Macroarchitecture

Property	Value

Memory organization	4096 Bytes. 12-bit word address.
	
	`0xxx xxxx xxxx: 0000-2047 (0000-07FF): 2K ROM (Program)`
	`10xx xxxx xxxx: 2048-3071 (0800-0BFF): 1K RAM (Data)`
	`1111 1111 1100: 4092 (0FFC): PIO Input data register`
	`1111 1111 1101: 4093 (0FFD): PIO Input status register`
	`1111 1111 1110: 4094 (0FFE): PIO output data register`
	`1111 1111 1111: 4095 (0FFF): PIO output status register`

Registers	PC: Program counter AC: Accumulator SP: Stack pointer
Adressing modes	Direct: Use address in low order 12 bits of opcode Indirect: Use address stored in AC Local: Address in low order 8 bits of opcode is relative to SP

Macroinstruction Set

Mnemonic	Instruction	Cycles	Binary	Hex	Meaning
LODD	Load direct	9	`0000 xxxx xxxx xxxx`	`0xxx`	ac := m[x]
STOD	Store direct	8	`0001 xxxx xxxx xxxx`	`1xxx`	m[x] := ac
ADDD	Add direct	9	`0010 xxxx xxxx xxxx`	`2xxx`	ac := ac + m[x]
SUBD	Subtract direct	10	`0011 xxxx xxxx xxxx`	`3xxx`	ac := ac - m[x]
JPOS	Jump positive	8	`0100 xxxx xxxx xxxx`	`4xxx`	if ac >= 0 the pc := x
JZER	Jump zero	8	`0101 xxxx xxxx xxxx`	`5xxx`	if ac = 0 then pc := x
JUMP	Jump unconditionally	7	`0110 xxxx xxxx xxxx`	`6xxx`	pc := x
LOCO	Load constant	7	`0111 xxxx xxxx xxxx`	`7xxx`	ac := x (0 <= x <= 4095)
LODL	Load local	10	`1000 xxxx xxxx xxxx`	`8xxx`	ac := m[sp + x]
STOL	Store local	9	`1001 xxxx xxxx xxxx`	`9xxx`	m[sp + x] := ac
ADDL	Add local	10	`1010 xxxx xxxx xxxx`	`Axxx`	ac := ac + m[sp + x]
SUBL	Subtract local	11	`1011 xxxx xxxx xxxx`	`Bxxx`	ac := ac - m[sp + x]
JNEG	Jump negative	8	`1100 xxxx xxxx xxxx`	`Cxxx`	if ac < 0 then pc := x
JNZE	Jump nonzero	8	`1101 xxxx xxxx xxxx`	`Dxxx`	if ac <> 0 then pc := x
CALL	Call procedure	9	`1110 xxxx xxxx xxxx`	`Exxx`	sp := sp - 1; m[sp] := pc; pc := x
PSHI	Push indirect	13	`1111 0000 0000 0000`	`F000`	sp := sp - 1; m[sp] := m[ac]
POPI	Pop indirect	13	`1111 0010 0000 0000`	`F200`	m[ac] := m[sp]; sp := sp + 1
PUSH	Push onto stack	12	`1111 0100 0000 0000`	`F400`	sp := sp - 1; m[sp] := ac
POP	Pop from stack	12	`1111 0110 0000 0000`	`F600`	ac := m[sp]; sp := sp + 1
RETN	Return	12	`1111 1000 0000 0000`	`F800`	pc := m[sp]; sp := sp + 1
SWAP	Swap ac, sp	12	`1111 1010 0000 0000`	`FA00`	tmp := ac; ac := sp; sp := tmp
INSP	Increment sp	11	`1111 1100 yyyy yyyy`	`FCyy`	sp := sp + y (0 <= y <= 255)
DESP	Decrement sp	14	`1111 1110 yyyy yyyy`	`FEyy`	sp := sp - y (0 <= y <= 255)
HLT	Halt program	11	`1111 1111 0000 0000`	`FF00`	Halt program

Microinstruction

Signal	Width	Pos	Factor	Mask	Description
ADDR	8	0	1	FF	Address
A	4	8	256	F00	Register number of A bus source
B	4	12	4096	F000	Register number of B bus source
C	4	16	65536	F0000	Register number of C bus destination
ENC	1	20	1048576	100000	Controls storing into the scratchpad: 0 = don't store. 1 = store.
WR	1	21	2097152	200000	Requests memory write: 0 = no write. 1 = write MBR to memory.
RD	1	22	4194304	400000	Request memory read: 0 = no read. 1 = load MBR from memory.
MAR	1	23	8388608	800000	Load MAR from B latch: 0 = don't load MAR. 1 = load MAR.
MBR	1	24	16777216	1000000	Load MBR from shifter: 0 = don't load MBR. 1 = load MBR.

Register

Name	Nummer	Beschreibung
PC	0	Program Counter
AC	1	Accumulator
SP	2	Stack Pointer
IR	3	Instruction Register. Holds the macroinstruction currently being executed.
TIR	4	Temporary copy of IR, used for decoding the opcode of the macroinstruction.
0	5	Constant
+1	6	Constant
-1	7	Constant
AMASK	8	Adress mask 0FFFh used to separate out opcode and address bits.
SMASK	9	Stack mask 00FFh used to isolate the 8-bit offset in INSP and DESP.
A	10	General purpose register for microprogram
B	11	General purpose register for microprogram
C	12	General purpose register for microprogram
D	13	General purpose register for microprogram
E	14	General purpose register for microprogram
F	15	General purpose register for microprogram

Microprogram

Row	Hex	Statements	Comment	Hexop	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR
0	00	mar := pc; rd;	main loop	10C00000	0	0	2	0	0	1	1	0	0	0	0	0	0
1	01	pc := pc + 1; rd;	increment pc	00506000	0	0	0	0	0	0	1	0	1	0	6	0	0
2	02	ir := mbr; if n then goto 28;	save, decode mbr	B013001C	1	1	2	0	0	0	0	0	1	3	0	0	28
3	03	tir := lshift(ir + ir); if n then goto 19;		24143313	0	1	0	2	0	0	0	0	1	4	3	3	19
4	04	tir := lshift(tir); if n then goto 11;	000x or 001x?	3414040B	0	1	2	2	0	0	0	0	1	4	0	4	11
5	05	alu := tir; if n then goto 9;	0000 or 0001?	30000409	0	1	2	0	0	0	0	0	0	0	0	4	9
6	06	mar := ir; rd;	0000 = **LODD**	10C03000	0	0	2	0	0	1	1	0	0	0	3	0	0
7	07	rd;		10400000	0	0	2	0	0	0	1	0	0	0	0	0	0
8	08	ac := mbr; goto 0;		F0110300	1	3	2	0	0	0	0	0	1	1	0	3	0
9	09	mar := ir; mbr := ac; wr;	0001 = **STOD**	11A03100	0	0	2	0	1	1	0	1	0	0	3	1	0
10	0A	wr; goto 0;		70200000	0	3	2	0	0	0	0	1	0	0	0	0	0
11	0B	alu := tir; if n then goto 15;	0010 or 0011?	3000040F	0	1	2	0	0	0	0	0	0	0	0	4	15
12	0C	mar := ir; rd;	0010 = **ADD**	10C03000	0	0	2	0	0	1	1	0	0	0	3	0	0
13	0D	rd;		10400000	0	0	2	0	0	0	1	0	0	0	0	0	0
14	0E	ac := mbr + ac; goto 0;		E0111000	1	3	0	0	0	0	0	0	1	1	1	0	0
15	0F	mar := ir; rd;	0011 = **SUBD**	10C03000	0	0	2	0	0	1	1	0	0	0	3	0	0
16	10	ac := ac + 1; rd;	x-y = x + 1 + not y	00516100	0	0	0	0	0	0	1	0	1	1	6	1	0
17	11	a := inv(mbr)		981A0000	1	0	3	0	0	0	0	0	1	10	0	0	0
18	12	ac := ac + a; goto 0;		6011A100	0	3	0	0	0	0	0	0	1	1	10	1	0
19	13	tir := lshift(tir); if n then goto 25;	010x or 011x?	34140419	0	1	2	2	0	0	0	0	1	4	0	4	25
20	14	alu := tir; if n then goto 23;	0100 or 0101?	30000417	0	1	2	0	0	0	0	0	0	0	0	4	23
21	15	alu := ac; if n then goto 0;	0100 = **JPOS**	30000100	0	1	2	0	0	0	0	0	0	0	0	1	0
22	16	pc := band(ir, amask); goto 0;	perform jump	68108300	0	3	1	0	0	0	0	0	1	0	8	3	0
23	17	alu := ac; if z then goto 22;	0101 = **JZER**	50000116	0	2	2	0	0	0	0	0	0	0	0	1	22
24	18	goto 0;	jump failed	60000000	0	3	0	0	0	0	0	0	0	0	0	0	0
25	19	alu := tir; if n then goto 27;	0110 or 0111?	3000041B	0	1	2	0	0	0	0	0	0	0	0	4	27
26	1A	pc := band(ir, amask); goto 0;	0110 = **JUMP**	68108300	0	3	1	0	0	0	0	0	1	0	8	3	0
27	1B	ac := band(ir, amask); goto 0;	0111 = **LOCO**	68118300	0	3	1	0	0	0	0	0	1	1	8	3	0
28	1C	tir := lshift(ir + ir); if n then goto 40;	10xx or 11xx?	24143328	0	1	0	2	0	0	0	0	1	4	3	3	40
29	1D	tir := lshift(tir); if n then goto 35;	100x or 101x?	34140423	0	1	2	2	0	0	0	0	1	4	0	4	35
30	1E	alu := tir; if n then goto 33;	1000 or 1001?	30000421	0	1	2	0	0	0	0	0	0	0	0	4	33
31	1F	a := ir + sp;	1000 = **LODL**	001A2300	0	0	0	0	0	0	0	0	1	10	2	3	0

Signal	Width	Pos	Factor	Mask	Description
SH	2	25	33554432	6000000	Shifter function: 0 = no shift 1 = right 2 = left.
ALU	2	27	134217728	18000000	ALU function: 0 = A + B 1 = A and B 2 = A 3 = not A.
COND	2	29	536870912	40000000	Condition: 0 = no jump 1 = Jump if N = 1 2 = Jump if Z = 1 3 = Jump always
AMUX	1	30	1073741824	80000000	Controls left ALU input: 0 = A latch 1 = MBR

Example: Sum of all numbers between I and L-1

Row	Label	Command	Hex	Comment
0		LOCO 0	7000	`Initialize index I=0 at 0800`
1		STOD I	1800	
2		LOCO 0x65	7065	`Initialize limit L=101 at 0801`
3		STOD L	1801	
4		LOCO 0	7000	`Initialize sum S=0 at 0802`
5		STOD S	1802	
6		LOCO 1	7001	`Initialize constant ONE=1 at 0803`
7		STOD ONE	1803	
8	**LOOP**	LODD L	0801	`IF (I == L) THEN GOTO END`
9		SUBD I	3800	
A		JZER END	5013	
B		LODD S	0802	`S = S + I`
C		ADDD I	2800	
D		STOD S	1802	
E		STOD OUT	1FFE	`Output S`
F		LODD I	0800	`I = I + 1`
10		ADDD ONE	2803	
11		STOD I	1800	
12		JUMP LOOP	6008	`GOTO LOOP`
13	**END**	HALT	FF00	`Stop program`

Row	Hex	Statements	Comment	Hexop	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR
32	20	mar := a; rd; goto 7;		70C0A007	0	3	2	0	0	1	1	0	0	0	10	0	7
33	21	a := ir + sp;	1001 = **STOL**	001A2300	0	0	0	0	0	0	0	0	1	10	2	3	0
34	22	mar := a; mbr := ac; wr; goto 10;		71A0A10A	0	3	2	0	1	1	0	1	0	0	10	1	10
35	23	alu := tir; if n then goto 38;	1010 or 1011?	30000426	0	1	2	0	0	0	0	0	0	0	0	4	38
36	24	a := ir + sp;	1010 = **ADDL**	001A2300	0	0	0	0	0	0	0	0	1	10	2	3	0
37	25	mar := a; rd; goto 13;		70C0A00D	0	3	2	0	0	1	1	0	0	0	10	0	13
38	26	a := ir + sp;	1011 = **SUBL**	001A2300	0	0	0	0	0	0	0	0	1	10	2	3	0
39	27	mar := a; rd; goto 16;		70C0A010	0	3	2	0	0	1	1	0	0	0	10	0	16
40	28	tir := lshift(tir); if n then goto 46	110x or 111x?	3414042E	0	1	2	2	0	0	0	0	1	4	0	4	46
41	29	alu := tir; if n then goto 44;	1100 or 1101?	3000042C	0	1	2	0	0	0	0	0	0	0	0	4	44
42	2A	alu := ac; if n then goto 22;	1100 = **JNEG**	30000116	0	1	2	0	0	0	0	0	0	0	0	1	22
43	2B	goto 0;		60000000	0	3	0	0	0	0	0	0	0	0	0	0	0
44	2C	alu := ac; if z then goto 0;	1101 = **JNZE**	50000100	0	2	2	0	0	0	0	0	0	0	0	1	0
45	2D	pc := band(ir, amask); goto 0;		68108300	0	3	1	0	0	0	0	0	1	0	8	3	0
46	2E	tir := lshift(tir); if n then goto 50;		34140432	0	1	2	2	0	0	0	0	1	4	0	4	50
47	2F	sp := sp + (-1)	1110 = **CALL**	00127200	0	0	0	0	0	0	0	0	1	2	7	2	0
48	30	mar := sp; mbr := pc; wr;	save return add	11A02000	0	0	2	0	1	1	0	1	0	0	2	0	0
49	31	pc := band(ir, amask); wr; goto 0;	jump to subroutine	68308300	0	3	1	0	0	0	0	1	1	0	8	3	0
50	32	tir := lshift(tir); if n then goto 65;	1111, examine add	34140441	0	1	2	2	0	0	0	0	1	4	0	4	65
51	33	tir := lshift(tir); if n then goto 59;		3414043B	0	1	2	2	0	0	0	0	1	4	0	4	59
52	34	alu := tir; if n then goto 56;		30000438	0	1	2	0	0	0	0	0	0	0	0	4	56
53	35	mar := ac; rd;	1111 000 = **PSHI**	10C01000	0	0	2	0	0	1	1	0	0	0	1	0	0
54	36	sp := sp + (-1); rd;		00527200	0	0	0	0	0	0	1	0	1	2	7	2	0
55	37	mar := sp; wr; goto 10;		70A0200A	0	3	2	0	0	1	0	1	0	0	2	0	10
56	38	mar := sp; sp := sp + 1; rd;	1111 001 = **POPI**	00D22600	0	0	0	0	0	1	1	0	1	2	2	6	0
57	39	rd;		10400000	0	0	2	0	0	0	1	0	0	0	0	0	0
58	3A	mar := ac; wr; goto 10;		70A0100A	0	3	2	0	0	1	0	1	0	0	1	0	10
59	3B	alu := tir; if n then goto 62;		3000043E	0	1	2	0	0	0	0	0	0	0	0	4	62
60	3C	sp := sp + (-1)	1111 010 = **PUSH**	00127200	0	0	0	0	0	0	0	0	1	2	7	2	0
61	3D	mar := sp; mbr := ac; wr; goto 10;		71A0210A	0	3	2	0	1	1	0	1	0	0	2	1	10
62	3E	mar := sp; sp := sp + 1; rd;	1111 011 = **POP**	00D22600	0	0	0	0	0	1	1	0	1	2	2	6	0
63	3F	rd;		10400000	0	0	2	0	0	0	1	0	0	0	0	0	0
64	40	ac := mbr; goto 0;		F0110000	1	3	2	0	0	0	0	0	1	1	0	0	0
65	41	tir = lshift(tir); if n then goto 73;		34140449	0	1	2	2	0	0	0	0	1	4	0	4	73
66	42	alu := tir; if n then goto 70;		30000446	0	1	2	0	0	0	0	0	0	0	0	4	70
67	43	mar := sp; sp := sp + 1; rd;	1111 100 = **RETN**	00D22600	0	0	0	0	0	1	1	0	1	2	2	6	0
68	44	rd;		10400000	0	0	2	0	0	0	1	0	0	0	0	0	0
69	45	pc := mbr; goto 0;		F0100000	1	3	2	0	0	0	0	0	1	0	0	0	0
70	46	a := ac;	1111 101 = **SWAP**	101A0100	0	0	2	0	0	0	0	0	1	10	0	1	0
71	47	ac := sp;		10110200	0	0	2	0	0	0	0	0	1	1	0	2	0
72	48	sp := a; goto 0;		70120A00	0	3	2	0	0	0	0	0	1	2	0	10	0
73	49	tir = lshift(tir); if n then goto 76;	1111 110x or 1111 111x?	3414044C	0	1	2	2	0	0	0	0	1	4	0	4	76
74	4A	a := band(ir, smask);	1111 110 = **INSP**	081A9300	0	0	1	0	0	0	0	0	1	10	9	3	0
75	4B	sp := sp + a; goto 0;		6012A200	0	3	0	0	0	0	0	0	1	2	10	2	0
76	4C	alu := tir; if n the goto 80;	1111 1110 or 1111 1111?	30000450	0	1	2	0	0	0	0	0	0	0	0	4	80
77	4D	a := band(ir, smask);	1111 111 = **DESP**	081A9300	0	0	1	0	0	0	0	0	1	10	9	3	0
78	4E	a := inv(a);		181A0A00	0	0	3	0	0	0	0	0	1	10	0	10	0
79	4F	a := a + 1; goto 75;		601A6A4B	0	3	0	0	0	0	0	0	1	10	6	10	75
80	50	goto 80;	1111 1111 = **HALT**	60000050	0	3	0	0	0	0	0	0	0	0	0	0	80