

# 10 Alpha Instruction Summary

This section contains a summary of all Alpha architecture instructions. All values are in hexadecimal radix. Table 15 describes the contents of the Format and Opcode columns that are in Table 16.

**table 15 Instruction Format and Opcode Notation**

Instruction Format	Format Symbol	Opcode Notation	Meaning
Branch	Bra	<i>oo</i>	<i>oo</i> is the 6-bit opcode field.
Floating-point	F-P	<i>oo,fff</i>	<i>oo</i> is the 6-bit opcode field. <i>fff</i> is the 11-bit function code field.
Memory	Mem	<i>oo</i>	<i>oo</i> is the 6-bit opcode field.
Memory/ function code	Mfc	<i>oo,ffff</i>	<i>oo</i> is the 6-bit opcode field. <i>ffff</i> is the 16-bit function code in the displacement field.
Memory/ branch	Mbr	<i>oo.h</i>	<i>oo</i> is the 6-bit opcode field. <i>h</i> is the high-order 2 bits of the displacement field.
Operate	Opr	<i>oo,ff</i>	<i>oo</i> is the 6-bit opcode field. <i>ff</i> is the 7-bit function code field.
PALcode	Pcd	<i>oo</i>	<i>oo</i> is the 6-bit opcode field; the particular PALcode instruction is specified in the 26-bit function code field.

Qualifiers for operate instructions are shown in Table 16. Qualifiers for IEEE and VAX floating-point instructions are shown in Tables 19 and 20, respectively.

**table 16 Architecture Instructions**

*(Sheet 1 of 7)*

<b>Mnemonic</b>	<b>Format</b>	<b>Opcode</b>	<b>Description</b>
ADDF	F-P	15.080	Add F_floating
ADDG	F-P	15.0A0	Add G_floating
ADDL	Opr	10.00	Add longword
ADDL/V	Opr	10.40	Add longword
ADDQ	Opr	10.20	Add quadword
ADDQ/V	Opr	10.60	Add quadword
ADDS	F-P	16.080	Add S_floating
ADDT	F-P	16.0A0	Add T_floating
AMASK	Opr	11.61	Determine byte/word instruction implementation
AND	Opr	11.00	Logical product
BEQ	Bra	39	Branch if = zero
BGE	Bra	3E	Branch if $\geq$ zero
BGT	Bra	3F	Branch if $>$ zero
BIC	Opr	11.0	Bit clear
BIS	Opr	11.20	Logical sum
BLBC	Bra	38	Branch if low bit clear
BLBS	Bra	3C	Branch if low bit set
BLE	Bra	3B	Branch if $\leq$ zero
BLT	Bra	3A	Branch if $<$ zero
BNE	Bra	3D	Branch if $\neq$ zero
BR	Bra	30	Unconditional branch
BSR	Mbr	34	Branch to subroutine
CALL_PAL	Pcd	00	Trap to PALcode
CMOVEQ	Opr	11.24	CMOVE if = zero

**table 16 Architecture Instructions**

(Sheet 2 of 7)

<b>Mnemonic</b>	<b>Format</b>	<b>Opcode</b>	<b>Description</b>
CMOVGE	Opr	11.46	CMOVE if $\geq$ zero
CMOVGT	Opr	11.66	CMOVE if $>$ zero
CMOVLBC	Opr	11.16	CMOVE if low bit clear
CMOVLBS	Opr	11.14	CMOVE if low bit set
CMOVLE	Opr	11.64	CMOVE if $\leq$ zero
CMOVLT	Opr	11.44	CMOVE if $<$ zero
CMOVNE	Opr	11.26	CMOVE if $\neq$ zero
CMPBGE	Opr	10.0F	Compare byte
CMPEQ	Opr	10.2D	Compare signed quadword equal
CMPGEQ	F-P	15.0A5	Compare G_floating equal
CMPGLE	F-P	15.0A7	Compare G_floating less than or equal
CMPGLT	F-P	15.0A6	Compare G_floating less than
CMPLE	Opr	10.6D	Compare signed quadword less than or equal
CMPLT	Opr	10.4D	Compare signed quadword less than
CMPTEQ	F-P	16.0A5	Compare T_floating equal
CMPTLE	F-P	16.0A7	Compare T_floating less than or equal
CMPTLT	F-P	16.0A6	Compare T_floating less than
CMPTUN	F-P	16.0A4	Compare T_floating unordered
CMPULE	Opr	10.3D	Compare unsigned quadword less than or equal
CMPULT	Opr	10.1D	Compare unsigned quadword less than
CPYS	F-P	17.020	Copy sign
CPYSE	F-P	17.022	Copy sign and exponent
CPYSN	F-P	17.021	Copy sign negate
CVTDG	F-P	15.09E	Convert D_floating to G_floating
CVTGD	F-P	15.0AD	Convert G_floating to D_floating
CVTGF	F-P	15.0AC	Convert G_floating to F_floating

**table 16 Architecture Instructions***(Sheet 3 of 7)*

<b>Mnemonic</b>	<b>Format</b>	<b>Opcode</b>	<b>Description</b>
CVTGQ	F-P	15.0AF	Convert G_floating to quadword
CVTLQ	F-P	17.010	Convert longword to quadword
CVTQF	F-P	15.0BC	Convert quadword to F_floating
CVTQG	F-P	15.0BE	Convert quadword to G_floating
CVTQL	F-P	17.030	Convert quadword to longword
CVTQL/SV	F-P	17.530	Convert quadword to longword
CVTQL/V	F-P	17.130	Convert quadword to longword
CVTQS	F-P	16.0BC	Convert quadword to S_floating
CVTQT	F-P	16.0BE	Convert quadword to T_floating
CVTST	F-P	16.2AC	Convert S_floating to T_floating
CVTTQ	F-P	16.0AF	Convert T_floating to quadword
CVTTS	F-P	16.0AC	Convert T_floating to S_floating
DIVF	F-P	15.083	Divide F_floating
DIVG	F-P	15.0A3	Divide G_floating
DIVS	F-P	16.083	Divide S_floating
DIVT	F-P	16.0A3	Divide T_floating
EQV	Opr	11.48	Logical equivalence
EXCB	Mfc	18.0400	Exception barrier
EXTBL	Opr	12.06	Extract byte low
EXTLH	Opr	12.6A	Extract longword high
EXTLL	Opr	12.26	Extract longword low
EXTQH	Opr	12.7A	Extract quadword high
EXTQL	Opr	12.36	Extract quadword low
EXTWH	Opr	12.5A	Extract word high
EXTWL	Opr	12.16	Extract word low
FBEQ	Bra	31	Floating branch if = zero

**table 16 Architecture Instructions**

(Sheet 4 of 7)

<b>Mnemonic</b>	<b>Format</b>	<b>Opcode</b>	<b>Description</b>
FBGE	Bra	36	Floating branch if $\geq$ zero
FBGT	Bra	37	Floating branch if $>$ zero
FBLE	Bra	33	Floating branch if $\leq$ zero
FBLT	Bra	32	Floating branch if $<$ zero
FBNE	Bra	35	Floating branch if $\neq$ zero
FCMOVEQ	F-P	17.02A	FCMOVE if = zero
FCMOVGE	F-P	17.02D	FCMOVE if $\geq$ zero
FCMOVGT	F-P	17.02F	FCMOVE if $>$ zero
FCMOVLE	F-P	17.02E	FCMOVE if $\leq$ zero
FCMOVLT	F-P	17.02C	FCMOVE if $<$ zero
FCMOVNE	F-P	17.02B	FCMOVE if $\neq$ zero
FETCH	Mfc	18.80	Prefetch data
FETCH_M	Mfc	18.A0	Prefetch data, modify intent
IMPLVER	Opr	11.6C	Determine CPU type
INSBL	Opr	12.0B	Insert byte low
INSLH	Opr	12.67	Insert longword high
INSLL	Opr	12.2B	Insert longword low
INSQH	Opr	12.77	Insert quadword high
INSQL	Opr	12.3B	Insert quadword low
INSWH	Opr	12.57	Insert word high
INSWL	Opr	12.1B	Insert word low
JMP	Mbr	1A.0	Jump
JSR	Mbr	1A.1	Jump to subroutine
JSR_COROUTINE	Mbr	1A.3	Jump to subroutine return
LDA	Mem	08	Load address
LDAH	Mem	09	Load address high

**table 16 Architecture Instructions***(Sheet 5 of 7)*

<b>Mnemonic</b>	<b>Format</b>	<b>Opcode</b>	<b>Description</b>
LDBU	Mem	0A	Load zero-extended byte
LDF	Mem	20	Load F_floating
LDG	Mem	21	Load G_floating
LDL	Mem	28	Load sign-extended longword
LDL_L	Mem	2A	Load sign-extended longword locked
LDQ	Mem	29	Load quadword
LDQ_L	Mem	2B	Load quadword locked
LDQ_U	Mem	0B	Load unaligned quadword
LDS	Mem	22	Load S_floating
LDT	Mem	23	Load T_floating
LDWU	Mem	0C	Load zero-extended word
MB	Mfc	18.4000	Memory barrier
MF_FPCR	F-P	17.025	Move from floating-point control register
MSKBL	Opr	12.02	Mask byte low
MSKHL	Opr	12.62	Mask longword high
MSKLL	Opr	12.22	Mask longword low
MSKQH	Opr	12.72	Mask quadword high
MSKQL	Opr	12.32	Mask quadword low
MSKWH	Opr	12.52	Mask word high
MSKWL	Opr	12.12	Mask word low
MT_FPCR	F-P	17.024	Move to floating-point control register
MULF	F-P	15.082	Multiply F_floating
MULG	F-P	15.0A2	Multiply G_floating
MULL	Opr	13.00	Multiply longword
MULL/V	Opr	13.40	Multiply longword
MULQ	Opr	13.20	Multiply quadword

**table 16 Architecture Instructions***(Sheet 6 of 7)*

<b>Mnemonic</b>	<b>Format</b>	<b>Opcode</b>	<b>Description</b>
MULQ/V	Opr	13.60	Multiply quadword
MULS	F-P	16.082	Multiply S_floating
MULT	F-P	16.0A2	Multiply T_floating
ORNOT	Opr	11.28	Logical sum with complement
RC	Mfc	18.E0	Read and clear
RET	Mbr	1A.2	Return from subroutine
RPCC	Mfc	18.C0	Read process cycle counter
RS	Mfc	18.F000	Read and set
S4ADDL	Opr	10.02	Scaled add longword by 4
S4ADDQ	Opr	10.22	Scaled add quadword by 4
S4SUBL	Opr	10.0B	Scaled subtract longword by 4
S4SUBQ	Opr	10.2B	Scaled subtract quadword by 4
S8ADDL	Opr	10.12	Scaled add longword by 8
S8ADDQ	Opr	10.32	Scaled add quadword by 8
S8SUBL	Opr	10.1B	Scaled subtract longword by 8
S8SUBQ	Opr	10.3B	Scaled subtract quadword by 8
SEXTB	Opr	1C.00	Store byte
SEXTW	Opr	1C.01	Store word
SLL	Opr	12.39	Shift left logical
SRA	Opr	12.3C	Shift right arithmetic
SRL	Opr	12.34	Shift right logical
STB	Mem	0E	Store byte
STF	Mem	24	Store F_floating
STG	Mem	25	Store G_floating
STS	Mem	26	Store S_floating
STL	Mem	2C	Store longword

## Reserved Opcodes

**table 16 Architecture Instructions**

(Sheet 7 of 7)

Mnemonic	Format	Opcode	Description
STL_C	Mem	2E	Store longword conditional
STQ	Mem	2D	Store quadword
STQ_C	Mem	2F	Store quadword conditional
STQ_U	Mem	0F	Store unaligned quadword
STT	Mem	27	Store T_floating
STW	Mem	0D	Store word
SUBF	F-P	15.081	Subtract F_floating
SUBG	F-P	15.0A1	Subtract G_floating
SUBL	Opr	10.09	Subtract longword
SUBL/V		10.49	
SUBQ	Opr	10.29	Subtract quadword
SUBQ/V		10.69	
SUBS	F-P	16.081	Subtract S_floating
SUBT	F-P	16.0A1	Subtract T_floating
TRAPB	Mfc	18.00	Trap barrier
UMULH	Opr	13.30	Unsigned multiply quadword high
WMB	Mfc	18.44	Write memory barrier
XOR	Opr	11.40	Logical difference
ZAP	Opr	12.30	Zero bytes
ZAPNOT	Opr	12.31	Zero bytes not

### 10.1 Reserved Opcodes

This section describes the opcodes that are reserved in the Alpha architecture.

## Reserved Opcodes Opcodes Reserved

### 10.1.1 Opcodes Reserved

Table 17 lists opcodes reserved.

**table 17 Opcodes Reserved**

Mnemonic	Opcode	Mnemonic	Opcode	Mnemonic	Opcode
OPC01	01	OPC05	05	OPC0B	0B
OPC02	02	OPC06	06	OPC0C	0C <sup>1</sup>
OPC03	03	OPC07	07	OPC0D	0D <sup>1</sup>
OPC04	04	OPC0A	0A <sup>1</sup>	OPC0E	0E <sup>1</sup>

<sup>1</sup> Reserved when byte/word instructions are not enabled.

### 10.1.2 Opcodes Reserved for PALcode

Table 18 lists the 21164-specific instructions. For more information, refer to the *Samsung 21164 Alpha Microprocessor Hardware Reference Manual*.

**table 18 Opcodes Reserved for PALcode**

21164		Architecture	
Mnemonic	Opcode	Mnemonic	Function
HW_LD	1B	PAL1B	Performs Dstream load instructions.
HW_ST	1F	PAL1F	Performs Dstream store instructions.
HW_REI	1E	PAL1E	Returns instruction flow to the program counter (PC) pointed to by EXC_ADDR internal processor register (IPR).
HW_MFPR	19	PAL19	Accesses the IDU, MTU, and Dcache IPRs.
HW_MTPR	1D	PAL1D	Accesses the IDU, MTU, and Dcache IPRs.

## IEEE Floating-Point Instructions

Opcodes Reserved

### 10.2 IEEE Floating-Point Instructions

Table 19 lists the hexadecimal value of the 11-bit function code field for the IEEE floating-point instructions, with and without qualifiers. The opcode for these instructions is  $16_{16}$ .

**table 19 IEEE Floating-Point Instruction Function Codes** *(Sheet 1 of 2)*

Mnemonic	None	/C	/M	/D	/U	/UC	/UM	/UD
ADDS	080	000	040	0C0	180	100	140	1C0
ADDT	0A0	020	060	0E0	1A0	120	160	1E0
CMPTEQ	0A5							
CMPTLT	0A6							
CMPTLE	0A7							
CMPTUN	0A4							
CVTQS	0BC	03C	07C	0FC				
CVTQT	0BE	03E	07E	0FE				
CVTTS	0AC	02C	06C	0EC	1AC	12C	16C	1EC
DIVS	083	003	043	0C3	183	103	143	1C3
DIVT	0A3	023	063	0E3	1A3	123	163	1E3
MULS	082	002	042	0C2	182	102	142	1C2
MULT	0A2	022	062	0E2	1A2	122	162	1E2
SUBS	081	001	041	0C1	181	101	141	1C1
SUBT	0A1	021	061	0E1	1A1	121	161	1E1

  

Mnemonic	/SU	/SUC	/SUM	/SUD	/SUI	/SUIC	/SUIM	/SUID
ADDS	580	500	540	5C0	780	700	740	7C0
ADDT	5A0	520	560	5E0	7A0	720	760	7E0
CMPTEQ	5A5							
CMPTLT	5A6							
CMPTLE	5A7							
CMPTUN	5A4							

## IEEE Floating-Point Instructions Opcodes Reserved

**table 19 IEEE Floating-Point Instruction Function Codes** *(Sheet 2 of 2)*

Mnemonic	/SU	/SUC	/SUM	/SUD	/SUI	/SUIC	/SUIM	/SUID
CVTQS					7BC	73C	77C	7FC
CVTQT					7BE	73E	77E	7F3
CVTTS	5AC	52C	56C	5EC	7AC	72C	76C	7EC
DIVS	583	503	543	5C3	783	703	743	7C3
DIVT	5A3	523	563	5E3	7A3	723	763	7E3
MULS	582	502	542	5C2	782	702	742	7C2
MULT	5A2	522	562	5E2	7A2	722	762	7E2
SUBS	581	501	541	5C1	781	701	741	7C1
SUBT	5A1	521	561	5E1	7A1	721	761	7E1

  

Mnemonic	None	/S
CVTST	2AC	6AC

  

Mnemonic	None	/C	/V	/VC	/SV	/SVC	/SVI	/SVIC
CVTTQ	0AF	02F	1AF	12F	5AF	52F	7AF	72F

  

Mnemonic	D	/VD	/SVD	/SVID	/M	/VM	/SVM	/SVIM
CVTTQ	0EF	1EF	5EF	7EF	06F	16F	56F	76F

**Note:** Because underflow cannot occur for CMPT<sub>xx</sub>, there is no difference in function or performance between CMPT<sub>xx</sub>/S and CMPT<sub>xx</sub>/SU. It is intended that software generate CMPT<sub>xx</sub>/SU in place of CMPT<sub>xx</sub>/S.

In the same manner, CVTQS and CVTQT can take an inexact result trap, but not an underflow. Because there is no encoding for a CVTQ<sub>x</sub>/SI instruction, it is intended that software generate CVTQ<sub>x</sub>/SUI in place of CVTQ<sub>x</sub>/SI.

## VAX Floating-Point Instructions

Opcodes Reserved

### 10.3 VAX Floating-Point Instructions

Table 20 lists the hexadecimal value of the 11-bit function code field for the VAX floating-point instructions. The opcode for these instructions is  $15_{16}$ .

**table 20 VAX Floating-Point Instruction Function Codes**

Mnemonic	None	/C	/U	/UC	/S	/SC	/SU	/SUC
ADDF	080	000	180	100	480	400	580	500
CVTDG	09E	01E	19E	11E	49E	41E	59E	51E
ADDG	0A0	020	1A0	120	4A0	420	5A0	520
CMPGEQ	0A5				4A5			
CMPGLT	0A6				4A6			
CMPGLE	0A7				4A7			
CVTGF	0AC	02C	1AC	12C	4AC	42C	5AC	52C
CVTGD	0AD	02D	1AD	12D	4AD	42D	5AD	52D
CVTQF	0BC	03C						
CVTQG	0BE	03E						
DIVF	083	003	183	103	483	403	583	503
DIVG	0A3	023	1A3	123	4A3	423	5A3	523
MULF	082	002	182	102	482	402	582	502
MULG	0A2	022	1A2	122	4A2	422	5A2	522
SUBF	081	001	181	101	481	401	581	501
SUBG	0A1	021	1A1	121	4A1	421	5A1	521
Mnemonic	None	/C	/V	/VC	/S	/SC	/SV	/SVC
CVTGQ	0AF	02F	1AF	12F	4AF	42F	5AF	52F

## 10.4 Opcode Summary

Table 21 lists all Alpha opcodes from 00 (CALL\_PAL) through 3F (BGT). In the table, the column headings that appear over the instructions have a granularity of  $8_{16}$ . The rows beneath the Offset column supply the individual hexadecimal number to resolve that granularity.

If an instruction column has a 0 in the right (low) hexadecimal digit, replace that 0 with the number to the left of the backslash (\) in the Offset column on the instruction's row. If an instruction column has an 8 in the right (low) hexadecimal digit, replace that 8 with the number to the right of the backslash in the Offset column.

For example, the third row (2/A) under the  $10_{16}$  column contains the symbol INTS\*, representing the all-integer shift instructions. The opcode for those instructions would then be  $12_{16}$  because the 0 in 10 is replaced by the 2 in the Offset column. Likewise, the third row under the  $18_{16}$  column contains the symbol JSR\*, representing all jump instructions. The opcode for those instructions is 1A because the 8 in the heading is replaced by the number to the right of the backslash in the Offset column.

## Opcode Summary

### Opcodes Reserved

The instruction format is listed under the instruction symbol.

**table 21 Opcode Summary**

Offset	00	08	10	18	20	28	30	38
0/8	PAL* (pal)	LDA (mem)	INTA* (op)	MISC* (mem)	LDF (mem)	LDL (mem)	BR (br)	BLBC (br)
1/9	Res	LDAH (mem)	INTL* (op)	\PAL\ (mem)	LDG (mem)	LDQ (mem)	FBEQ (br)	BEQ (br)
2/A	Res	LDBU (mem)	INTS* (op)	JSR* (mem)	LDS (mem)	LDL_L (mem)	FBLT (br)	BLT (br)
3/B	Res	LDQ_U (mem)	INTM* (op)	\PAL\ (mem)	LDT (mem)	LDQ_L (mem)	FBLE (br)	BLE (br)
4/C	Res	LDWU (mem)	Res	SEXT* (op)	STF (mem)	STL (mem)	BSR (br)	BLBS (br)
5/D	Res	STW (mem)	FLTV* (op)	\PAL\ (mem)	STG (mem)	STQ (mem)	FBNE (br)	BNE (br)
6/E	Res	STB (mem)	FLTI* (op)	\PAL\ (mem)	STS (mem)	STL_C (mem)	FBGE (br)	BGE (br)
7/F	Res	STQ_U (mem)	FLTL* (op)	\PAL\ (mem)	STT (mem)	STQ_C (mem)	FBGT (br)	BGT (br)
<b>Symbol</b>	<b>Meaning</b>							
FLTI*	IEEE floating-point instruction opcodes							
FLTL*	Floating-point operate instruction opcodes							
FLTV*	VAX floating-point instruction opcodes							
INTA*	Integer arithmetic instruction opcodes							
INTL*	Integer logical instruction opcodes							
INTM*	Integer multiply instruction opcodes							
INTS*	Integer shift instruction opcodes							
JSR*	Jump instruction opcodes							
MISC*	Miscellaneous instruction opcodes							
PAL*	PALcode instruction (CALL_PAL) opcodes							
\PAL\ Res	Reserved for PALcode Reserved							
SEXT*	Sign extend opcodes							

## Required PALcode Function Codes Opcodes Reserved

### 10.5 Required PALcode Function Codes

The opcodes listed in Table 22 are required for all Alpha implementations. The notation used is *oo.fff*, where *oo* is the hexadecimal 6-bit opcode and *fff* is the hexadecimal 26-bit function code.

**table 22 Required PALcode Function Codes**

<b>Mnemonic</b>	<b>Type</b>	<b>Function Code</b>
DRAINA	Privileged	00.0002
HALT	Privileged	00.0000
IMB	Unprivileged	00.0086