

Instruction index

opcode	operands	description	p.
<b>(Load instructions)</b>			
LDI	immed,SRd	immed -> SRd	1
LD	addr,SRd	M[addr] -> SRd	1
LDB	SRb,disp,SRd	M[SRb+disp] -> SRd	2
LDBI	SRb,SRi,SRd	M[SRb+SRi] -> SRd	2
LDBS	SRb,sft,SRd	M[SRb>>sft] -> SRd	3
LDPCB	disp,SRd	M[PC+disp] -> SRd	3
LDPCBI	SRi,SRd	M[PC+SRi] -> SRd	4
<b>(Store instructions)</b>			
ST	addr,SRs	SRs -> M[addr]	4
STB	SRb,disp,SRs	SRs -> M[SRb+disp]	5
STBI	SRb,SRi,SRs	SRs -> M[SRb+SRi]	5
STBS	SRb,sft,SRs	SRs -> M[SRb>>sft]	6
STPCB	disp,SRs	SRs -> M[PC+disp]	6
STPCBI	SRi,SRs	SRs -> M[PC+SRi]	7
<b>(Integer Operations)</b>			
ADD	SRs1,SRs2,SRd	SRs2 + SRs1 -> SRd	8
ADDI	immed,SRs2,SRd	SRs2 + immed -> SRd	8
SUB	SRs1,SRs2,SRd	SRs2 - SRs1 -> SRd	9
SUBI	immed,SRs2,SRd	SRs2 - immed -> SRd	9
MUL	SRs1,SRs2,SRd	SRs2 x SRs1 -> SRd	10
MULI	immed,SRs2,SRd	SRs2 x immed -> SRd	10
DIV	SRs1,SRs2,SRd	SRs2 ÷ SRs1 -> SRd	11
DIVI	immed,SRs2,SRd	SRs2 ÷ immed -> SRd	11
MOD	SRs1,SRs2,SRd	SRs2 % SRs1 -> SRd	12
MODI	immed,SRs2,SRd	SRs2 % immed -> SRd	12
RSUB	SRs1,SRs2,SRd	SRs1 - SRs2 -> SRd	13
RSUBI	immed,SRs2,SRd	immed - SRs2 -> SRd	13
UMUL	SRs1,SRs2,SRd	SRs2 x SRs1 -> SRd (unsigned)	14
UMULI	immed,SRs2,SRd	SRs2 x immed -> SRd (unsigned)	14
UDIV	SRs1,SRs2,SRd	SRs2 ÷ SRs1 -> SRd (unsigned)	15
UDIVI	immed,SRs2,SRd	SRs2 ÷ immed -> SRd (unsigned)	15
<b>(Logical Operations)</b>			
AND	SRs1,SRs2,SRd	SRs2 & SRs1 -> SRd	16
ANDI	immed,SRs2,SRd	SRs2 & immed -> SRd	16
OR	SRs1,SRs2,SRd	SRs2   SRs1 -> SRd	17
ORI	immed,SRs2,SRd	SRs2   immed -> SRd	17
XOR	SRs1,SRs2,SRd	SRs2 ^ SRs1 -> SRd	18
XORI	immed,SRs2,SRd	SRs2 ^ immed -> SRd	18
NOT	SRs1,SRd	~SRs1 -> SRd	7
SRL	SRs1,SRs2,SRd	SRs2 >> SRs1 -> SRd (logical shift)	19
SRLI	immed,SRs2,SRd	SRs2 >> immed -> SRd (logical shift)	19
SLL	SRs1,SRs2,SRd	SRs2 << SRs1 -> SRd (logical shift)	20

**Naming conventions**

- SRs : Number of Scalar Register (source) , 0 - 255
- SRs' : Number of Scalar Register (source) , 0 - 128
- SRs1 : Number of Scalar Register (source-1) , 0 - 255
- SRs2 : Number of Scalar Register (source-2) , 0 - 255
- SRb : Number of Scalar Register (base register) , 0 - 255
- SRi : Number of Scalar Register (index register) , 0 - 255
- SRd : Number of Scalar Register (destination), 0 - 255
- SRp : Number of Scalar Register (VR pointer), 0 - 255
- VRs : Number of Vector Register (source) , 0 - 63
- VRs1 : Number of Vector Register (source-1) , 0 - 63
- VRs2 : Number of Vector Register (source-2) , 0 - 63
- VRd : Number of Vector Register (destination) , 0 - 63
- VRI : Number of Vector Register (list vector) , 0 - 63
- CNs[12] : Number of Chaining Network (source[12]) , 0 - 7
- CNd : Number of Chaining Network (destination) , 0 - 7
- VMR : Number of Vector Mask Register , 0 - 7 , 0 = VMR not used
- VMRs[12]: Number of Vector Mask Register (source[12]) , 0 - 7 , VMR0 = ALL TRUE
- VMRd : Number of Vector Mask Register (destination) , 1 - 7
- Immed: Immediate value
- Addr : Immediate value (address)
- Disp : Immediate value (displacement)
- Sft : Immediate value (shift amount)
- M[addr] : Content of Memory which address is addr
- PC : Program counter
- ZR : Zero Register (SR0)
- FR : Flag Register
- SP : Stack Pointer
- FP : Frame Pointer
- LR : Link Register

opcode	operands	description	p.
SLLI	immed,SRs2,SRd	SRs2 << immed -> SRd (logical shift)	20
SRA	SRs1,SRs2,SRd	SRs2 >> SRs1 -> SRd (arithmetic shift)	21
SRAI	immed,SRs2,SRd	SRs2 >> immed -> SRd (arithmetic shift)	21
SLA	SRs1,SRs2,SRd	SRs2 << SRs1 -> SRd (arithmetic shift)	22
SLAI	immed,SRs2,SRd	SRs2 << immed -> SRd (arithmetic shift)	22
(Float operations)			
FADD	SRs1,SRs2,SRd	SRs2 + SRs1 -> SRd	23
FSUB	SRs1,SRs2,SRd	SRs2 - SRs1 -> SRd	23
FMUL	SRs1,SRs2,SRd	SRs2 x SRs1 -> SRd	24
FDIV	SRs1,SRs2,SRd	SRs2 ÷ SRs1 -> SRd	24
INT	SRs1,SRd	(int)SRs1 -> SRd	25
FLT	SRs1,SRd	(float)SRs1 -> SRd	25
(Jump Instructions)			
JMP	disp	(cond = P,M,Z,NP,NM,NZ,GT,GE,LT,LE,EQ,NE) PC + disp -> PC	26
JR	disp	PC + disp -> PC	26
JC	cond,disp	if cond then PC + disp -> PC	26
JRC	cond,disp	if cond then PC + disp -> PC	26
JB	SRb,disp	SRb + disp -> PC	26
JBC	cond,SRb,disp	if cond then SRb + disp -> PC	26
JMPL	disp	PC + 2 -> LR; PC + disp -> PC	27
JRL	disp	PC + 2 -> LR; PC + disp -> PC	27
JCL	cond,addr	if cond then PC + 2 -> LR; PC + disp -> PC	27
JRCL	cond,addr	if cond then PC + 2 -> LR; PC + disp -> PC	27
JBL	SRb,disp	PC + 2 -> LR; SRb+disp -> PC	27
JBCL	cond,SRb,disp	if cond then PC + 2 -> LR; SRb+disp -> PC	27
(Compare&jump Instructions)			
CJP	opr,SRs1,SRs2,disp	(opr=SUB,RSUB,AND,OR) if (SRs2 opr SRs1) > 0 then PC + disp -> PC	28
CJM	opr,SRs1,SRs2,disp	if (SRs2 opr SRs1) < 0 then PC + disp -> PC	28
CJZ	opr,SRs1,SRs2,disp	if (SRs2 opr SRs1) == 0 then PC + disp -> PC	28
CJNZ	opr,SRs1,SRs2,disp	if (SRs2 opr SRs1) != 0 then PC + disp -> PC	28
CJPI	opr,immed,SRs2,disp	if (SRs2 opr immed) > 0 then PC + disp -> PC	28
CJMI	opr,immed,SRs2,disp	if (SRs2 opr immed) < 0 then PC + disp -> PC	28
CJZI	opr,immed,SRs2,disp	if (SRs2 opr immed) == 0 then PC + disp -> PC	28
CJNZI	opr,immed,SRs2,disp	if (SRs2 opr immed) != 0 then PC + disp -> PC	28
(Vector operations)			
VFADD	VRs1,VRs2,VRd,VMR	(VMR = Vector Mask Reg.) VRs2 + VRs1 -> VRd (float)	29
VFSUB	VRs1,VRs2,VRd,VMR	VRs2 - VRs1 -> VRd (float)	30
VFMUL	VRs1,VRs2,VRd,VMR	VRs2 x VRs1 -> VRd (float)	31
VFDIV	VRs1,VRs2,VRd,VMR	VRs2 ÷ VRs1 -> VRd (float)	32
VADD	VRs1,VRs2,VRd,VMR	VRs2 + VRs1 -> VRd (int)	33
VSUB	VRs1,VRs2,VRd,VMR	VRs2 - VRs1 -> VRd (int)	34
VMUL	VRs1,VRs2,VRd,VMR	VRs2 x VRs1 -> VRd (int)	35
VDIV	VRs1,VRs2,VRd,VMR	VRs2 ÷ VRs1 -> VRd (int)	36

opcode	operands	description	p.
VAND	VRs1,VRs2,VRd,VMR	VRs2 & VRs1 -> VRd (logic)	37
VOR	VRs1,VRs2,VRd,VMR	VRs2   VRs1 -> VRd (logic)	38
VXOR	VRs1,VRs2,VRd,VMR	VRs2 ^ VRs1 -> VRd (logic)	39
VNOT	VRs1,VRd,VMR	~VRs1 -> VRd (logic)	40
(Chaining vector operations)			
CVFADD	src1,src2,dstn,VMR	(VMR=Vector Mask Reg.; src1,src2,dstn=VR/CN) src2 + src1 -> dstn (float)	29
CVFSUB	src1,src2,dstn,VMR	src2 - src1 -> dstn (float)	30
CVFMUL	src1,src2,dstn,VMR	src2 x src1 -> dstn (float)	31
CVFDIV	src1,src2,dstn,VMR	src2 ÷ src1 -> dstn (float)	32
CVADD	src1,src2,dstn,VMR	src2 + src1 -> dstn (int)	33
CVSUB	src1,src2,dstn,VMR	src2 - src1 -> dstn (int)	34
CVMJL	src1,src2,dstn,VMR	src2 x src1 -> dstn (int)	35
CVDIV	src1,src2,dstn,VMR	src2 ÷ src1 -> dstn (int)	36
CVAND	src1,src2,dstn,VMR	src2 & src1 -> dstn (logic)	37
CVOR	src1,src2,dstn,VMR	src2   src1 -> dstn (logic)	38
CVXOR	src1,src2,dstn,VMR	src2 ^ src1 -> dstn (logic)	39
CVNOT	src1,dstn,VMR	~src1 -> dstn (logic)	40
(V&S mixed operations)			
VBCAST	SRs',VRd	SRs' -> VRd (broadcast)	41
VSRL	SRs',VRs,VRd	VRs >> SRs' -> VRd	42
VSLL	SRs',VRs,VRd	VRs << SRs' -> VRd	42
(Vector Load/Store)			
VLIR	VRd,SRb,SRI,VMR	for(i=0;i<128;i++) M[SRb+SRi] -> VRd[i]	43
VLIJ	VRd,SRb,immed,VMR	for(i=0;i<128;i++) M[SRb+immed+i] -> VRd[i]	43
VSIR	VRs,SRb,SRI,VMR	for(i=0;i<128;i++) VRs[i] -> M[SRb+SRi]x[i]	44
VSII	VRs,SRb,immed,VMR	for(i=0;i<128;i++) VRs[i] -> M[SRb+immed+i]x[i]	44
(List Vector L/S)			
LVLD	VRi,VRd,VMR	for(i=0;i<128;i++) M[VRi] -> VRd[i]	45
LVST	VRi,VRs,VMR	for(i=0;i<128;i++) VRs[i] -> M[VRi] -> VRd[i]	45
LVBLD	VRi,VRd,SRb	for(i=0;i<128;i++) M[VRi]x[SRb] -> VRd[i]	46
LVBST	VRi,VRs,SRb	for(i=0;i<128;i++) VRs[i] -> M[VRi]x[SRb]	46
(VMR set)			
VMRSETI	cond,VRs,VMRd	(cond=Z,NZ,P,NP,M,NM) for(i=0;i<128;i++)	47
VMRSETF	cond,VRs,VMRd	if cond_int(VRs[i]) then TRUE -> VMRd[i] if cond_float(VRs[i]) then TRUE -> VMRd[i]	47
(VMR operations)			
VMRAND	VMRs1,VMRs2,VMRd	VMRs1 & VMRs2 -> VMRd	48
VMROR	VMRs1,VMRs2,VMRd	VMRs1   VMRs2 -> VMRd	48
VMRXOR	VMRs1,VMRs2,VMRd	VMRs1 ^ VMRs2 -> VMRd	49
VMRNOT	VMRs1,VMRd	~VMRs1 -> VMRd	49

opcode	operands	description	p.
(Vector INT/FLT conversion)			
VINT	VRs,VRd,VMIR	(int)VRs -> VRd	50
VFLT	VRs,VRd,VMIR	(float)VRs -> VRd	50
(VR <-> SR move)			
MOVWS	VRs,SRp,SRd	VRs[SRp] -> SRd	51
MOVSV	VRd,SRp,SRs	SRs -> VRd[SRp]	51
(Synthetic Instructions)			
MOV	SRs,SRd	SRs -> SRd (= OR SRs,ZR,SRd)	
CMP	SRs1,SRs2	compare SRs1,SRs2 (= SUB SRs1,SRs2,ZR)	
CMPI	immed,SRs2	compare Immed,SRs2 (= SUBI Immed,SRs2,ZR)	
RCMP	SRs1,SRs2	reversecompareSRs1,SRs2(=RSUBSRs1,SRs2,ZR)	
RCMPI	immed,SRs2,SRd	reversecompareImmed,SRs2(=RSUBIImmed,SRs2,ZR)	
INC	SRd	increment SRd (= ADDI 1,SRd,SRd)	
DEC	SRd	decrement SRd (= SUBI 1,SRd,SRd)	
CLR	SRd	clear SRd (= ORI ZR,ZR,SRd)	
NOP		no operation (= LDI 0,ZR)	
HLT		halt CPU (= LDI haltflag,flagreg)	
(Pseudo Instructions)			
ORG	addr	アセンブラにて現在のアドレスを設定	
DS	expr	expr分のスペースを確保(0で初期化)	
DW	expr[,expr...]	1word分のスペースを確保し、exprの値で初期化	
.data		exprには整数式の他、浮動小数点定数、文字列が使用可能	
.code		アセンブラのモードをデータモードへ、ラベルの制限無し	
LABEL	EQU expr	アセンブラのモードをコードモードへ、ラベルは4で割り切れる値のみ exprの値をLABELに代入	