

1 INTRODUCTION

Purpose

The ADSP-219x DSP Instruction Set Reference provides assembly syntax information for the ADSP-219x Digital Signal Processor (DSP). The syntax descriptions cover instructions that execute within the DSP's processor core (processing elements, program sequencer, and data address generators). For architecture and design information on the DSP, see the *ADSP-219x/2191 DSP Hardware Reference*.

Audience

DSP system designers and programmers who are familiar with signal processing concepts are the primary audience for this manual. This manual assumes that the audience has a working knowledge of microcomputer technology and DSP-related mathematics.

DSP system designers and programmers who are unfamiliar with signal processing can use this manual, but should supplement this manual with other texts, describing DSP techniques.

All readers, particularly programmers, should refer to the DSP's development tools documentation for software development information. For additional suggested reading, see [“For More Information About Analog Products”](#) on page 1-6.

Contents Overview

This reference presents instruction information organized by the type of the instruction. Instruction types relate to the machine language opcode for the instruction. On this DSP, the opcodes categorize the instructions by the portions of the DSP architecture that execute the instructions. The following chapters cover the different types of instructions:

- [“Instruction Set Summary” on page 2-1](#)—This chapter provides a syntax summary of all instructions and describes the conventions that are used on the instruction reference pages.
- [“ALU Instructions” on page 3-1](#)—These instructions specify operations that occur in the DSP’s ALU.
- [“MAC Instructions” on page 4-1](#)—These instructions specify operations that occur in the DSP’s Multiply–Accumulator.
- [“Shifter Instructions” on page 5-1](#)—These instructions specify operations that occur in the DSP’s Shifter.
- [“Multifunction Instructions” on page 6-1](#)—These instructions specify parallel, single-cycle operations.
- [“Data Move Instructions” on page 7-1](#)—These instructions specify memory and register access operations.
- [“Program Flow Instructions” on page 8-1](#)—These instructions specify program sequencer operations.
- [“Instruction Opcodes” on page 9-1](#)—This chapter lists the instruction encoding fields for all instructions.

Each of the DSP’s instructions is specified in this text. The reference page for an instruction shows the syntax of the instruction, describes its function, gives one or two assembly-language examples, and identifies fields of its opcode. The instructions are referred to by type, ranging from 1 to 37.

These types correspond to the opcodes that ADSP-219x DSPs recognize, but are for reference only and have no bearing on programming.

Some instructions have more than one syntactical form; for example, instruction “[Type 9: Compute](#)” on [page 9-27](#) has many distinct forms.

Many instructions can be conditional. These instructions are prefaced by IF COND; for example:

```
If COND compute;
```

In a conditional instruction, the execution of the entire instruction is based on the specified condition.

Development Tools

The ADSP-219x is supported by VisualDSP®, an easy-to-use project management environment, comprised of an Integrated Development Environment (IDE) and Debugger. VisualDSP lets you manage projects from start to finish from within a single, integrated interface. Because the project development and debug environments are integrated, you can move easily between editing, building, and debugging activities.

Flexible Project Management. The IDE provides flexible project management for the development of DSP applications. The IDE includes access to all the activities necessary to create and debug DSP projects. You can create or modify source files or view listing or map files with the IDE Editor. This powerful Editor is part of the IDE and includes multiple language syntax highlighting, OLE drag and drop, bookmarks, and standard editing operations such as undo/redo, find/replace, copy/paste/cut, and goto.

Also, the IDE includes access to the C Compiler, C Runtime Library, Assembler, Linker, Loader, Simulator, and Splitter. You specify options for these Tools through Property Page dialogs. Property Page dialogs are easy to use, and make configuring, changing, and managing your projects

Development Tools

simple. These options control how the tools process inputs and generate outputs, and have a one-to-one correspondence to the tools' command line switches. You can define these options once, or modify them to meet changing development needs. You can also access the Tools from the operating system command line if you choose.

Greatly Reduced Debugging Time. The Debugger has an easy-to-use, common interface for all processor simulators and emulators available through Analog Devices and third parties or custom developments. The Debugger has many features that greatly reduce debugging time. You can view C source interspersed with the resulting Assembly code. You can profile execution of a range of instructions in a program; set simulated watch points on hardware and software registers, program and data memory; and trace instruction execution and memory accesses. These features enable you to correct coding errors, identify bottlenecks, and examine DSP performance. You can use the custom register option to select any combination of registers to view in a single window. The Debugger can also generate inputs, outputs, and interrupts so you can simulate real world application conditions.

Software Development Tools. Software Development Tools, which support the ADSP-219x Family, allow you to develop applications that take full advantage of the DSP architecture, including shared memory and memory overlays. Software Development Tools include C Compiler, C Runtime Library, DSP and Math Libraries, Assembler, Linker, Loader, Simulator, and Splitter.

C Compiler & Assembler. The C Compiler generates efficient code that is optimized for both code density and execution time. The C Compiler allows you to include Assembly language statements inline. Because of this, you can program in C and still use Assembly for time-critical loops. You can also use pretested Math, DSP, and C Runtime Library routines to help shorten your time to market. The ADSP-219x Family Assembly language is based on an algebraic syntax that is easy to learn, program, and

debug. The add instruction, for example, is written in the same manner as the actual equation (for example, $AR = AX0 + AY0$);).

Linker & Loader. The Linker provides flexible system definition through Linker Description Files (.LDF). In a single LDF, you can define different types of executables for a single or multiprocessor system. The Linker resolves symbols over multiple executables, maximizes memory use, and easily shares common code among multiple processors. The Loader supports creation of host (8- or 16-bit) port, SPI port, UART port, and PROM boot images. Along with the Linker, the Loader allows a variety of system configurations with smaller code and faster boot time.

Simulator. The Simulator is a cycle-accurate, instruction-level simulator — allowing you to simulate your application in real time.

3rd-Party Extensible. The VisualDSP environment enables third-party companies to add value using Analog Devices' published set of Application Programming Interfaces (API). Third party products—runtime operating systems, emulators, high-level language compilers, multiprocessor hardware —can interface seamlessly with VisualDSP thereby simplifying the tools integration task. VisualDSP follows the COM API format. Two API tools, Target Wizard and API Tester, are also available for use with the API set. These tools help speed the time-to-market for vendor products. Target Wizard builds the programming shell based on API features the vendor requires. The API tester exercises the individual features independently of VisualDSP. Third parties can use a subset of these APIs that meets their application needs. The interfaces are fully supported and backward compatible.

Further details and ordering information are available in the VisualDSP Development Tools data sheet. This data sheet can be requested from any Analog Devices sales office or distributor.

For More Information About Analog Products

Analog Devices is online on the internet at <http://www.analog.com>. Our Web pages provide information on the company and products, including access to technical information and documentation, product overviews, and product announcements.

You may also obtain additional information about Analog Devices and its products in any of the following ways:

- Visit our World Wide Web site at www.analog.com
- FAX questions or requests for information to 1(781)461-3010.
- Access the Computer Products Division File Transfer Protocol (FTP) site at [ftp ftp.analog.com](ftp://ftp.analog.com) or [ftp 137.71.23.21](ftp://137.71.23.21) or <ftp://ftp.analog.com>.

For Technical or Customer Support

You can reach our Customer Support group in the following ways:

- E-mail questions to dsp.support@analog.com or dsp.europe@analog.com (European customer support)
- Telex questions to 924491, TWX:710/394-6577
- Cable questions to ANALOG NORWOODMASS
- Contact your local ADI sales office or an authorized ADI distributor

- Send questions by mail to:

Analog Devices, Inc.
DSP Division
One Technology Way
P.O. Box 9106
Norwood, MA 02062-9106
USA

What's New in This Manual

This is the preliminary edition of the ADSP-219x DSP Instruction Set Reference. Summaries of changes between editions will start with the next edition.

Related Documents

For more information about Analog Devices DSPs and development products, see the following documents:

- *ADSP-2191 DSP Microcomputer Data Sheet*
- *ADSP-219x/2191 DSP Hardware Reference*
- *Getting Started Guide for VisualDSP & ADSP-219x Family DSPs*
- *VisualDSP User's Guide for ADSP-219x Family DSPs*
- *C Compiler & Library Manual for ADSP-219x Family DSPs*
- *Assembler Manual for ADSP-219x Family DSPs*
- *Linker & Utilities Manual for ADSP-219x Family DSPs*

All the manuals are included in the software distribution CD-ROM. To access these manuals, use the Help Topics command in the VisualDSP environment's Help menu and select the Online Manuals book. From this

Conventions

Help topic, you can open any of the manuals, which are in Adobe Acrobat PDF format.

Conventions

The following are conventions that apply to all chapters. Note that additional conventions, which apply only to specific chapters, appear throughout this document.

Table 1-1. Instruction set notation

Notation	Meaning
UPPERCASE	Explicit syntax— assembler keyword. (The assembler is case-insensitive.)
;	A semicolon terminates an instruction line.
,	A comma separates multiple, parallel instructions in the same instruction line.
// single line comment /* multi line comment */	// or /* */ indicate comments or remarks that explain program code, but that the assembler ignores. For more details, see the <i>Assembler Manual for ADSP-219x Family DSPs</i> .
option1 option2	You must choose one of the items enclosed within two vertical bars.
[(DB)]	Brackets enclose an optional instruction component. The option's syntax includes everything within the brackets, as shown. In this case, (DB) is the delayed branch option.
<imm#> <data#>	Denotes an immediate value (data or address) of # bits.
COND	Denotes a status condition.

Table 1-1. Instruction set notation (Cont'd)

Notation	Meaning
TERM	Denotes a loop termination condition. For details, see “Type 11: Do ... Until” on page 9-34 .
Dreg	Denotes an unrestricted (Group 0) data register used as either an x or y -input operand. For details, see “Core Registers” on page 7-2 .
XOP	Denotes a restricted data register used for the x -input operand in a compute instruction. For details, see “Input Registers” on page 3-2 .
YOP	Denotes a restricted data register used for the y -input operand in a compute operation. For details, see “Input Registers” on page 3-2 .
Ireg	DAG address register. Denotes an index register (I0–I7).
Mreg	DAG address register. Denotes a modify register (M0–M7).
Lreg	DAG address register. Denotes a length register (L0–L7).
Breg	DAG address register. Denotes a base register (B0–B7).
(Ireg + Mreg)	Indirect addressing mode. Denotes premodify addressing with no update. Same as (Mreg, Ireg) syntax.
(Ireg += Mreg)	Indirect addressing mode. Denotes postmodify addressing with update. Same as (Ireg, Mreg) syntax.
0x	Denotes number in hexadecimal format (0xFFFF).
h#	Denotes number in hexadecimal format (h#FFFF).
b#	Denotes number in binary format (b#0001000100010001).

Conventions