ALPHA

CODE CARD

Index

digital™

Instruction set (alphabetic order)

Opcode	Mnemonic	Operands	Description
15.080	ADDF	Fa.rf,Fb.rf,Fc.wq	Fc ← Fav + Fbv
15.0A0	ADDG	Fa.rg,Fb.rg,Fc.wq	Fc ← Fav + Fbv
10.00	ADDL	Ra.rq,Rb.rq,Rc.wq	Rc ← SEXT({Rav + Rbv}<31:0>)
10.00	ADDL	Ra.rq,#b.ib,Rc.wq	Rc ← SEXT({Rav + Rbv}<31:0>)
10.20	ADDQ	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav + Rbv
10.20	ADDQ	Ra.rq,#b.ib,Rc.wq	Rc ← Rav + Rbv
16.080	ADDS	Fa.rs,Fb.rs,Fc.wq	Fc ← Fav + Fbv
16.0A0	ADDT	Fa.rt,Fb.rt,Fc.wq	Fc ← Fav + Fbv
11.00	AND	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav AND Rbv
39	BEQ	Ra.rq,disp.al	If Rav = 0 Then PC ← PC + {4*SEXT(disp)}
3E	BGE	Ra.rq,disp.al	If Rav ≥ 0 Then PC ← PC + {4*SEXT(disp)}
3F	BGT	Ra.rq,disp.al	If Rav > 0 Then PC ← PC + {4*SEXT(disp)}
11.08	BIC	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav AND NOT Rbv
38	BLBC	Ra.rq,disp.al	If Rav<0> = 0 Then PC ← PC + {4*SEXT(disp)}
3C	BLBS	Ra.rq,disp.al	If Rav<0> = 1 Then PC ← PC + {4*SEXT(disp)}
3B	BLE	Ra.rq,disp.al	If Rav ≤ 0 Then PC ← PC + {4*SEXT(disp)}
3A	BLT	Ra.rq,disp.al	If Rav < 0 Then PC ← PC + {4*SEXT(disp)}
3D	BNE	Ra.rq,disp.al	If Rav ≠ 0 Then PC ← PC + {4*SEXT(disp)}
30	BR	Ra.wq,disp.al	Ra ← PC; PC ← PC + {4*SEXT(disp)}

Opcode	Mnemonic	Operands	Description
34	BSR	Ra.wq,disp.al	Ra ← PC; PC ← PC + {4*SEXT(disp)}
00	CALL_PAL	fnc.ir	Trap to PAL code
11.24	CMOVEQ	Ra.rq,Rb.rq,Rc.wq	If Rav = 0 Then Rc ← Rbv
11.24	CMOVEQ	Ra.rq,#b.ib,Rc.wq	If Rav = 0 Then Rc ← Rbv
11.46	CMOVGE	Ra.rq,Rb.rq,Rc.wq	If Rav ≥ 0 Then Rc ← Rbv
11.46	CMOVGE	Ra.rq,#b.ib,Rc.wq	If Rav ≥ 0 Then Rc ← Rbv
11.66	CMOVGT	Ra.rq,Rb.rq,Rc.wq	If Rav > 0 Then Rc ← Rbv
11.66	CMOVGT	Ra.rq,#b.ib,Rc.wq	If Rav > 0 Then Rc ← Rbv
11.14	CMOVLBC	Ra.rq,Rb.rq,Rc.wq	If Rav<0> = 0 Then Rc ← Rbv
11.14	CMOVLBC	Ra.rq,#b.ib,Rc.wq	If Rav<0> = 0 Then Rc ← Rbv
11.16	CMOVLBS	Ra.rq,Rb.rq,Rc.wq	If Rav<0> = 1 Then Rc ← Rbv
11.16	CMOVLBS	Ra.rq,#b.ib,Rc.wq	If Rav<0> = 1 Then Rc ← Rbv
11.64	CMOVLE	Ra.rq,Rb.rq,Rc.wq	If Rav ≤ 0 Then Rc ← Rbv
11.64	CMOVLE	Ra.rq,#b.ib,Rc.wq	If Rav ≤ 0 Then Rc ← Rbv
11.44	CMOVLT	Ra.rq,Rb.rq,Rc.wq	If Rav < 0 Then Rc ← Rbv
11.44	CMOVLT	Ra.rq,#b.ib,Rc.wq	If Rav < 0 Then Rc ← Rbv
11.26	CMOVNE	Ra.rq,Rb.rq,Rc.wq	If Rav ≠ 0 Then Rc ← Rbv
11.26	CMOVNE	Ra.rq,#b.ib,Rc.wq	If Rav ≠ 0 Then Rc ← Rbv
10.0F	CMPBGE	Ra.rq,Rb.rq,Rc.wq	Rc ← Bytewise compare mask of {Rav ≥ Rbv}
10.0F	CMPBGE	Ra.rq,#b.ib,Rc.wq	Rc ← Bytewise compare mask of {Rav ≥ Rbv}
10.2D	CMPEQ	Ra.rq,Rb.rq,Rc.wq	If Rav = Rbv Then Rc ← 1 Else Rc ← 0

Opcode	Mnemonic	Operands	Description
10.2D	CMPEQ	Ra.rq,#b.ib,Rc.wq	If Rav = Rbv Then Rc ← 1 Else Rc ← 0
15.0A5	CMPGEQ	Fa.rg,Fb.rg,Fc.wq	If Fav = Fbv Then Fc ← 0.5 Else Fc ← 0.0
15.0A7	CMPGLE	Fa.rg,Fb.rg,Fc.wq	If Fav ≤ Fbv Then Fc ← 0.5 Else Fc ← 0.0
15.0A6	CMPGLT	Fa.rg,Fb.rg,Fc.wq	If Fav < Fbv Then Fc ← 0.5 Else Fc ← 0.0
10.6D	CMPLE	Ra.rq,Rb.rq,Rc.wq	If Rav ≤ Rbv Then Rc ← 1 Else Rc ← 0
10.6D	CMPLE	Ra.rq,#b.ib,Rc.wq	If Rav ≤ Rbv Then Rc ← 1 Else Rc ← 0
10.4D	CMPLT	Ra.rq,Rb.rq,Rc.wq	If Rav < Rbv Then Rc ← 1 Else Rc ← 0
10.4D	CMPLT	Ra.rq,#b.ib,Rc.wq	If Rav < Rbv Then Rc ← 1 Else Rc ← 0
16.0A5	CMPTEQ	Fa.rt,Fb.rt,Fc.wq	If Fav = Fbv Then Fc ← 0.5 Else Fc ← 0.0
16.0A7	CMPTLE	Fa.rt,Fb.rt,Fc.wq	If Fav ≤ Fbv Then Fc ← 0.5 Else Fc ← 0.0
16.0A6	CMPTLT	Fa.rt,Fb.rt,Fc.wq	If Fav < Fbv Then Fc ← 0.5 Else Fc ← 0.0
16.0A4	CMPTUN	Fa.rt,Fb.rt,Fc.wq	If Fav Unordered Fbv Then Fc ← 0.5 Else Fc ← 0.0
10.3D	CMPULE	Ra.rq,Rb.rq,Rc.wq	If Rav U≤ Rbv Then Rc ← 1 Else Rc ← 0
10.3D	CMPULE	Ra.rq,#b.ib,Rc.wq	If Rav U≤ Rbv Then Rc ← 1 Else Rc ← 0
10.1D	CMPULT	Ra.rq,Rb.rq,Rc.wq	If Rav U< Rbv Then Rc ← 1 Else Rc ← 0
10.1D	CMPULT	Ra.rq,#b.ib,Rc.wq	If Rav U< Rbv Then Rc ← 1 Else Rc ← 0
17.020	CPYS	Fq.rq,Fb.rq,Fc.wq	Fc ← Fav<63> || Fbv<62:0>
17.022	CPYSE	Fq.rq,Fb.rq,Fc.wq	Fc ← Fav<63:52> || Fbv<51:0>
17.021	CPYSN	Fq.rq,Fb.rq,Fc.wq	Fc ← NOT Fav<63> || Fbv<62:0>
15.09E	CVTDG	Fb.rd,Fc.wg	Fc ← D_Float to G_Float of Fbv
15.0AD	CVTGD	Fb.rg,Fc.wd	Fc ← G_Float to D_Float of Fbv

Opcode	Mnemonic	Operands	Description
15.0AC	CVTGF	Fb.rg,Fc.wf	Fc ← G_Float to F_Float of Fbv
15.0AF	CVTGQ	Fb.rg,Fc.wq	Fc ← G_Float to quad of Fbv
17.010	CVTLQ	Fb.rq,Fc.wq	Fc ← Long to quad of Fbv
15.0BC	CVTQF	Fb.rq,Fc.wf	Fc ← Quad to F_Float of Fbv
15.0BE	CVTQG	Fb.rq,Fc.wg	Fc ← Quad to G_Float of Fbv
17.030	CVTQL	Fb.rq,Fc.wq	Fc ← Quad to long of Fbv
16.0BC	CVTQS	Fb.rq,Fc.ws	Fc ← Quad to S_Float of Fbv
16.0BE	CVTQT	Fb.rq,Fc.wt	Fc ← Quad to T_Float of Fbv
16.2AC	CVTST	Fb.rs,Fc.wT	Fc ← S_Float to T_Float of Fbv
16.0AF	CVTTQ	Fb.rt,Fc.wq	Fc ← T_Float to quad of Fbv
16.0AC	CVTTS	Fb.rt,Fc.ws	Fc ← T_Float to S_Float of Fbv
15.083	DIVF	Fa.rf,Fb.rf,Fc.wq	Fc ← Fav / Fbv
15.0A3	DIVG	Fa.rg,Fb.rg,Fc.wq	Fc ← Fav / Fbv
16.083	DIVS	Fa.rs,Fb.rs,Fc.wq	Fc ← Fav / Fbv
16.0A3	DIVT	Fa.rt,Fb.rt,Fc.wq	Fc ← Fav / Fbv
11.48	EQV	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav XOR NOT Rbv
12.06	EXTBL	Ra.rq,Rb.rq,Rc.wq	Rc ← Byte extract low from Rav at Rbv<5:0>
12.06	EXTBL	Ra.rq,#b.ib,Rc.wq	Rc ← Byte extract low from Rav at Rbv<5:0>
12.6A	EXTLH	Ra.rq,Rb.rq,Rc.wq	Rc ← Long extract high from Rav at Rbv<5:0>
12.6A	EXTLH	Ra.rq,#b.ib,Rc.wq	Rc ← Long extract high from Rav at Rbv<5:0>
12.26	EXTLL	Ra.rq,Rb.rq,Rc.wq	Rc ← Long extract low from Rav at Rbv<5:0>

Opcode	Mnemonic	Operands	Description
12.26	EXTLL	Ra.rq,#b.ib,Rc.wq	Rc ← Long extract low from Rav at Rbv<5:0>
12.7A	EXTQH	Ra.rq,Rb.rq,Rc.wq	Rc ← Quad extract high from Rav at Rbv<5:0>
12.7A	EXTQH	Ra.rq,#b.ib,Rc.wq	Rc ← Quad extract high from Rav at Rbv<5:0>
12.36	EXTQL	Ra.rq,Rb.rq,Rc.wq	Rc ← Quad extract low from Rav at Rbv<5:0>
12.36	EXTQL	Ra.rq,#b.ib,Rc.wq	Rc ← Quad extract low from Rav at Rbv<5:0>
12.5A	EXTWH	Ra.rq,Rb.rq,Rc.wq	Rc ← Word extract high from Rav at Rbv<5:0>
12.5A	EXTWH	Ra.rq,#b.ib,Rc.wq	Rc ← Word extract high from Rav at Rbv<5:0>
12.16	EXTWL	Ra.rq,Rb.rq,Rc.wq	Rc ← Word extract low from Rav at Rbv<5:0>
12.16	EXTWL	Ra.rq,#b.ib,Rc.wq	Rc ← Word extract low from Rav at Rbv<5:0>
31	FBEQ	Fa.rq,disp.al	If Fav = 0 Then PC ← PC + {4*SEXT(disp)}
36	FBGE	Fa.rq,disp.al	If Fav ≥ 0 Then PC ← PC + {4*SEXT(disp)}
37	FBGT	Fa.rq,disp.al	If Fav > 0 Then PC ← PC + {4*SEXT(disp)}
33	FBLE	Fa.rq,disp.al	If Fav ≤ 0 Then PC ← PC + {4*SEXT(disp)}
32	FBLT	Fa.rq,disp.al	If Fav < 0 Then PC ← PC + {4*SEXT(disp)}
35	FBNE	Fa.rq,disp.al	If Fav ≠ 0 Then PC ← PC + {4*SEXT(disp)}
17.02A	FCMOVEQ	Fa.rq,Fb.rq,Fc.wq	If Fav = 0 Then Fc ← Fbv
17.02D	FCMOVGE	Fa.rq,Fb.rq,Fc.wq	If Fav ≥ 0 Then Fc ← Fbv
17.02F	FCMOVGT	Fa.rq,Fb.rq,Fc.wq	If Fav > 0 Then Fc ← Fbv
17.02E	FCMOVLE	Fa.rq,Fb.rq,Fc.wq	If Fav ≤ 0 Then Fc ← Fbv
17.02C	FCMOVLT	Fa.rq,Fb.rq,Fc.wq	If Fav < 0 Then Fc ← Fbv
17.02B	FCMOVNE	Fa.rq,Fb.rq,Fc.wq	If Fav ≠ 0 Then Fc ← Fbv

Opcode	Mnemonic	Operands	Description
18.8000	FETCH	0(Rb.ab)	Prefetch around (Rbv)
18.A000	FETCH_M	0(Rb.ab)	Prefetch around (Rbv), modify intent
12.0B	INSBL	Ra.rq,Rb.rq,Rc.wq	Rc ← Byte insert low from Rav at Rbv<5:0>
12.0B	INSBL	Ra.rq,#b.ib,Rc.wq	Rc ← Byte insert low from Rav at Rbv<5:0>
12.67	INSLH	Ra.rq,Rb.rq,Rc.wq	Rc ← Long insert high from Rav at Rbv<5:0>
12.67	INSLH	Ra.rq,#b.ib,Rc.wq	Rc ← Long insert high from Rav at Rbv<5:0>
12.2B	INSLL	Ra.rq,Rb.rq,Rc.wq	Rc ← Long insert low from Rav at Rbv<5:0>
12.2B	INSLL	Ra.rq,#b.ib,Rc.wq	Rc ← Long insert low from Rav at Rbv<5:0>
12.77	INSQH	Ra.rq,Rb.rq,Rc.wq	Rc ← Quad insert high from Rav at Rbv<5:0>
12.77	INSQH	Ra.rq,#b.ib,Rc.wq	Rc ← Quad insert high from Rav at Rbv<5:0>
12.3B	INSQL	Ra.rq,Rb.rq,Rc.wq	Rc ← Quad insert low from Rav at Rbv<5:0>
12.3B	INSQL	Ra.rq,#b.ib,Rc.wq	Rc ← Quad insert low from Rav at Rbv<5:0>
12.57	INSWH	Ra.rq,Rb.rq,Rc.wq	Rc ← Word insert high from Rav at Rbv<5:0>
12.57	INSWH	Ra.rq,#b.ib,Rc.wq	Rc ← Word insert high from Rav at Rbv<5:0>
12.1B	INSWL	Ra.rq,Rb.rq,Rc.wq	Rc ← Word insert low from Rav at Rbv<5:0>
12.1B	INSWL	Ra.rq,#b.ib,Rc.wq	Rc ← Word insert low from Rav at Rbv<5:0>
1A.0	JMP	Ra.wq,(Rb.ab),hint	Ra ← PC; PC ← Rbv AND NOT 3
1A.1	JSR	Ra.wq,(Rb.ab),hint	Ra ← PC; PC ← Rbv AND NOT 3
1A.3	JSR_COROUTINE	Ra.wq,(Rb.ab),hint	Ra ← PC; PC ← Rbv AND NOT 3
08	LDA	Ra.wq,disp.ab(Rb.ab)	Ra ← Rbv + SEXT(disp)
09	LDAH	Ra.wq,disp.ab(Rb.ab)	Ra ← Rbv + SEXT(disp*65536)

Opcode	Mnemonic	Operands	Description
20	LDF	Fa.wf,disp.ab(Rb.ab)	Fa ← ({Rbv + SEXT(disp)})
21	LDG	Fa.wg,disp.ab(Rb.ab)	Fa ← ({Rbv + SEXT(disp)})
28	LDL	Ra.wq,disp.ab(Rb.ab)	Ra ← SEXT(({Rbv + SEXT(disp)})<31:0>)
2A	LDL_L	Ra.wq,disp.ab(Rb.ab)	Ra ← SEXT(({Rbv + SEXT(disp)})<31:0>)
29	LDQ	Ra.wq,disp.ab(Rb.ab)	Ra ← ({Rbv + SEXT(disp)})
2B	LDQ_L	Ra.wq,disp.ab(Rb.ab)	Ra ← ({Rbv + SEXT(disp)})
0B	LDQ_U	Ra.wq,disp.ab(Rb.ab)	Ra ← ({{Rbv + SEXT(disp)} AND NOT 7})
22	LDS	Fa.ws,disp.ab(Rb.ab)	Fa ← ({Rbv + SEXT(disp)})
23	LDT	Fa.wt,disp.ab(Rb.ab)	Fa ← ({Rbv + SEXT(disp)})
18.4000	MB		Memory barrier
17.025	MF_FPCR	Fa.rq,Fq.rq,Fa.wq	Fa ←FPCR
12.02	MSKBL	Ra.rq,Rb.rq,Rc.wq	Rc ← Byte mask low from Rav at Rbv<5:0>
12.02	MSKBL	Ra.rq,#b.ib,Rc.wq	Rc ← Byte mask low from Rav at Rbv<5:0>
12.62	MSKLH	Ra.rq,Rb.rq,Rc.wq	Rc ← Long mask high from Rav at Rbv<5:0>
12.62	MSKLH	Ra.rq,#b.ib,Rc.wq	Rc ← Long mask high from Rav at Rbv<5:0>
12.22	MSKLL	Ra.rq,Rb.rq,Rc.wq	Rc ← Long mask low from Rav at Rbv<5:0>
12.22	MSKLL	Ra.rq,#b.ib,Rc.wq	Rc ← Long mask low from Rav at Rbv<5:0>
12.72	MSKQH	Ra.rq,Rb.rq,Rc.wq	Rc ← Quad mask high from Rav at Rbv<5:0>
12.72	MSKQH	Ra.rq,#b.ib,Rc.wq	Rc ← Quad mask high from Rav at Rbv<5:0>
12.32	MSKQL	Ra.rq,Rb.rq,Rc.wq	Rc ← Quad mask low from Rav at Rbv<5:0>
12.32	MSKQL	Ra.rq,#b.ib,Rc.wq	Rc ← Quad mask low from Rav at Rbv<5:0>

Opcode	Mnemonic	Operands	Description
12.52	MSKWH	Ra.rq,Rb.rq,Rc.wq	Rc ← Word mask high from Rav at Rbv<5:0>
12.52	MSKWH	Ra.rq,#b.ib,Rc.wq	Rc ← Word mask high from Rav at Rbv<5:0>
12.12	MSKWL	Ra.rq,Rb.rq,Rc.wq	Rc ← Word mask low from Rav at Rbv<5:0>
12.12	MSKWL	Ra.rq,#b.ib,Rc.wq	Rc ← Word mask low from Rav at Rbv<5:0>
17.024	MT_FPCR	Fa.rq,Fq.rq,Fa.rq	FPCR ← Fa
15.082	MULF	Fa.rf,Fb.rf,Fc.wq	Fc ← Fav * Fbv
15.0A2	MULG	Fa.rg,Fb.rg,Fc.wq	Fc ← Fav * Fbv
13.00	MULL	Ra.rq,Rb.rq,Rc.wq	Rc ← SEXT({Rav * Rbv}<31:0>)
13.00	MULL	Ra.rq,#b.ib,Rc.wq	Rc ← SEXT({Rav * Rbv}<31:0>)
13.20	MULQ	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav * Rbv
13.20	MULQ	Ra.rq,#b.ib,Rc.wq	Rc ← Rav * Rbv
16.082	MULS	Fa.rs,Fb.rs,Fc.wq	Fc ← Fav * Fbv
16.0A2	MULT	Fa.rt,Fb.rt,Fc.wq	Fc ← Fav * Fbv
11.20	OR	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav OR Rbv
11.28	ORNOT	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav OR NOT Rbv
18.E000	RC	Ra.wq	Ra ← intr_flag; intr_flag ← 0
1A.2	RET	Ra.wq,(Rb.ab),hint	Ra ← PC; PC ← Rbv AND NOT 3
18.C000	RPCC	Ra.wq	Ra ← Process cycle counter
18.F000	RS	Ra.wq	Ra ← intr_flag; intr_flag ← 1
10.02	S4ADDL	Ra.rq,Rb.rq,Rc.wq	Rc ← SEXT({SLL(Rav,2) + Rbv}<31:0>)
10.02	S4ADDL	Ra.rq,#b.ib,Rc.wq	Rc ← SEXT({SLL(Rav,2) + Rbv}<31:0>)

Opcode	Mnemonic	Operands	Description
10.22	S4ADDQ	Ra.rq,Rb.rq,Rc.wq	Rc ← SLL(Rav,2) + Rbv
10.22	S4ADDQ	Ra.rq,#b.ib,Rc.wq	Rc ← SLL(Rav,2) + Rbv
10.0B	S4SUBL	Ra.rq,Rb.rq,Rc.wq	Rc ← SEXT({SLL(Rav,2) – Rbv}<31:0>)
10.0B	S4SUBL	Ra.rq,#b.ib,Rc.wq	Rc ← SEXT({SLL(Rav,2) – Rbv}<31:0>)
10.2B	S4SUBQ	Ra.rq,Rb.rq,Rc.wq	Rc ← SLL(Rav,2) – Rbv
10.2B	S4SUBQ	Ra.rq,#b.ib,Rc.wq	Rc ← SLL(Rav,2) – Rbv
10.12	S8ADDL	Ra.rq,Rb.rq,Rc.wq	Rc ← SEXT({SLL(Rav,3) + Rbv}<31:0>)
10.12	S8ADDL	Ra.rq,#b.ib,Rc.wq	Rc ← SEXT({SLL(Rav,3) + Rbv}<31:0>)
10.32	S8ADDQ	Ra.rq,Rb.rq,Rc.wq	Rc ← SLL(Rav,3) + Rbv
10.32	S8ADDQ	Ra.rq,#b.ib,Rc.wq	Rc ← SLL(Rav,3) + Rbv
10.1B	S8SUBL	Ra.rq,Rb.rq,Rc.wq	Rc ← SEXT({SLL(Rav,3) – Rbv}<31:0>)
10.1B	S8SUBL	Ra.rq,#b.ib,Rc.wq	Rc ← SEXT({SLL(Rav,3) – Rbv}<31:0>)
10.3B	S8SUBQ	Ra.rq,Rb.rq,Rc.wq	Rc ← SLL(Rav,3) – Rbv
10.3B	S8SUBQ	Ra.rq,#b.ib,Rc.wq	Rc ← SLL(Rav,3) – Rbv
12.39	SLL	Ra.rq,Rb.rq,Rc.wq	Rc ← SLL(Rav,Rvb<5:0>)
12.39	SLL	Ra.rq,#b.ib,Rc.wq	Rc ← SLL(Rav,Rvb<5:0>)
12.3C	SRA	Ra.rq,Rb.rq,Rc.wq	Rc ← SRA(Rav,Rvb<5:0>)
12.3C	SRA	Ra.rq,#b.ib,Rc.wq	Rc ← SRA(Rav,Rvb<5:0>)
12.34	SRL	Ra.rq,Rb.rq,Rc.wq	Rc ← SRL(Rav,Rvb<5:0>)
12.34	SRL	Ra.rq,#b.ib,Rc.wq	Rc ← SRL(Rav,Rvb<5:0>)
24	STF	Fa.rf,disp.ab(Rb.ab)	({Rbv + SEXT(disp)}) ← Fav

Opcode	Mnemonic	Operands	Description
25	STG	Fa.rg,disp.ab(Rb.ab)	({Rbv + SEXT(disp)}) ← Fav
2C	STL	Ra.rq,disp.ab(Rb.ab)	({Rbv + SEXT(disp)})<31:0> ← Rav<31:0>
2E	STL_C	Ra.rq,disp.ab(Rb.ab)	({Rbv + SEXT(disp)})<31:0> ← Rav<31:0>
2D	STQ	Ra.rq,disp.ab(Rb.ab)	({Rbv + SEXT(disp)}) ← Rav
2F	STQ_C	Ra.rq,disp.ab(Rb.ab)	({Rbv + SEXT(disp)}) ← Rav
0F	STQ_U	Ra.rq,disp.ab(Rb.ab)	({{Rbv + SEXT(disp)} AND NOT 7}) ← Rav
26	STS	Fa.rs,disp.ab(Rb.ab)	({Rbv + SEXT(disp)}) ← Fav
27	STT	Fa.rt,disp.ab(Rb.ab)	({Rbv + SEXT(disp)}) ← Fav
15.081	SUBF	Fa.rf,Fb.rf,Fc.wq	Fc ← Fav – Fbv
15.0A1	SUBG	Fa.rg,Fb.rg,Fc.wq	Fc ← Fav – Fbv
10.09	SUBL	Ra.rq,Rb.rq,Rc.wq	Rc ← SEXT({Rav – Rbv}<31:0>)
10.09	SUBL	Ra.rq,#b.ib,Rc.wq	Rc ← SEXT({Rav – Rbv}<31:0>)
10.29	SUBQ	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav – Rbv
10.29	SUBQ	Ra.rq,#b.ib,Rc.wq	Rc ← Rav – Rbv
16.081	SUBS	Fa.rs,Fb.rs,Fc.wq	Fc ← Fav – Fbv
16.0A1	SUBT	Fa.rt,Fb.rt,Fc.wq	Fc ← Fav – Fbv
18.0000	TRAPB		Drain any pending traps
13.30	UMULH	Ra.rq,Rb.rq,Rc.wq	Rc ← {Rav U* Rbv}<127:64>
13.30	UMULH	Ra.rq,#b.ib,Rc.wq	Rc ← {Rav U* Rbv}<127:64>
18.4400	WMB		Write memory barrier
11.40	XOR	Ra.rq,Rb.rq,Rc.wq	Rc ← Rav XOR Rbv

Opcode	Mnemonic	Operands	Description
12.30	ZAP	Ra.rq,Rb.rq,Rc.wq	Rc ← Zap from Rav, byte mask Rbv<7:0>
12.30	ZAP	Ra.rq,#b.ib,Rc.wq	Rc ← Zap from Rav, byte mask Rbv<7:0>
12.31	ZAPNOT	Ra.rq,Rb.rq,Rc.wq	Rc ← Zap from Rav, byte mask NOT Rbv<7:0>
12.31	ZAPNOT	Ra.rq,#b.ib,Rc.wq	Rc ← Zap from Rav, byte mask NOT Rbv<7:0>

PALcode instructions

PALcode	Mnemonic	Priv	OSF	VMS	Description
00A1	AMOVRM			√	Atomic move register/memory
00A0	AMOVRR			√	Atomic move register/register
0080	BPT		√	√	Breakpoint exception
0081	BUGCHK		√	√	Bugcheck exception
0083	CALLSYS		√		System call
0001	CFLUSH	√		√	Flush page from cache
0082	CHME			√	Change mode to executive
0083	CHMK			√	Change mode to Kernel
0084	CHMS			√	Change mode to supervisor
0085	CHMU			√	Change mode to user
0002	DRAINA	√	√	√	Drain any pending aborts
00AA	GENTRAP		√	√	Generate software trap
0000	HALT	√	√	√	Halt processor
0086	IMB		√	√	I-stream memory barrier
008B	INSQUEL			√	Insert into longword queue
008C	INSQUEQ			√	Insert into quadword queue
0087	INSQHIL			√	Insert into longword queue at head, interlocked
0089	INSQHIQ			√	Insert into quadword queue at head, interlocked
0088	INSQTIL			√	Insert into longword queue at tail, interlocked

PALcode	Mnemonic	Priv	OSF	VMS	Description
008A	INSQTIQ			√	Insert into quadword queue at tail, interlocked
0003	LDQP	√		√	Load quadword physical
	MFPR	√		√	Move from processor register
	MTPR	√		√	Move to processor register
008F	PROBER			√	Probe read access
0090	PROBEW			√	Probe write access
0091	RD_PS			√	Read processor status
0036	RDPS	√	√		Read processor status
003A	RDUSP	√	√		Read user stack pointer
0032	RDVAL	√	√		Read system value
009E	READ_UNQ		√	√	Read process unique context
0092	REI	√		√	Return from exception or interrupt
0097	REMQUEL			√	Remove from longword queue
0098	REMQUEQ			√	Remove from quadword queue
0093	REMQHIL			√	Remove from longword queue at head, interlocked
0095	REMQHIQ			√	Remove from quadword queue at head, interlocked
0094	REMQTIL			√	Remove from longword queue at tail, interlocked
0096	REMQTIQ			√	Remove from quadword queue at tail, interlocked
003D	RETSYS	√	√		Return from system call
009D	RSCC			√	Read system cycle counter
003F	RTI	√	√		Return from trap, fault, or interrupt

PALcode	Mnemonic	Priv	OSF	VMS	Description
0004	STQP	√		√	Store quadword physical
009B	SWASTEN			√	Swap AST enable
0005	SWPCTX	√		√	Swap privileged context
0030	SWPCTX	√	√		Swap process context
0035	SWPIPL	√	√		Swap IPL
0033	TBI	√	√		Translation buffer invalidate
003C	WHAMI	√	√		Who am I
009C	WR_PS_SW			√	Write processor status software field
0034	WRENT	√	√		Write system entry address
002B	WRFEN	√	√		Write floating point enable
009F	WRITE_UNQ		√	√	Write process unique context
0037	WRKGP	√	√		Write kernel global pointer
0038	WRUSP	√	√		Write user stack pointer
0031	WRVAL	√	√		Write system value
002D	WRVPTPTR	√	√		Write virtual page table pointer

Processor registers (MFPR/MTPR)

For processor registers with R/W access, the PALcode shown is the one for read access (MFPR). The code for write access (MTPR) is one higher.

PALcode	Mnemonic	Access	Description
0006	ASN	R	Address space number
0007	ASTEN	R/W*	AST enable
0008	ASTSR	R/W*	AST summary register
002E	DATFX	W	Data alignment trap fixup
001E	ESP	R/W	Executive stack pointer
000B	FEN	R/W	Floating point enable
000D	IPIR	W	Interprocessor interrupt request
000E	IPL	R/W*	Interrupt priority level
0010	MCES	R/W	Machine check error summary
0012	PCBB	R	Privileged context block base
002B	PERFMON	W*	Performance monitor
0013	PRBR	R/W	Processor base register
0015	PTBR	R	Page table base register
0016	SCBB	R/W	System control block base
0018	SIRR	W	Software interrupt request register
0019	SISR	R	Software interrupt summary register

PALcode	Mnemonic	Access	Description
0020	SSP	R/W	Supervisor stack pointer
001A	TBCHK	R	Translation buffer check
001B	TBIA	W	Translation buffer invalidate all
001C	TBIAP	W	Translation buffer invalidate all process
001D	TBIS	W	Translation buffer invalidate single
0024	TBISD	W	Translation buffer invalidate single data
0025	TBISI	W	Translation buffer invalidate single instruction
0022	USP	R/W	User stack pointer
0029	VPTB	R/W	Virtual page table base
003F	WHAMI	R	Who am I

Instruction set (numeric order)

00	CALL_PAL	00.001E	MFPR_ESP	00.0080	BPT	00.009B	SWASTEN
00.0000	HALT	00.0020	MFPR_SSP	00.0081	BUGCHK	00.009C	WR_PS_SW
00.0001	CFLUSH	00.0022	MFPR_USP	00.0082	CHME	00.009D	RSCC
00.0002	DRAINA	00.0024	MTPR_TBISD	00.0083	CALLSYS	00.009E	READ_UNQ
00.0003	LDQP	00.0025	MTPR_TBISI	00.0083	CHMK	00.009F	WRITE_UNQ
00.0004	STQP	00.0029	MFPR_VPTB	00.0084	CHMS	00.00A0	AMOVRR
00.0005	SWPCTX	00.002B	MTPR_PERFMON	00.0085	CHMU	00.00A1	AMOVRM
00.0006	MFPR_ASN	00.002B	WRFEN	00.0086	IMB	00.00AA	GENTRAP
00.0007	MFPR_ASTEN	00.002D	WRVPTPTR	00.0087	INSQHIL	08	LDA
00.0008	MFPR_ASTSR	00.002E	MTPR_DATFX	00.0088	INSQTIL	09	LDAH
00.000B	MFPR_FEN	00.0030	SWPCTX	00.0089	INSQHIQ	0B	LDQ_U
00.000D	MTPR_IPIR	00.0031	WRVAL	00.008A	INSQTIQ	0F	STQ_U
00.000E	MFPR_IPL	00.0032	RDVAL	00.008B	INSQUEL	10.00	ADDL
00.0010	MFPR_MCES	00.0033	TBI	00.008C	INSQUEQ	10.00	ADDL
00.0012	MFPR_PCBB	00.0034	WRENT	00.008F	PROBER	10.02	S4ADDL
00.0013	MFPR_PRBR	00.0035	SWPIPL	00.0090	PROBEW	10.02	S4ADDL
00.0015	MFPR_PTBR	00.0036	RDPS	00.0091	RD_PS	10.09	SUBL
00.0016	MFPR_SCBB	00.0037	WRKGP	00.0092	REI	10.09	SUBL
00.0018	MTPR_SIRR	00.0038	WRUSP	00.0093	REMQHIL	10.0B	S4SUBL
00.0019	MFPR_SISR	00.003A	RDUSP	00.0094	REMQTIL	10.0B	S4SUBL
00.001A	MFPR_TBCHK	00.003C	WHAMI	00.0095	REMQHIQ	10.0F	CMPBGE
00.001B	MTPR_TBIA	00.003D	RETSYS	00.0096	REMQTIQ	10.0F	CMPBGE
00.001C	MTPR_TBIAP	00.003F	MFPR_WHAMI	00.0097	REMQUEL	10.12	S8ADDL
00.001D	MTPR_TBIS	00.003F	RTI	00.0098	REMQUEQ	10.12	S8ADDL

10.1B	S8SUBL	11.14	CMOVLBC	12.12	MSKWL	12.3C	SRA
10.1B	S8SUBL	11.14	CMOVLBC	12.12	MSKWL	12.3C	SRA
10.1D	CMPULT	11.16	CMOVLBS	12.16	EXTWL	12.52	MSKWH
10.1D	CMPULT	11.16	CMOVLBS	12.16	EXTWL	12.52	MSKWH
10.20	ADDQ	11.20	OR	12.1B	INSWL	12.57	INSWH
10.20	ADDQ	11.24	CMOVEQ	12.1B	INSWL	12.57	INSWH
10.22	S4ADDQ	11.24	CMOVEQ	12.22	MSKLL	12.5A	EXTWH
10.22	S4ADDQ	11.26	CMOVNE	12.22	MSKLL	12.5A	EXTWH
10.29	SUBQ	11.26	CMOVNE	12.26	EXTLL	12.62	MSKLH
10.29	SUBQ	11.28	ORNOT	12.26	EXTLL	12.62	MSKLH
10.2B	S4SUBQ	11.40	XOR	12.2B	INSLL	12.67	INSLH
10.2B	S4SUBQ	11.44	CMOVLT	12.2B	INSLL	12.67	INSLH
10.2D	CMPEQ	11.44	CMOVLT	12.30	ZAP	12.6A	EXTLH
10.2D	CMPEQ	11.46	CMOVGE	12.30	ZAP	12.6A	EXTLH
10.32	S8ADDQ	11.46	CMOVGE	12.31	ZAPNOT	12.72	MSKQH
10.32	S8ADDQ	11.48	EQV	12.31	ZAPNOT	12.72	MSKQH
10.3B	S8SUBQ	11.64	CMOVLE	12.32	MSKQL	12.77	INSQH
10.3B	S8SUBQ	11.64	CMOVLE	12.32	MSKQL	12.77	INSQH
10.3D	CMPULE	11.66	CMOVGT	12.34	SRL	12.7A	EXTQH
10.3D	CMPULE	11.66	CMOVGT	12.34	SRL	12.7A	EXTQH
10.4D	CMPLT	12.02	MSKBL	12.36	EXTQL	13.00	MULL
10.4D	CMPLT	12.02	MSKBL	12.36	EXTQL	13.00	MULL
10.6D	CMPLE	12.06	EXTBL	12.39	SLL	13.20	MULQ
10.6D	CMPLE	12.06	EXTBL	12.39	SLL	13.20	MULQ
11.00	AND	12.0B	INSBL	12.3B	INSQL	13.30	UMULH
11.08	BIC	12.0B	INSBL	12.3B	INSQL	13.30	UMULH

15.080	ADDF	16.0A5	CMPTEQ	18.C000	RPCC	33	FBLE
15.081	SUBF	16.0A6	CMPTLT	18.E000	RC	34	BSR
15.082	MULF	16.0A7	CMPTLE	18.F000	RS	35	FBNE
15.083	DIVF	16.0AC	CVTTS	1A.0	JMP	36	FBGE
15.09E	CVTDG	16.0AF	CVTTQ	1A.1	JSR	37	FBGT
15.0A0	ADDG	16.0BC	CVTQS	1A.2	RET	38	BLBC
15.0A1	SUBG	16.0BE	CVTQT	1A.3	JSR_COROUTINE	39	BEQ
15.0A2	MULG	16.2AC	CVTST	20	LDF	3A	BLT
15.0A3	DIVG	17.010	CVTLQ	21	LDG	3B	BLE
15.0A5	CMPGEQ	17.020	CPYS	22	LDS	3C	BLBS
15.0A6	CMPGLT	17.021	CPYSN	23	LDT	3D	BNE
15.0A7	CMPGLE	17.022	CPYSE	24	STF	3E	BGE
15.0AC	CVTGF	17.024	MT_FPCR	25	STG	3F	BGT
15.0AD	CVTGD	17.025	MF_FPCR	26	STS		
15.0AF	CVTGQ	17.02A	FCMOVEQ	27	STT		
15.0BC	CVTQF	17.02B	FCMOVNE	28	LDL		
15.0BE	CVTQG	17.02C	FCMOVLT	29	LDQ		
16.080	ADDS	17.02D	FCMOVGE	2A	LDL_L		
16.081	SUBS	17.02E	FCMOVLE	2B	LDQ_L		
16.082	MULS	17.02F	FCMOVGT	2C	STL		
16.083	DIVS	17.030	CVTQL	2D	STQ		
16.0A0	ADDT	18.0000	TRAPB	2E	STL_C		
16.0A1	SUBT	18.4000	MB	2F	STQ_C		
16.0A2	MULT	18.4400	WMB	30	BR		
16.0A3	DIVT	18.8000	FETCH	31	FBEQ		
16.0A4	CMPTUN	18.A000	FETCH_M	32	FBLT		

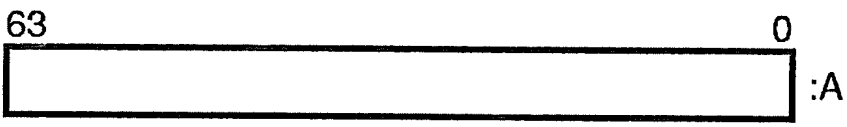
Data types

Byte:

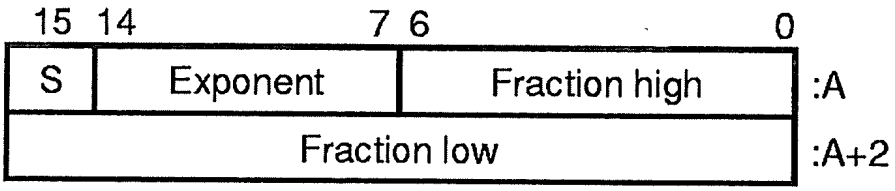
Word:

Longword:

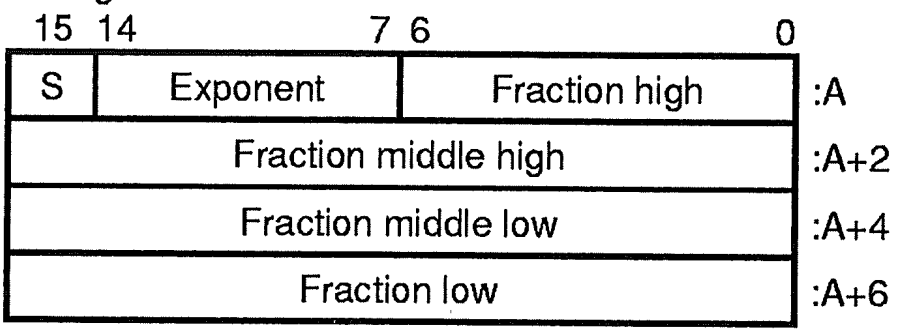
Quadword:



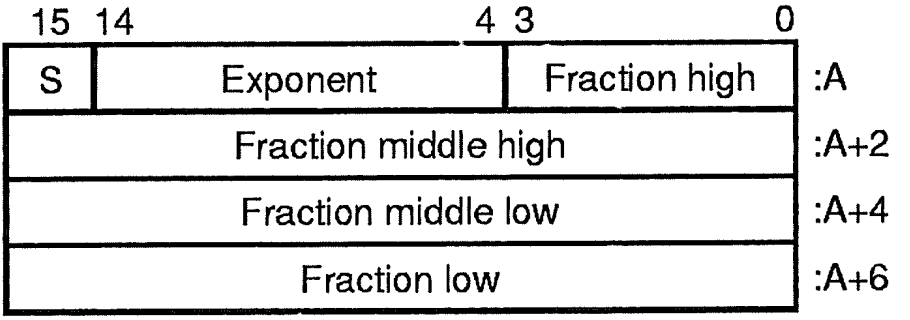
F_Floating:



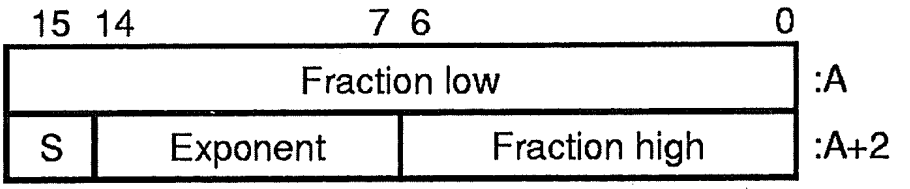
D_Floating:



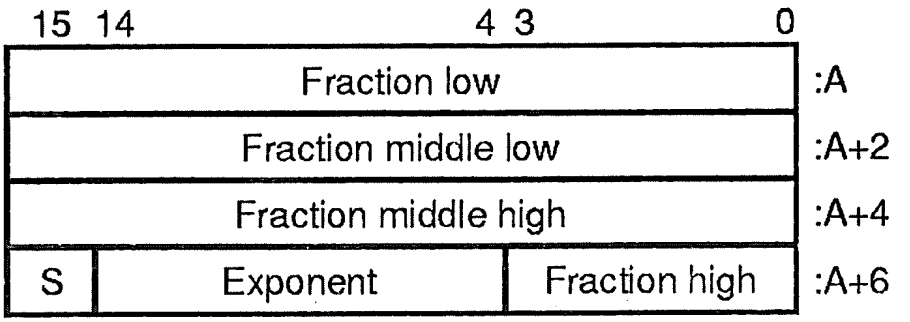
G_Floating:



S_Floating:



T_Floating:



Operand notation

Notation used in instruction operands:

<name>.<access type><data type>

1. *<name>* is the instruction field and register type:
   - disp displacement field
   - fnc PAL function field
   - Ra integer register operand in Ra field
   - Rb integer register operand in Rb field
   - #b integer literal operand in Rb field
   - Rc integer register operand in Rc field
   - Fa floating point register operand in Ra field
   - Fb floating point register operand in Rb field
   - Fc floating point register operand in Rc field
2. *<access type>* is the operand access type:
   - a operand is an effective address
   - i immediate literal operand
   - r read only operand
   - w write only operand
3. *<data type* is the operand data type:
   - b byte
   - f F_Floating
   - g G_Floating
   - l longword
   - q quadword
   - s IEEE single floating (S_Floating)
   - t IEEE double floating (T_Floating)
   - w word
   - x data type specified by instruction

Notation used in instruction descriptions:

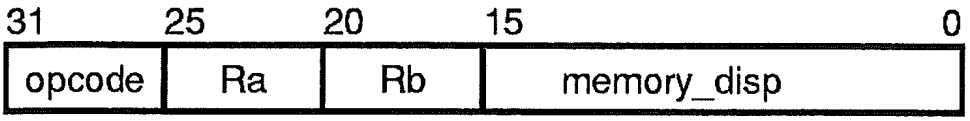
Rav	Value of integer source operand Ra
Rbv	Value of integer source operand Rb
Fav	Value of floating point source operand Fa
Fbv	Value of floating point source operand Fb
←	Assignment
||	Concatenation
{ }	Explicit precedence
(*x*)	Contents of memory at address *x*
x<*m*:*n*>	Bits *m* to *n* of *x*
x<*m*>	Bit *m* of *x*
SLL(*x*,*y*)	*x* logical left shifted by *y*
SRL(*x*,*y*)	*x* logical right shifted by *y*
SRA(*x*,*y*)	*x* arithmetic right shifted by *y*
SEXT(*x*)	*x* sign extended to the required size

Opcode qualifiers:

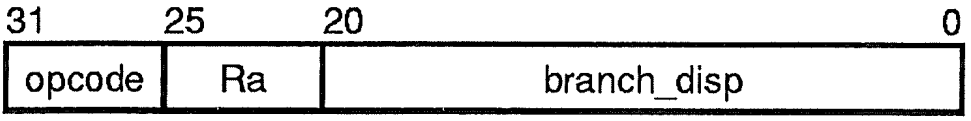
/C	chopped rounding
/M	round towards $-\infty$
/D	dynamic control of rounding
/I	inexact results enable
/S	software completion enable
/U	floating underflow enable
/V	integer overflow enable

Instruction formats

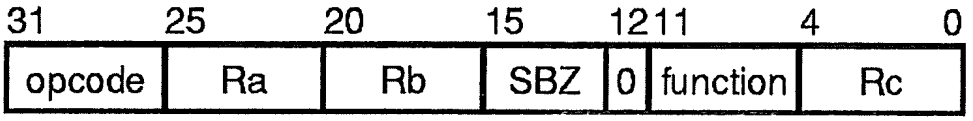
Memory instruction:

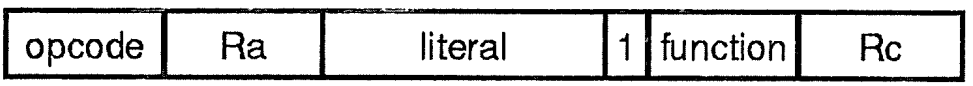


Branch instruction:

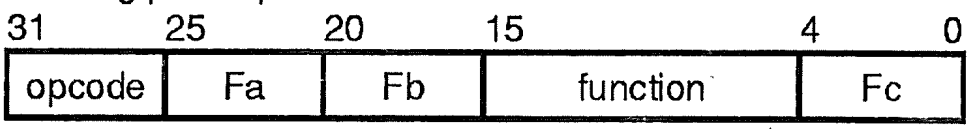


Operate instruction:

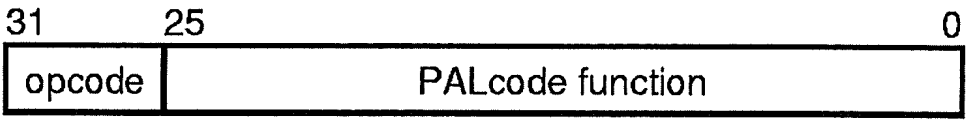




Floating point operate instruction:



PALcode instruction:



Register usage conventions

Conventional register usage for VMS:

R0		integer function return value
R1		scratch register
R2–R15		saved registers
R16–R21		argument registers
R22–R24		scratch registers
R25	AI	Argument Information register
R26	RA	Return Address register
R27	PV	Procedure Value register
R28		volatile scratch register
R29	FP	stack frame base register
R30	SP	Stack Pointer
R31		zero

F0	floating point function return value
F1	floating point complex function return value
R2–F9	saved registers
F10–F15	scratch register
F16–F22	argument registers
F23–F30	scratch registers
F31	zero

Conventional register usage for OSF:

R0	v0	integer function return value
R1–R8	t0–t7	scratch registers
R9–R14	s0–s5	saved registers
R15	s6/fp	saved/frame pointer register
R16–R21	a0–a5	argument registers
R22–R25	t8–t11	scratch registers
R26	ra	Return Address register
R27	t12/pv	scratch/Procedure Value register
R28	at	volatile scratch register
R29	fp	Global Pointer
R30	sp	Stack Pointer
R31		zero

Floating point registers same as above

Hexadecimal to decimal conversion

	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0
1	268,435,456	16,777,216	1,048,576	65,536	4,096	256	16	1
2	536,870,912	33,554,432	2,097,152	131,072	8,192	512	32	2
3	805,306,368	50,331,648	3,145,728	196,608	12,288	768	48	3
4	1,073,741,824	67,108,864	4,194,304	262,144	16,384	1,024	64	4
5	1,342,177,280	83,886,080	5,242,880	327,680	20,480	1,280	80	5
6	1,610,612,736	100,663,296	6,291,456	393,216	24,576	1,536	96	6
7	1,879,048,192	117,440,512	7,340,032	458,752	28,672	1,792	112	7
8	2,147,483,648	134,217,728	8,388,608	524,288	32,768	2,048	128	8
9	2,415,919,104	150,994,944	9,437,184	589,824	36,864	2,304	144	9
A	2,684,354,560	167,772,160	10,485,760	655,360	40,960	2,560	160	10
B	2,952,790,016	184,549,376	11,534,336	720,896	45,056	2,816	176	11
C	3,221,225,472	201,326,592	12,582,912	786,432	49,152	3,072	192	12
D	3,489,660,928	218,103,808	13,631,488	851,968	53,248	3,328	208	13
E	3,758,096,384	234,881,024	14,680,064	917,504	57,344	3,584	224	14
F	4,026,531,840	251,658,240	15,728,640	983,040	61,440	3,840	240	15

Character code table

	2x	3x	4x	5x	6x	7x	Ax	Bx	Cx	Dx	Ex	Fx
x0		0	@	P	`	p		°	À	Ð	à	ð
x1	!	1	A	Q	a	q	¡	±	Á	Ñ	á	ñ
x2	"	2	B	R	b	r	¢	²	Â	Ò	â	ò
x3	#	3	C	S	c	s	£	³	Ã	Ó	ã	ó
x4	$	4	D	T	d	t	¤	´	Ä	Ô	ä	ô
x5	%	5	E	U	e	u	¥	µ	Å	Õ	å	õ
x6	&	6	F	V	f	v	¦	¶	Æ	Ö	æ	ö
x7	'	7	G	W	g	w	§	·	Ç	×	ç	÷
x8	(	8	H	X	h	x	¨	¸	È	Ø	è	ø
x9	)	9	I	Y	i	y	©	¹	É	Ù	é	ù
xA	*	:	J	Z	j	z	ª	º	Ê	Ú	ê	ú
xB	+	;	K	[	k	{	«	»	Ë	Û	ë	û
xC	,	<	L	\	l	|	¬	¼	Ì	Ü	ì	ü
xD	-	=	M	]	m	}		½	Í	Ý	í	ý
xE	.	>	N	^	n	~	®	¾	Î	Þ	î	þ
xF	/	?	O	_	o		¯	¿	Ï	ß	ï	ÿ