Instruction Set

The ADSP-21xx assembly language uses an algebraic syntax for ease of coding and readability. The sources and destinations of computations and data movements are written explicitly in each assembly statement, eliminating cryptic assembler mnemonics.

Every instruction assembles into a single 24-bit word and executes in a single cycle. The instructions encompass a wide variety of instruction types along with a high degree of operational parallelism. There are five basic categories of instructions: data move instructions, computational instructions, multifunction instructions, program flow control instructions and miscellaneous instructions. Multifunction instructions perform one or two data moves and a computation.

The instruction set is summarized below. The *ADSP-2100 Family Users Manual* contains a complete reference to the instruction set.

ALU Instructions

[IF cond]	AR|AF	=	xop + yop [+ C] ;	*Add/Add with Carry*
		=	xop – yop [+ C– 1] ;	*Subtract X – Y/Subtract X – Y with Borrow*
		=	yop – xop [+ C– 1] ;	*Subtract Y – X/Subtract Y – X with Borrow*
		=	xop AND yop ;	*AND*
		=	xop OR yop ;	*OR*
		=	xop XOR yop ;	*XOR*
		=	PASS xop ;	*Pass, Clear*
		=	– xop ;	*Negate*
		=	NOT xop ;	*NOT*
		=	ABS xop ;	*Absolute Value*
		=	yop + 1 ;	*Increment*
		=	yop – 1 ;	*Decrement*
		=	DIVS yop, xop ;	*Divide*
		=	DIVQ xop ;	

MAC Instructions

[IF cond]	MR|MF	=	xop * yop ;	*Multiply*
		=	MR + xop * yop ;	*Multiply/Accumulate*
		=	MR – xop * yop ;	*Multiply/Subtract*
		=	MR ;	*Transfer MR*
		=	0 ;	*Clear*
IF MV	SAT MR ;			*Conditional MR Saturation*

Shifter Instructions

[IF cond]	SR = [SR OR] ASHIFT xop ;	*Arithmetic Shift*
[IF cond]	SR = [SR OR] LSHIFT xop ;	*Logical Shift*
	SR = [SR OR] ASHIFT xop BY <exp>;	*Arithmetic Shift Immediate*
	SR = [SR OR] LSHIFT xop BY <exp>;	*Logical Shift Immediate*
[IF cond]	SE = EXP xop ;	*Derive Exponent*
[IF cond]	SB = EXPADJ xop ;	*Block Exponent Adjust*
[IF cond]	SR = [SR OR] NORM xop ;	*Normalize*

Data Move Instructions

reg = reg ;	*Register-to-Register Move*
reg = <data> ;	*Load Register Immediate*
reg = DM (<addr>) ;	*Data Memory Read (Direct Address)*
dreg = DM (Ix , My) ;	*Data Memory Read (Indirect Address)*
dreg = PM (Ix , My) ;	*Program Memory Read (Indirect Address)*
DM (<addr>) = reg ;	*Data Memory Write (Direct Address)*
DM (Ix , My) = dreg ;	*Data Memory Write (Indirect Address)*
PM (Ix , My) = dreg ;	*Program Memory Write (Indirect Address)*

Multifunction Instructions

<ALU>|<MAC>|<SHIFT> , dreg = dreg ;	*Computation with Register-to-Register Move*
<ALU>|<MAC>|<SHIFT> , dreg = DM (Ix , My) ;	*Computation with Memory Read*
<ALU>|<MAC>|<SHIFT> , dreg = PM (Ix , My) ;	*Computation with Memory Read*
DM (Ix , My) = dreg , <ALU>|<MAC>|<SHIFT> ;	*Computation with Memory Write*
PM (Ix , My) = dreg , <ALU>|<MAC>|<SHIFT> ;	*Computation with Memory Write*
dreg = DM (Ix , My) , dreg = PM (Ix , My) ;	*Data & Program Memory Read*
<ALU>|<MAC> , dreg = DM (Ix , My) , dreg = PM (Ix , My) ;	*ALU/MAC with Data & Program Memory Read*

Program Flow Instructions

DO <addr> [UNTIL term] ;	*Do Until Loop*
[IF cond] JUMP (Ix) ;	*Jump*
[IF cond] JUMP <addr>;	
[IF cond] CALL (Ix) ;	*Call Subroutine*
[IF cond] CALL <addr>;	
IF [NOT] FLAG_IN JUMP <addr>;	*Jump/Call on Flag In Pin*
IF [NOT] FLAG_IN CALL <addr>;	
[IF cond] SET|RESET|TOGGLE FLAG_OUT [, ...] ;	*Modify Flag Out Pin*
[IF cond] RTS ;	*Return from Subroutine*
[IF cond] RTI ;	*Return from Interrupt Service Routine*
IDLE [(n)] ;	*Idle*

Miscellaneous Instructions

NOP ;	*No Operation*
MODIFY (Ix , My);	*Modify Address Register*
[PUSH STS] [, POP CNTR] [, POP PC] [, POP LOOP] ;	*Stack Control*
ENA|DIS SEC_REG [, ...] ;	*Mode Control*
BIT_REV	
AV_LATCH	
AR_SAT	
M_MODE	
TIMER	
G_MODE	

Notation Conventions

Ix	Index registers for indirect addressing
My	Modify registers for indirect addressing
<data>	Immediate data value
<addr>	Immediate address value
<exp>	Exponent (shift value) in shift immediate instructions (8-bit signed number)
<ALU>	Any ALU instruction (except divide)
<MAC>	Any multiply-accumulate instruction
<SHIFT>	Any shift instruction (except shift immediate)
cond	Condition code for conditional instruction
term	Termination code for DO UNTIL loop
dreg	Data register (of ALU, MAC, or Shifter)
reg	Any register (including dregs)
;	A semicolon terminates the instruction
,	Commas separate multiple operations of a single instruction
[ ]	Optional part of instruction
[, ...]	Optional, multiple operations of an instruction
option1 | *option2*	List of options; choose one.

Assembly Code Example

The following example is a code fragment that performs the filter tap update for an adaptive filter based on a least-mean-squared algorithm. Notice that the computations in the instructions are written like algebraic equations.

```
        MF=MX0*MY1(RND), MX0=DM(I2,M1);       {MF=error*beta}
        MR=MX0*MF(RND), AY0=PM(I6,M5);
        DO adapt UNTIL CE;
           AR=MR1+AY0, MX0=DM(I2,M1), AY0=PM(I6,M7);
adapt:     PM(I6,M6)=AR, MR=MX0*MF(RND);

        MODIFY(I2,M3);                        {Point to oldest data}
        MODIFY(I6,M7);                        {Point to start of data}
```