Instruction Word

Opcode # (DEC)	Opcode # (HEX)	MSB(HEX)	Inst Size (15)	ExecCycles 00,01,10 = 1 Cyc 11 = 2 Cyc (14-13)	Opcode (12-9)	Register C (8-6)	Register B (5-3)	Register A (2-0)	Word2	Mneumonic	Mneumonic Fmt	Operation	Flags	Needed ALU	Actual ALU Op
Fetch Cyc															
Fetch Cycle 2 (LARGE INSTRUCTION)															
0	#0	#0,#1	0	00	0000					NOP	NOP	No Operation	~		a
1	#1	#2,#3	0	00	0001					HALT	HALT	STOPS ALL OPERATIONS			a
2	#2	#4,#5	0	00	0010	ccc	bbb	aaa		SUB	SUB a,b,c	Sub Reg-b from Reg-a, Store in Reg-c	ZF,CF,OVF	a-b	a-b
3	#3	#6,#7	0	00	0011	ccc	bbb	aaa		SUBC	SUBC a,b,c		ZF,CF,OVF	a-b	a-b
4	#4	#8,#9	0	00	0100	ccc	bbb	aaa		ADD	ADD a,b,c	Add Reg-a and Reg-b, Store in Reg-c	ZF,CF,OVF	a+b	a+b
5	#5	#A,#B	0	00	0101	ccc	bbb	aaa		ADDC	ADDC a,b,c		ZF,CF,OVF	a+b	a+b
6	#6	#C,#D	0	00	0110	ccc	bbb	aaa		XOR	XOR a,b,c	Logical XOR Reg-a and Reg-b, Store in Reg-c	ZF,CF/OVF=0	a $ b	a $ b
7	#7	#E,#F	0	00	0111										a $ b
8	#8	#10,#11	0	00	1000	ccc	bbb	aaa		OR	OR a,b,c	Logical OR Reg-a and Reg-b, Store in Reg-c	ZF,CF/OVF=0	a | b	a | b
9	#9	#12,#13	0	00	1001										a | b
10	#A	#14,#15	0	00	1010	ccc	bbb	aaa		AND	AND a,b,c	Logical AND Reg-a and Reg-b, Store in Reg-c	ZF,CF/OVF=0	a & b	a & b
11	#B	#16,#17	0	00	1011									~	a & b
12	#C	#18,#19	0	00	1100	ccc	bbb	000		NOT	NOT b,c	Logical NOT Reg-a into Reg-c	ZF,CF/OVF=0	!b	!b
13	#D	#1A,#1B	0	00	1101									~	!b
14	#E	#1C,#1D	0	00	1110									~	b
15	#F	#1E,#1F	0	00	1111									~	b
16	#10	#20,#21	0	01	0000	000	bbb	aaa		STORERW	STOREW a,[b]	Store word reg-a to address in reg-b	~	a	a
17	#11	#22,#23	0	01	0001					STORERB	STOREB a,[b]	Store byte reg-a to address in reg-b	~	a	a
18	#12	#24,#25	0	01	0010	000	bbb	aaa		CMP	CMP a,b	Set flags based on a-b	ZF,CF,OVF	a-b	a-b
19	#13	#26,#27	0	01	0011					CMPC	CMPC	Complement Carry	CF=!CF	~	a-b
20	#14	#28,#29	0	01	0100					LOADRW	LOADW [b],c	Load word from address in reg-b into reg-c	~	~	a+b
21	#15	#2A,#2B	0	01	0101					LOADRB	LOADB [b],c	Load byte from address in reg-b into reg-c	~	~	a+b
22	#16	#2C,#2D	0	01	0110									~	a $ b
23	#17	#2E,#2F	0	01	0111									~	a $ b
24	#18	#30,#31	0	01	1000	000	bbb	000		JMPR	JMP [b]	jmp to [b]	~	~	a | b
25	#19	#32,#33	0	01	1001	000	bbb	000		JZR	JZ [b]	if Z: jmp to [b]	~	~	a | b
26	#1A	#34,#35	0	01	1010	000	bbb	000		JNZR	JNZ [b]	if NZ: jmp to [b]	~	~	a & b
27	#1B	#36,#37	0	01	1011	000	bbb	000		JCR	JC [b]	if C: jmp to [b]	~	~	a-b
28	#1C	#38,#39	0	01	1100	000	bbb	000		JNCR	JNC [b]	if NC: jmp to [b]	~	~	!b
29	#1D	#3A,#3B	0	01	1101	000	bbb	000		JVR	JV [b]	if V: jmp to [b]	~	~	!b
30	#1E	#3C,#3D	0	01	1110	000	bbb	000		JNVR	JNV [b]	if NV: jmp to [b]	~	~	b
31	#1F	#3E,#3F	0	01	1111						Jreserved			~	b
32	#20	#40,#41	0	10	0000						Shift Reserved			~	a
33	#21	#42,#43	0	10	0001						Shift Reserved			~	a
34	#22	#44,#45	0	10	0010	ccc	bbb	cnt		SHL	SHL b,c,cnt	c = b << (cnt+1) (zero fill)	CF=LAST BIT	~	b
35	#23	#46,#47	0	10	0011	ccc	bbb	cnt		SHR	SHR b,c,cnt	c = b >> (cnt+1) (zero fill)	CF=LAST BIT	~	b
36	#24	#48,#49	0	10	0100	ccc	bbb	cnt		SHLC	SHLC b,c,cnt	c = b << (cnt+1) (carry flag fill)	CF=LAST BIT	~	b
37	#25	#4A,#4B	0	10	0101					SHRA	SHRA b,c,cnt	c = b >> (cnt+1) (sign extented shift)	CF=LAST BIT	~	b
38	#26	#4C,#4D	0	10	0110					??SHRC??	SHRC b,cnt,c	c = b >> (cnt+1) (carry flag fill)	CF=LAST BIT	~	b
39	#27	#4E,#4F	0	10	0111						Shift Reserved			~	b
40	#28	#50,#51	0	10	1000									~	b
41	#29	#52,#53	0	10	1001									~	b
42	#2A	#54,#55	0	10	1010	ccc	000	000		MOVSP	MOVSP c	Register c = SP	ZF	b	b
43	#2B	#56,#57	0	10	1011	000	bbb	000		MOVTSP	MOVTSP b	SP = Register b	~	b	b
44	#2C	#58,#59	0	10	1100	ccc	bbb	000		MOV	MOV b,c	c = b	ZF	b	b
45	#2D	#5A,#5B	0	10	1101	000	000	000		MOVPC	MOVPC c	c = PC	ZF	~	b
46	#2E	#5C,#5D	0	10	1110	000	bbb	000		PUSH	PUSH b		~	b	b
47	#2F	#5E,#5F	0	10	1111										b
48	#30	#60,#61	0	11	0000	ccc		000		POP	POP c (Cycle 1)		~	~	a
48											POP c (Cycle 2)		~	~	
49	#31	#62,#63	0	11	0001									~	a
50	#32	#64,#65	0	11	0010									~	b
51	#33	#66,#67	0	11	0011									~	b
52	#34	#68,#69	0	11	0100	000	bbb	000		CALLR	CALL b (Cycle 1)		~	b	b
52											CALL b (Cycle 2)		~	b	
53	#35	#6A,#6B	0	11	0101	000	bbb	000						b	b
54	#36	#6C,#6D	0	11	0110						*CALL [b] (Cycle 1)*	*Not possible with D->B Bridge inside IR2*		~	b
54											*CALL [b] (Cycle 2)*	*Not possible with D->B Bridge inside IR2*			
55	#37	#6E,#6F	0	11	0111									~	b
56	#38	#70,#71	0	11	1000	000	000	000		RET	RET (Cycle 1)		~	~	b
56											RET (Cycle 2)		~	~	

Internal Control Signals (Memory Access … /SFT_WC, /DBH_OE … H2L_DIR); **Memory Control Signals** (MEM_R/W … /MEM_L_EN); **Fetch Control Signals** (/IR1_LATCH … /EXECUTE2)

Opcode # (DEC)	Memory Access	8b or 16b Mem	Jump Condition	/IR2_OE	/PC_BBOE	/PC_ADOE	PC_MODE/3	/JMPINST	/SP_BBOE	/SP_ADOE	SP_MODE/2	/SP_D2B_OE	/B2A_OE	/RF_CEW	/RF_FHZ	/RF_BOE	ALU_FUNC/3	/ALU_OE	/ALU_FL	/ALU_WC	/ALU_CFLATCH	/SHFT_INVERTCARRY	/SFT_OE	SFT_DIR	/SFT_AR	/SFT_WC	/DBH_OE	/DBL_OE	DB_DIR (0=W)	H/2L_OE	H2L_DIR	MEM_R/W	/MEM_AS	/MEM_H_EN	/MEM_L_EN	/IR1_LATCH	/IR2_LATCH	/EXECUTE1	/EXECUTE2
Fetch Cyc	y	16		1	1	0	001	1	1	1			1	1	1	1		1	1	1	1	1	1	~	~	~	1	1		1		1	0	0	0	0	1	1	1
Fetch Cycle 2	y	16		1	1	0	001	1	1	1			1	1	1	1		1	1	1	1	1	1	~	~	~	1	1		1		1	0	0	0	1	0	1	1
0	n			1	1	1	000	1	1	1	00		1	1	1	~		~	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
1	n			1	1	1	000	1	1	1	00		1	~	1	~		~	~	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
2	n			1	1	1	000	1	1	1	00		1	0	1	0	001	0	0	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
3	n			1	1	1	000	1	1	1	00		1	0	1	0	001	0	0	0	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
4	n			1	1	1	000	1	1	1	00		1	0	1	0	010	0	0	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
5	n			1	1	1	000	1	1	1	00		1	0	1	0	010	0	0	0	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
6	n			1	1	1	000	1	1	1	00		1	0	1	0	011	0	0	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
7																																							
8	n			1	1	1	000	1	1	1	00		1	0	1	0	100	0	0	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
9																																							
10	n			1	1	1	000	1	1	1	00		1	0	1	0	101	0	0	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
11																																							
12	n			1	1	1	000	1	1	1	00		1	0	1	0	110	0	0	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
13																																							
14																																							
15																																							
16	y	16		1	1	1	000	1	1	1	00		0	1	1	0	000	0	1	1	1	1	1	~	~	~	0	0	0	1	~	0	0	0	0	1	1	0	1
17	y	8		1	1	1	000	1	1	1	00		0	1	1	0	000	0	1	1	1	1	1	~	~	~	1	AD0	0	!AD0	0	0	0	!AD0	AD0	1	1	0	1
18	n			1	1	1	000	1	1	1	00		1	1	1	0	001	0	0	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
19				1	1	1	000	1	1	1	00		1	1	1	~	001	~	1	1	0	0	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
20	y	16		1	1	1	000	1	1	1	00		0	0	1	0	~	1	1	1	1	1	1	~	~	~	0	0	1	1	~	1	0	0	0	1	1	0	1
21	y	8		1	1	1	000	1	1	1	00		0	0	0	0	~	1	1	1	1	1	1	~	~	~	1	AD0	1	!AD0	1	1	0	!AD0	AD0	1	1	0	1
22																																							
23																																							
24	n		000	1	1	1	000	0	1	1	00		1	1	1	0	~	1	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
25	n		001	1	1	1	000	0	1	1	00		1	1	1	0	~	1	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
26	n		010	1	1	1	000	0	1	1	00		1	1	1	0	~	1	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
27	n		011	1	1	1	000	0	1	1	00		1	1	1	0	~	1	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
28	n		100	1	1	1	000	0	1	1	00		1	1	1	0	~	1	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
29	n		101	1	1	1	000	0	1	1	00		1	1	1	0	~	1	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
30	n		110	1	1	1	000	0	1	1	00		1	1	1	0	~	1	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
31																																							
32																																							
33																																							
34	n			1	1	1	000	1	1	1	00		1	0	1	0	~	1	1	1	0	1	0	0	1	1	1	1	~	1	~	1	1	1	1	1	1	0	1
35	n			1	1	1	000	1	1	1	00		1	0	1	0	~	1	1	1	0	1	0	1	1	1	1	1	~	1	~	1	1	1	1	1	1	0	1
36	n			1	1	1	000	1	1	1	00		1	0	1	0	~	1	1	1	0	1	0	0	1	0	1	1	~	1	~	1	1	1	1	1	1	0	1
37	n			1	1	1	000	1	1	1	00		1	0	1	0	~	1	1	1	0	1	0	1	0	1	1	1	~	1	~	1	1	1	1	1	1	0	1
38				1	1	1	000	1	1	1	00		1	0	1	0	~	1	1	1	0	1	0	1	1	0	1	1	~	1	~	1	1	1	1	1	1	0	1
39																																							
40																																							
41																																							
42	n			1	1	1	000	1	0	1	00		1	0	1	1	111	0	0	1	1	1	1	~	~	~	~	1	~	1	~	1	1	1	1	1	1	0	1
43	n			1	1	1	000	1	1	1	10		1	1	1	0	~	1	1	1	1	1	1	~	~	~	~	1	~	1	~	1	1	1	1	1	1	0	1
44	n			1	1	1	000	1	1	1	00		1	0	1	0	111	0	0	1	1	1	1	~	~	~	~	1	~	1	~	1	1	1	1	1	1	0	1
45	n			1	0	1	000	1	1	1	00		1	0	1	1	111	0	0	1	1	1	1	~	~	~	~	1	~	1	~	1	1	1	1	1	1	0	1
46	y	16		1	1	1	000	1	1	0	11		1	1	1	0	111	0	1	1	1	1	1	~	~	~	0	0	0	1	~	0	0	0	0	1	1	0	1
47																																							
48 (Cycle 1)	y	16		1	1	1	000	1	1	1	01		1	1	1	1	~	1	1	1	1	1	1	~	~	~	1	1	~	1	~	1	1	1	1	1	1	0	1
48 (Cycle 2)				1	1	1	000	1	1	0	00		1	0	1	1	~	1	1	1	1	1	1	~	~	~	0	0	1	1	~	1	0	0	0	1	1	0	1
49																																							
50																																							
51																																							
52 (Cycle 1)	y	16		1	0	1	000	1	1	0	11		1	1	1	1	111	0	1	1	1	1	1	~	~	~	0	0	0	1	~	0	0	0	0	1	1	0	1
52 (Cycle 2)				1	1	1	010	1	1	1	00		1	1	1	0	~	1	1	1	1	1	1	~	~	~	~	1	~	1	~	1	1	1	1	1	1	1	0
53																																							
54																																							
54																																							
55																																							
56 (Cycle 1)	y	16		1	1	1	000	1	1	1	01	1	1	1	1	1	~	1	1	1	1	1	1	~	~	~	~	1	~	1	~	1	1	1	1	1	1	0	1
56 (Cycle 2)				1	1	1	010	1	1	0	00	0	1	1	1	1	~	1	1	1	1	1	1	~	~	~	~	1	~	1	~	1	0	0	0	1	1	1	0

Opcode # (DEC)	Opcode # (HEX)	MSB(HEX)	Inst Size	ExecCycles 00_01,10 = 1 Cyc 11 = 2 Cyc	Opcode	Register C	Register B	Register A				Needed ALU	Actual ALU Op	Memory Access	8b or 16b Mem	Jump Condition	/IR2_OE	/PC_BBOE	/PC_ADOE	PC_MODE/3	/JMPINST	/SP_BBOE	/SP_ADOE	SP_MODE/2	/SP_D2B_OE	/B2A_OE	/RF_CEW	/RF_FHZ	/RF_BOE	ALU_FUNC/3	/ALU_OE	/ALU_FL	/ALU_WC	/ALU_CFLATCH	/SHFT_INVERTCARRY	/SFT_OE	SFT_DIR	/SFT_AR	/SFT_WC	/DBH_OE	/DBL_OE	DB_DIR (0=W)	H/2L_OE	H2L_DIR	MEM_R/W	/MEM_AS	/MEM_H_EN	/MEM_L_EN	/IR1_LATCH	/IR2_LATCH	/EXECUTE1	/EXECUTE2
57	#39	#72,#73	0	11	1001								b																																						
58	#3A	#74,#75	0	11	1010								b																																						
59	#3B	#76,#77	0	11	1011								b																																						
60	#3C	#78,#79	0	11	1100								b																																						
61	#3D	#7A,#7B	0	11	1101								b																																						
62	#3E	#7C,#7D	0	11	1110								b																																						
63	#3F	#7E,#7F	0	11	1111								b																																						
64	#40	#80,#81	1	00	0000								a																																						
65	#41	#82,#83	1	00	0001								a																																						
66	#42	#84,#85	1	00	0010	000	000	000	imm16	SUBI	**SUB a,imm16,c**	**Sub Reg-b from Reg-a, Store in Reg-c**	ZF,CF,OVF	a-b	a-b	n		0	1	1	000	1	1	1	00	1	0	1	1	001	0	0	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
67	#43	#86,#87	1	00	0011	000	000	000	imm16	SUBCI	**SUBC a,imm16,c**		ZF,CF,OVF	a-b	a-b	n		0	1	1	000	1	1	1	00	1	0	1	1	001	0	0	0	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
68	#44	#88,#89	1	00	0100	000	000	000	imm16	ADDI	**ADD a,imm16,c**	**Add Reg-a and Reg-b, Store in Reg-c**	ZF,CF,OVF	a+b	a+b	n		0	1	1	000	1	1	1	00	1	0	1	1	010	0	0	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
69	#45	#8A,#8B	1	00	0101	000	000	000	imm16	ADDCI	**ADDC a,imm16,c**		ZF,CF,OVF	a+b	a+b	n		0	1	1	000	1	1	1	00	1	0	1	1	010	0	0	0	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
70	#46	#8C,#8D	1	00	0110	000	000	000	imm16	XORI	**XOR a,imm16,c**	**Logical XOR Reg-a and Reg-b, Store in Reg-c**	ZF,CF/OVF=0	a $ b	a $ b	n		0	1	1	000	1	1	1	00	1	0	1	1	011	0	0	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
71	#47	#8E,#8F	1	00	0111								a $ b																																						
72	#48	#90,#91	1	00	1000	000	000	000	imm16	ORI	**OR a,imm16,c**	**Logical OR Reg-a and Reg-b, Store in Reg-c**	ZF,CF/OVF=0	a | b	a | b	n		0	1	1	000	1	1	1	00	1	0	1	1	100	0	0	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
73	#49	#92,#93	1	00	1001								a | b																																						
74	#4A	#94,#95	1	00	1010	000	000	000	imm16	ANDI	**AND a,imm16,c**	**Logical AND Reg-a and Reg-b, Store in Reg-c**	ZF,CF/OVF=0	a & b	a & b	n		0	1	1	000	1	1	1	00	1	0	1	1	101	0	0	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
75	#4B	#96,#97	1	00	1011					TESTI	TESTI a,imm16	Logical AND of Reg-a and imm16, no store	ZF,CF/OVF=0	a & b				0	1	1	000	1	1	1	00	1	1	1	1	101	0	0	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
76	#4C	#98,#99	1	00	1100								!b																																						
77	#4D	#9A,#9B	1	00	1101								!b																																						
78	#4E	#9C,#9D	1	00	1110								b																																						
79	#4F	#9E,#9F	1	00	1111								b																																						
80	#50	#A0,#A1	1	01	0000	000	000	aaa	imm16	STOREWI	STOREW a,[imm16]	Store word reg-a to address imm16	-	a	a	y	16	0	1	1	000	1	1	1	00	0	1	1	1	000	0	1	1	1	1	1	~	~	~		0	0	0	1	~	0	0	0	0	1	1	0	1
81	#51	#A2,#A3	1	01	0001	000	000	aaa	imm16	STOREBI	STOREB a,[imm16]	Store byte reg-a to address imm16	-	a	a	y	8	0	1	1	000	1	1	1	00	0	1	1	1	000	0	1	1	1	1	1	~	~	~		1	AD0	0	!AD0	0	0	0	!AD0	AD0	1	1	0	1
82	#52	#A4,#A5	1	01	0010					*CMPI*	*CMPI a,#imm16*	*Potential new instruction*			a-b			0	1	1	000	1	1	1	00	1	1	1	1		0	0	1	1	1	1	~	~	~		1	1	1	1	1	1	1	1	1	1	1	0	1
83	#53	#A6,#A7	1	01	0011								a-b																																						
84	#54	#A8,#A9	1	01	0100	ccc	000	000	imm16	LOADWI	LOADW [imm16],c	Load word from address imm16 into reg-c	-		a+b	y	16	0	1	1	000	1	1	1	00	0	0	1	1	~	1	1	1	1	1	1	~	~	~		0	0	1	1	~	1	0	0	0	1	1	0	1
85	#55	#AA,#AB	1	01	0101	ccc	000	000	imm16	LOADBI	LOADB [imm16],c	Load byte from address imm16 into reg-c	-		a+b	y	8	0	1	1	000	1	1	1	00	0	0	0	1	~	1	1	1	1	1	1	~	~	~		1	AD0	~	!AD0	1	1	0	!AD0	AD0	1	1	0	1
86	#56	#AC,#AD	1	01	0110								a $ b			n																																			
87	#57	#AE,#AF	1	01	0111								a $ b																																						
88	#58	#B0,#B1	1	01	1000	000	000	000	imm16	JMPI	JMP imm16	r=0: jump to imm16, r=1 jump to +imm16	-	-	a | b	n	000	0	1	1	000	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
89	#59	#B2,#B3	1	01	1001	000	000	000	imm16	JZI	JZ imm16	if Z r=0: jump to imm16, r=1 jump to +imm16	-	-	a | b	n	001	0	1	1	000	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
90	#5A	#B4,#B5	1	01	1010	000	000	000	imm16	JNZI	JNZ imm16	if NZ r=0: jump to imm16, r=1 jump to +imm16	-	-	a & b	n	010	0	1	1	000	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
91	#5B	#B6,#B7	1	01	1011	000	000	000	imm16	JCI	JC imm16	if C r=0: jump to imm16, r=1 jump to +imm16	-	-	a & b	n	011	0	1	1	000	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
92	#5C	#B8,#B9	1	01	1100	000	000	000	imm16	JNCI	JNC imm16	if NC r=0: jump to imm16, r=1 jump to +imm16	-	-	!b	n	100	0	1	1	000	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
93	#5D	#BA,#BB	1	01	1101	000	000	000	imm16	JVI	JV imm16	if V r=0: jump to imm16, r=1 jump to +imm16	-	-	!b	n	101	0	1	1	000	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
94	#5E	#BC,#BD	1	01	1110	000	000	000	imm16	JNVI	JNV imm16	if NV r=0: jump to imm16, r=1 jump to +imm16	-	-	b	n	110	0	1	1	000	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
95	#5F	#BE,#BF	1	01	1111				imm16		Jreserved		-		b																																					
96	#60	#C0,#C1	1	10	0000	000	000	aaa	imm16	STOREWRI	STOREW a,[+imm16]	Store word reg-a to address PC+imm16	-	-	a	y	16	0	1	0	100	1	1	1	00	1	1	1	1	000	0	1	1	1	1	1	~	~	~		0	0	0	1	~	0	0	0	0	1	1	0	1
97	#61	#C2,#C3	1	10	0001	000	000	aaa	imm16	STOREBRI	STOREB a,[+imm16]	Store byte reg-a to address PC+imm16	-	-	a	y	8	0	1	0	100	1	1	1	00	1	1	1	1	000	0	1	1	1	1	1	~	~	~		1	AD0	0	!AD0	0	0	0	!AD0	AD0	1	1	0	1
98	#62	#C4,#C5	1	10	0010								b																																						
99	#63	#C6,#C7	1	10	0011								b																																						
100	#64	#C8,#C9	1	10	0100	ccc	000	000	imm16	LOADWRI	LOADW [+imm16],c	Load word from address PC+imm16 into reg-c	-	-	b	y	16	0	1	0	100	1	1	1	00	1	0	1	1	~	1	1	1	1	1	1	~	~	~		0	0	1	1	~	1	0	0	0	1	1	0	1
101	#65	#CA,#CB	1	10	0101	ccc	000	000	imm16	LOADBRI	LOADB [+imm16],c	Load byte from address PC+imm16 into reg-c	-	-	b	y	8	0	1	0	100	1	1	1	00	1	0	0	1	~	1	1	1	1	1	1	~	~	~		1	AD0	~	!AD0	1	1	0	!AD0	AD0	1	1	0	1
102	#66	#CC,#CD	1	10	0110	ccc	000	000	imm16	LOADI	LOADI c,imm16	Load imm16 value into register C	-	-	b	n		0	1	1	000	1	1	1	00	1	0	1	1	111	0	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
103	#67	#CE,#CF	1	10	0111					LOADEA	LOADEA c,+imm16	Load address calculated as PC+imm16 into reg-c	-	-	b																																									
104	#68	#D0,#D1	1	10	1000			000	imm16	JMPIO	JMP +imm16	jump to PC+imm16 offset	-	-	b	n	000	0	1	1	001	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
105	#69	#D2,#D3	1	10	1001			000	imm16	JZRIO	JZ +imm16	if Z jump to PC+imm16	-	-	b	n	001	0	1	1	001	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
106	#6A	#D4,#D5	1	10	1010			000	imm16	JNZIO	JNZ +imm16	if NZ r=0: jump to imm16, r=1 jump to PC+imm16	-	-	b	n	010	0	1	1	001	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
107	#6B	#D6,#D7	1	10	1011			000	imm16	JCIO	JC +imm16	if C r=0: jump to imm16, r=1 jump to PC+imm16	-	-	b	n	011	0	1	1	001	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
108	#6C	#D8,#D9	1	10	1100			000	imm16	JNCIO	JNC +imm16	if NC r=0: jump to imm16, r=1 jump to PC+imm16	-	-	b	n	100	0	1	1	001	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
109	#6D	#DA,#DB	1	10	1101			000	imm16	JVIO	JV +imm16	if V r=0: jump to imm16, r=1 jump to PC+imm16	-	-	b	n	101	0	1	1	001	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
110	#6E	#DC,#DD	1	10	1110			000	imm16	JNVIO	JNV +imm16	if NV r=0: jump to imm16, r=1 jump to PC+imm16	-	-	b	n	110	0	1	1	001	0	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	0	1
111	#6F	#DE,#DF	1	10	1111				imm16		Jreserved		-		b																																					
112	#70	#E0,#E1	1	11	0000								a																																						
113	#71	#E2,#E3	1	11	0001								a																																						
114	#72	#E4,#E5	1	11	0010								b																																						
115	#73	#E6,#E7	1	11	0011								b																																						
116	#74	#E8,#E9	1	11	0100								b																																						
116													b																																						
117	#75	#EA,#EB	1	11	0101								b																																						
118	#76	#EC,#ED	1	11	0110			000	imm16		CALL [imm16]	Not possible with D->B Bridge inside IR2	-	-	b																																									
119	#77	#EE,#EF	1	11	0111								-	-	b																																				
120	#78	#F0,#F1	1	11	1000			000	imm16	CALLI	CALL imm16 (Cycle 1)		-	-	b	y	16	1	0	1	000	1	1	0	11	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	0	0	0	0	1	1	0	1
120								000	imm16		CALL imm16 (Cycle 2)		-	-				0	1	1	010	1	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	1	0
121	#79	#F2,#F3	1	11	1001			000	imm16	CALLIO	CALL +imm16 (Cycle 1)		-	-	b	y	16	1	0	1	000	1	1	0	11	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	0	0	0	0	1	1	0	1
121								000	imm16		CALL +imm16 (Cycle 2)		-	-				0	1	1	011	1	1	1	00	1	1	1	1	~	1	1	1	1	1	1	~	~	~		1	1	~	1	~	1	1	1	1	1	1	1	0
122	#7A	#F4,#F5	1	11	1010								b																																						
123	#7B	#F6,#F7	1	11	1011								b																																						


Opcode # (DEC)
Opcode # (HEX)
MSB(HEX)
Inst Size
ExecCycles
00_01_10 = 1 Cyc
11 = 2 Cyc
Opcode
Register C
Register B
Register A
Needed ALU
Actual ALU Op
Memory Access
8b or 16b Mem
Jump Condition
/IR2_OE
/PC_BBOE
/PC_ADOE
PC_MODE/3
/JMPINST
/SP_BBOE
/SP_ADOE
SP_MODE/2
/SP_D2B_OE
/B2A_OE
/RF_CEW
/RF_FHZ
/RF_BOE
ALU_FUNC/3
/ALU_OE
/ALU_FL
/ALU_WC
/ALU_CFLATCH
/SHFT_INVERTCARRY
/SFT_OE
SFT_DIR
/SFT_AR
/SFT_WC
/DBH_OE
/DBL_OE
DB_DIR (0=W)
H/2L_OE
H2L_DIR
MEM_R/W
/MEM_AS
/MEM_H_EN
/MEM_L_EN
/IR1_LATCH
/IR2_LATCH
/EXECUTE1
/EXECUTE2
124 #7C #F8,#F9 1 11 1100
125 #7D #FA,#FB 1 11 1101
126 #7E #FC,#FD 1 11 1110
127 #7F #FE,#FF 1 11 1111
NOP