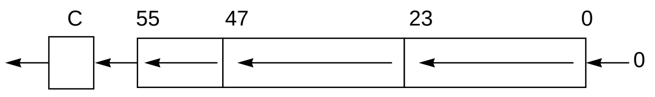
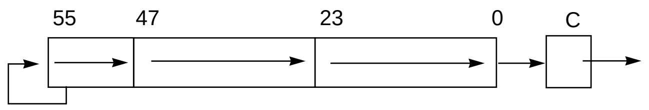
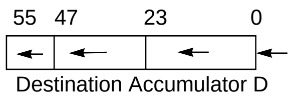
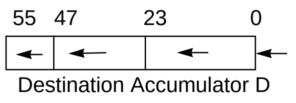
DSP56300 FAMILY INSTRUCTION SET

Mnemonic	Operation Examples	Assembler Syntax Examples
See the DSP56300 Family Manual for complete details on each instruction.		
ABS	**Absolute Value** |D| → D	ABS D [Parallel Move]
ADC	**Add Long With Carry** S + D + C → D	ADC S,D [Parallel Move]
ADD	**Addition** S + D → D	ADD S,D [Parallel Move]
ADDL	**Shift Left and Add Accumulators** S + 2xD → D	ADDL S,D [Parallel Move]
ADDR	**Arithmetic Shift Right and Add Accumulators** S + D/2 → D	ADDR S,D [Parallel Move]
AND	**Logical AND** S • D [47:24] → D [47:24]	AND S,D [Parallel Move]
ANDI	**AND Immediate With Control Register** #xx • D → D	AND(I) #xx,D
ASL	**Arithmetic Shift Accumulator Left** 	ASL D [Parallel Move]
ASR	**Arithmetic Shift Accumulator Right** 	ASR D [Parallel Move]
BCHG	**Bit Test and Change** **D[n] → C $\overline{\text{D[n]}}$ → D[n]	BCHG #n,D
BCLR	**Bit Test and Clear** **D[n] → C 0 → D[n]	BCLR #n,D
Bcc	**Branch Conditionally** If cc, then PC + xxxx → PC else PC + 1 → PC If cc, then PC + xxx → PC else PC + 1 → PC If cc, then PC + Rn → PC else PC + 1 → PC	Bcc xxxx Bcc xxx Bcc Rn
BRA	**Branch Always** PC + xxxx → PC PC + xxx → PC PC + Rn → PC	BRA xxxx BRA xxx BRA Rn
BRCLR	**Branch if bit Clear** If S{n} = 0, then PC + xxxx → PC else PC + 1 → PC	BRCLR #n,[X or Y]:ea,xxxx
BRKcc	**Exit Current DO Loop Conditionally** If cc LA + 1 → PC; SSL(LF,FV) → SR; SP - 1 → SP SSH → LA; SSL → LC; SP - 1 → SP else PC + 1 → PC	BRKcc
BRSET	**Branch if bit Set** If S{n} = 1, then PC + xxxx → PC else PC + 1 → PC	BRSET #n,[X or Y]:ea,xxxx
BScc	**Branch to Subroutine Conditionally** If cc, then PC → SSH; SR → SSL; PC + xxxx → PC else PC + 1 → PC If cc, then PC → SSH; SR → SSL; PC + xxx → PC else PC + 1 → PC If cc, then PC → SSH; SR → SSL; PC + Rn → PC else PC + 1 → PC	BScc xxxx BScc xxx BScc Rn
BSCLR	**Branch to Subroutine if bit Clear** If S{n} = 0, then PC → SSH; SR → SSL; PC + xxxx → PC else PC + 1 → PC	BSCLR #n,[X or Y]:ea,xxxx

Mnemonic	Operation Examples	Assembler Syntax Examples
See the DSP56300 Family Manual for complete details on each instruction.		
BSET	**Bit Set and Test** **D[n] → C 1 → D[n]	BSET #n,D
BSR	**Branch to Subroutine** PC → SSH; SR → SSL; PC + xxxx → PC PC → SSH; SR → SSL; PC + xxx → PC PC → SSH; SR → SSL; PC + Rn → PC	BSR xxx BSR xxx BSR Rn
BSSET	**Branch to Subroutine if bit Set** If S{n} = 1, then PC → SSH; SR → SSL; PC + xxxx → PC else PC + 1 → PC	BSSET #n,[X or Y]:ea,xxxx
BTST	**Bit Test** **D[n] → C	BTST #n,D
CLB	**Count Leading Bits** If S[55]=0 then 9 - (Number of consecutive leading zeros in S[55:0]) → D[47:24] else 9 - (Number of consecutive leading ones in S[55:0]) → D[47:24]	CLB S,D
CLR	**Clear Accumulator** 0 → D	CLR D [Parallel Move]
CMP	**Compare** S2 - S1	CMP S1,S2 [Parallel Move]
CMPM	**Compare Magnitude** |S2| - |S1|	CMPM S1,S2 [Parallel Movc]
CMPU	**Compare Unsigned** S2 - S1	CMPU S1,S2
DEBUG	Enter Debug Mode & wait for OnCe commands	DEBUG
DEBUGcc	If cc enter dubug mode & wait for OnCe commands	DEBUGcc
DEC	**Decrement by One** D - 1 → D	DEC D
DIV	**Divide Iteration** If D[55] ⊕ S[23] = 1 then  C + S ← D else  C - S ← D	DIV S,D
DMAC	**Double-Precision Multiply-Accumulate With Right Shift** [D>>24] ± S1 • S2 → D (S1 signed, S2 signed) [D>>24] ± S1 • S2 → D (S1 signed, S2 unsigned) [D>>24] ± S1 • S2 → D (S1 unsigned, S2 unsigned)	DMACss (±)S1,S2,D [No Parallel Move] DMACsu (±)S2,S1,D [No Parallel Move] DMACuu (±)S2,S1,D [No Parallel Move]
DO	**Start Hardware loop** **Start of Loop** SP + 1 → SP; LA → SSH; LC → SSL; #xxx → LC SP + 1 → SP; PC → SSH; SR → SSL; Expr - 1 → LA 1 → LF **End of Loop** If LC ≠ 1 then LC - 1 → LC; SSH → PC Else SSL(LF) → SR; SP - 1 → SP SSH → LA; SSL → LC; SP - 1 → SP	**Static** DO #xxx,Expr **Dynamic** DO X: ea,Expr DO Y: ea,Expr DO S,Expr
DO FOREVER	**Start Infinite Loop** SP + 1 → SP; LA → SSH; LC → SSL SP + 1 → SP; PC → SSH; SR → SSL; expr - 1 → LA 1 → LF; 1 → FV	DO FOREVER,expr
DOR	**Start PC Relative Hardware Loop** SP + 1 → SP; LA → SSH; LC → SSL; [X or Y]:ea → LC SP + 1 → SP; PC → SSH; SR → SSL; PC+xxxx → LA 1 → LF	DOR [X or Y]:ea,label

Mnemonic	Operation Examples	Assembler Syntax Examples
See the DSP56300 Family Manual for complete details on each instruction.		
DOR FOREVER	**Start PC Relative Infinite Loop** SP + 1 → SP; LA → SSH; LC → SSL SP + 1 → SP; PC → SSH; SR → SSL; PC+xxxx → LA 1 → LF; 1 → FV	DOR FOREVER,label
ENDDO	**End Current DO Loop** SSL(LF) → SR; SP - 1 → SP SSH → LA; SSL → LC, SP - 1 → SP	ENDDO
EOR	**Logical Exclusive OR** S ⊕ D [47:24] → D [47:24]	EOR S,D [Parallel Move]

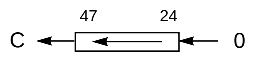
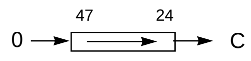
EXTRACT	**Extract Bit Field**
	Offset = S1[5:0]
	Width = S1[17:12]
	S2[(offset+width-1):offset] → D[(width-1):0]
	S2[(offset+width-1)] → D[(55:width)] (sign extention)
	
	Offset = #CO[5:0]
	Width = #CO[17:12]
	S2[(offset+width-1):offset] → D[(width-1):0]
	S2[(offset+width-1)] → D[(55:width)] (sign extention)
		EXTRACT S1,S2,D
		
		EXTRACT #CO,S2,D

EXTRACTU	**Extract Unsigned Bit Field**
	Offset = S1[5:0]
	Width = S1[17:12]
	S2[(offset+width-1):offset] → D[(width-1):0]
	zero → D[(55:width)]
	
	Offset = #CO[5:0]
	Width = #CO[17:12]
	S2[(offset+width-1):offset] → D[(width-1):0]
	zero → D[(55:width)]
		EXTRACTU S1,S2,D
		
		EXTRACTU #CO,S2,D

IFcc	**Execute Conditionally Without CCR Update** If cc, then opcode operation	Opcode-Operands IFcc
IFcc.U	**Execute Conditionally With CCR Update** If cc, then opcode operation	Opcode-Operands IFcc
INC	**Increment by One** 1 + D → D	INC D

INSERT	**Insert Bit Field**
	Offset = S1[5:0]
	Width = S1[17:12]
	S2[(width - 1):0] → D[(offset+width-1):offset]
	
	Offset = #CO[5:0]
	Width = #CO[17:12]
	S2[(width - 1):0] → D[(offset+width-1):offset]
		INSERT S1,S2,D
		
		INSERT #CO,S2,D

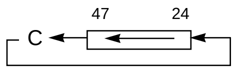
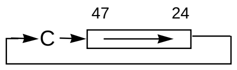
Jcc	**Jump Conditionally** If cc then ea → PC Else PC+ → PC	Jcc ea
JCLR	**Jump if Bit Clear** If S[n] = 0 Then xxxx → PC Else PC + 1 → PC	JCLR #n,D,xxxxxx
JMP	**Jump** ea → PC	JMP ea
JScc	**Conditional Jump to Subroutine** If cc then SP + 1 → SP PC → SSH SR → SSL ea → PC Else PC + 1 → PC	JScc ea

Mnemonic	Operation Examples	Assembler Syntax Examples
See the DSP56300 Family Manual for complete details on each instruction.		
JSCLR	**Jump to Subroutine if Bit Clear** If S[n] = 0 Then SP + 1 $\rightarrow$ SP PC $\rightarrow$ SSH SR $\rightarrow$ SSL xxxx $\rightarrow$ PC Else PC + 1 $\rightarrow$ PC	JSCLR #n,D,xxxxxx
JSET	**Jump if Bit Set** If S[n] = 1 Then xxxx $\rightarrow$ PC Else PC + 1 $\rightarrow$ PC	JSET #n,D,xxxxxx
JSR	**Jump to Subroutine** SP + 1 $\rightarrow$ SP PC $\rightarrow$ SSH SR $\rightarrow$ SSL ea $\rightarrow$ PC	JSR ea
JSSET	**Jump to Subroutine if Bit Set** If S[n] = 1 Then SP + 1 $\rightarrow$ SP PC $\rightarrow$ SSH SR $\rightarrow$ SSL xxxx $\rightarrow$ PC Else PC + 1 $\rightarrow$ PC	JSSET #n,D,xxxxxx
L:	**Long Memory Data Move** (. . . . .); X:ea $\rightarrow$ D1; Y:ea $\rightarrow$ D2 (. . . . .); X:aa $\rightarrow$ D1; Y:aa $\rightarrow$ D2 (. . . . .); S1 $\rightarrow$ X:ea; S2 $\rightarrow$ Y:ea (. . . . .); S1 $\rightarrow$ X:aa; S2 $\rightarrow$ Y:aa	(. . . . .) L:ea,D (. . . . .) L:aa,D (. . . . .) S,L:ea (. . . . .) S,L:aa
LRA	**Load PC-Relative Address** PC + Rn $\rightarrow$ D PC + xxxx $\rightarrow$ D	LRA Rn,D LRA xxxx,D
LSL	**Logical Shift Accumulator Left** 	LSL D [Parallel Move]
LSR	**Logical Shift Right** 	LSR D [Parallel Move]
LUA	**Load Updated Address** ea $\rightarrow$ D	LUA ea, D
MAC	**Signed Multiply-Accumulate** D ± (S1 • S2) $\rightarrow$ D	MAC ± S1,S2,D [Parallel Move]
MACI	**Signed Multiply-Accumulate With Immediate Operand** D ± #xxxxxx •S $\rightarrow$ D	MACI (±)#xxxxxx,S,D
MAC (su, uu)	**Mixed Multiply-Accumulate** D ± S1 • S2 $\rightarrow$ D (S1 unsigned, S2 unsigned) D ± S1 • S2 $\rightarrow$ D (S1 signed, S2 unsigned)	MACuu (±) S1, S2,D [No Parallel Move] MACsu (±) S2, S1,D [No Parallel Move]
MACR	**Signed Multiply-Accumulate and Round** D ± (S1 • S2) + r $\rightarrow$ D	MACR (±) S1, S2,D [Parallel Move]
MACRI	**Signed MAC and Round With Immediate Operand** D ± #xxxxxx • S $\rightarrow$ D	MACRI (±)#xxxxxx,S,D
MAX	**Transfer by Signed Value** If B - A ≤ 0 then A $\rightarrow$ B	MAX A,B [Parallel Move]
MAXM	**Transfer by Magnitude** If |B| - |A| ≤ 0 then A $\rightarrow$ B	MAXM A,B [Parallel Move]
MERGE	**Merge Two Half Words** {S[11:0],D[35:24]} $\rightarrow$ D[47:24]	MERGE S,D
MOVE	**Move Data Registers** S $\rightarrow$ D	MOVE S,D
MOVEC	**Move Control Register** S $\rightarrow$ D	MOVEC S,D
MOVEM	**Move Program Memory** S $\rightarrow$ P:ea P:ea $\rightarrow$ D	MOVE(M) S,P:ea MOVE(M) P:ea,D

Mnemonic	Operation Examples	Assembler Syntax Examples
See the DSP56300 Family Manual for complete details on each instruction.		
MOVEP	**Move Peripheral Data** S → X or Y Peripheral X OR Y Peripheral → D	MOVEP S,X:<pp>|<qq> MOVEP S,Y:<pp>|<qq> MOVEP X:<pp>|<qq>,D MOVEP Y:<pp>|<qq>,D
Note: There are extensive variations in the MOVE instructions. Please see the DSP56300 UM available from the Reference section of this CD-ROM for a complete listing.		
MPY	**Signed Multiply** ± (S1 • S2) → D	MPY (±) S1,S2,D [Parallel Move]
MPY (su, uu)	**Mixed Multiply** ±S1 • S2 → D (S1 unsigned, S2 unsigned) ±S1 • S2 → D (S1 signed, S2 unsigned)	 MPYuu (±)S1,S2,D MPYsu (±)S2,S1,D
MPYI	**Signed Multiply With Immediate Operand** ±#xxxxxx•S → D	 MPYI (±)#xxxxxx,S,D
MPYR	**Signed Multiply and Round** ± (S1 * S2) + r→ D	MPYR (±) S1,S2,D [Parallel Move]
MPYRI	**Signed Multiply and Round With Immediate Operand** ±#xxxxxx • S + r → D	 MPYRI (±)#xxxxxx,S,D
NEG	**Negate Accumulator** 0 - D → D	NEG D [Parallel Move]
NOP	**No Operation** PC + 1 → PC	NOP

NORM	**Norm Accumulator Iteration**
	If $\overline{E} \bullet U \bullet \overline{Z} = 1$, then ASL D and Rn - 1 → Rn
	else if E = 1, then ASR D and Rn + 1 → R
	else NOP
		NORM Rn,D

NORMF	**Fast Accumulator Normalization** If S[23]=0 then ASR S,D else ASL -S,D	NORMF S,D
NOT	**Logical Complement** $\overline{\text{D [47:24]}}$ → D [47:24]	NOT D [Parallel Move]
OR	**Logical Inclusive OR** S + D [47:24] → D [47:24]	OR S,D [Parallel Move]
ORI	**OR Immediate With Control Register** #xx + D → D	OR(I) #xx,D
PFLUSH	**Program Cache Flush**	PFLUSH
PFLUSHUN	**Program Cache Flush Unlocked Sectors**	PFLUSHUN
PFREE	**Program Cache Global Unlock**	PFREE
PLOCK	**Lock Instruction Cache Sector**	PLOCK ea
PLOCKR	**Lock Instruction Cache Relative Sector**	PLOCKR xxxx
PUNLOCK	**Unlock Instruction Cache Sector**	PUNLOCK ea
PUNLOCKR	**Unlock Instruction Cache Relative Sector**	PUNLOCKR xxxx
R	**Register-to-Register Data Move** (. . . . .); S → D	 (. . . . .) S,D
R:Y	**Register and Y Data Move** **Class I** (. . . . .); S1 → D1; Y:ea → D2 (. . . . .); S1 → D1; S2 → Y:ea (. . . . .); S1 → D1; #xxxxxx → D2 **Class II** (. . . . .); Y0 → A; A → Y:ea (. . . . .); Y0 → B; B → Y:ea	 (. . . . .) S1,D1 Y:ea,D2 (. . . . .) S1,D1 S2,Y:ea (. . . . .) S1,D1 #xxxxxx,D2 (. . . . .) Y0,A A,Y:ea (. . . . .) Y0,B B,Y:ea
REP	**Repeat Next Instruction** LC → TEMP; #xxx → LC Repeat Next Instruction Until LC = 1 TEMP → LC	REP #xxx REP X:ea REP Y:ea REP S
RESET	**Reset On-Chip Peripheral Devices** 1 → I1 1 → I0 0 → IPR-C & IPR-P PC + 1 → PC	RESET

Mnemonic	Operation Examples	Assembler Syntax Examples
See the DSP56300 Family Manual for complete details on each instruction.		
RND	**Round Accumulator** D + r → D	RND D [Parallel Move]
ROL	**Rotate Accumulator Left** 	ROL D [Parallel Move]
ROR	**Rotate Right** 	ROR D [Parallel Move]
RTI	**Return From Interrupt** SSH → PC SSL → SR SP - 1 → SP	RTI
RTS	**Return From Subroutine** SSH → PC SP - 1 → SP	RTS
SBC	**Subtract Long With Carry** D - S - C → D	SBC S,D [Parallel Move]
STOP	**Stop Instruction Processing** (Enter the Stop Processing State and Stop Clock to on-chip ckts)	STOP
SUB	**Subtraction** D - S → D	SUB S,D [Parallel Move]
SUBL	**Shift Left and Subtract Accumulators** 2xD - S → D	SUBL S,D [Parallel Move]
SUBR	**Shift Right and Subtract Accumulators** D/2 - S → D	SUBR S,D [Parallel Move]
SWI	**Software Interrupt** Begin Software Exception Process	SWI
Tcc	**Transfer Condiitionally Data ALU Registers** If cc Then S1 → D1 or If cc Then S1 → D1 and S2 → D2	Tcc S1,D1 [S2,D2]
TFR	**Transfer Data ALU Register** S → D	TFR S,D [Parallel Move]
TRAP	**Software Interrupt** Begin trap execution process	TRAP
TRAPcc	**Conditional Software Interrupt** If cc, then Begin software exception processing	TRAPcc
TST	**Test Accumulator** S - 0	TST S [Parallel Move]
U	**Address Register Update** (. . . . .); ea → Rn	(. . . . .) ea
VSL	**Viterbi Shift Left**	
WAIT	**Wait for Interrupt or DMA Request** (Disable Clocks to Processor Core and Enter the Wait Processing State)	WAIT
X:	**X Memory Data Move** (. . . . .); X:ea → D (. . . . .); X:aa → D (. . . . .); S → X:ea (. . . . .); S → X:aa X:(Rn + xxx) → D X:(Rn + xxxx) → D D → X:(Rn + xxx) D → X:(Rn + xxxx)	(. . . . .); X:ea,D (. . . . .); X:aa,D (. . . . .); S,X:ea (. . . . .); S,X:aa MOVE X:(Rn + xxx),D MOVE X:(Rn + xxxx),D MOVE D,X:(Rn + xxx) MOVE D,X:(Rn + xxxx)
X:R	**X Memory and Register Data Move** **Class I** (. . . . .); X:ea → D1; S2 → D2 (. . . . .); S1 → X:ea; S2 → D2 (. . . . .); #xxxxxx → D1; S2 → D2 **Class II** (. . . . .); A → X:ea; X0 → A (. . . . .); B → X:ea; X0 → B	 (. . . . .) X:ea,D1 S2,D2 (. . . . .) S1,X:ea S2,D2 (. . . . .) #xxxxxx,D1; S2,D2 (. . . . .) A,X:ea X0,A (. . . . .) B,X:ea X0,B

Mnemonic	Operation Examples	Assembler Syntax Examples
See the DSP56300 Family Manual for complete details on each instruction.		
X: Y:	**XY Memory Data Move** (. . . . .); X:<eax> → D1; Y:<eay> → D2 (. . . . .); X:<eax> → D1; S2 → Y:<eay> (. . . . .); S1 → X:<eax>; Y:<eay> → D2 (. . . . .); S1 → X:<eax>; S2 → Y:<eay>	 (. . . . .) X:<eax>,D1; Y:<eay>,D2 (. . . . .) X:<eax>,D1; S2,Y:<eay> (. . . . .) S1,X:<eax>; Y:<eay>,D2 (. . . . .) S1,X:<eax>; S2,Y:<eay>
Y:	**Y Memory Data Move** (. . . . .); Y:ea → D (. . . . .); Y:aa → D (. . . . .); S → Y:ea (. . . . .); S → Y:aa Y:(Rn + xxx) → D Y:(Rn + xxxx) → D D → Y:(Rn + xxx) D → Y:(Rn + xxxx)	 (. . . . .) Y:ea,D (. . . . .) Y:aa,D (. . . . .) S,Y:ea (. . . . .) S,Y:aa MOVE Y:(Rn + xxx),D MOVE Y:(Rn + xxxx),D MOVE D,Y:(Rn + xxx) MOVE D,Y:(Rn + xxxx)