * * * 8

Format:

	op1	op2	op3	binary	hexa
ARED	AX33	R3		|1 AX33|0 R3|0101|0111|	~~8053~~ 08

Function:

OP2:= OP1 WITH NO DETECTION AND CORRECTION;
CLEAR MODIFICATION;
RETURN FROM ALTERNATIVE INSTRUCTION SET;

Description:

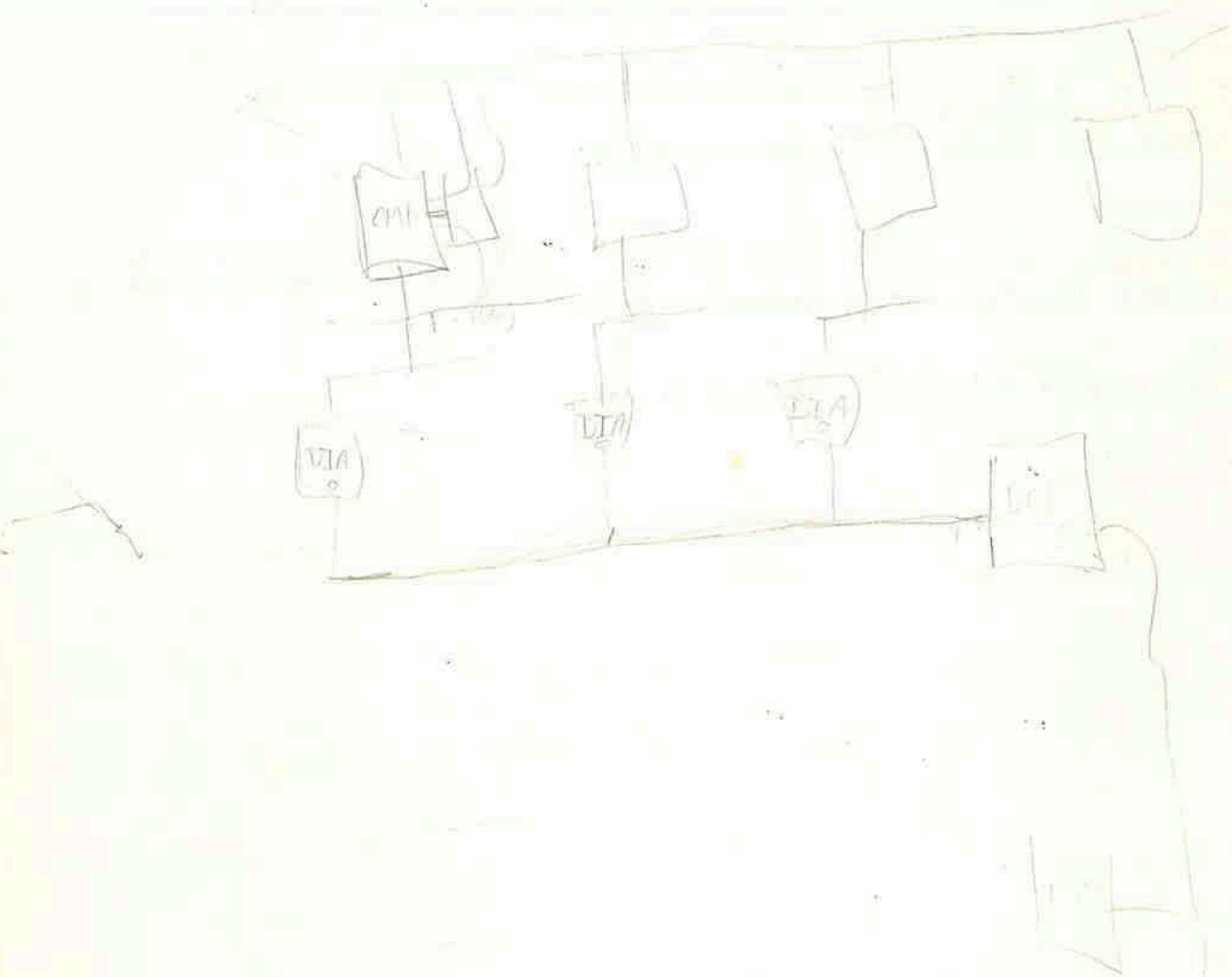
Reads the uncorrected data word in memory without generating parityam error interrupt even if the word has a multiple bit failure. (Correct parity generated by RAM).

Notice:

Available in CPU type 8 (MX step 2) only.

8853
8853

8053 skriv check bit
8853 skriv ord uden checkbit


CIA
VIA
VIA
VIA
CIA