Outline

Instruction Set

Instruction types
- Arithmetic and multiplication
- Logic operations
- Shifting and rotating
- Comparison
- Instruction flow control (looping, branch, call, and return)
- Conditional instruction execution
- Special function instructions

Registers
- Accumulators
- General- and special-purpose registers
- Address registers
- Others

Parallel move support

Orthogonality

Assembly language format
Arithmetic and Multiplication, and Logic Operations

Arithmetic and multiplication
- Critical functions for DSP applications
- Add, subtract, increment, decrement, negate, round, absolute value, and multiplication
- Multiply-accumulate except TI TMS320C1x
- Square with indirectly addressed memory as operands
  - DSP Group Pine and Oak, and TI TMS320C5x
- Support for extended-precision arithmetic
  - Signed/signed, signed/unsigned, and unsigned/unsigned multiplication
  - Add-with-carry and subtract-with-borrow

Logic operations
- Logical \textit{and}, \textit{or}, \textit{exclusive-or}, and \textit{not}
- Used in error correction and decision-processing applications
Shifting

Provides a way of scaling data
- Important on fixed-point processors

Logical
- Left shift: inserts zero bits in the least significant bits by the shift amount
- Right shift: inserts zero bits in the most significant bits by the shift amount

Arithmetic
- Left shift: the same as the logical left shift
- Right shift: inserts one or zero bits (depending on the sign bit) in the most significant bits by the shift amount

The form of shift instructions (operations)
- One-bit shift
- Two- or four-bit shift
  - Can be combined with one-bit shifts to synthesize n-bit shifts
- Barrel shifter
  - AT&T DSP16xx, Analog Devices ADSP-21xx, etc.
Rotation

Circular shifting
- The bits that are shifted off one end of the word loop around and are shifted in on the other end

Example
- In a left rotate by one bit
  - The most significant bit is rotated into the least significant bit, and all other bits shift left by one position

Applications
- Error correction coding (e.g., for bit interleaving)
- Slow generation of bit-reversed addresses on processors that do not have bit-reversal built in to their address generation units

Most processors provide to rotate a word by one bit, except
- DSP16xx, NEC μPD7701x, Z893xx, and ZR3800x
Comparison

Status bits
- Provide information about the results of arithmetic (or logical) operations
- N (minus), Z (zero), V (overflow), and C (carry or borrow)
- Used in conditional branches or conditional execution instructions

A processor may need to compare a series of values to a known value stored in a register and obtain status bits as results
- Subtract
  - Changes the reference value or another register
  - Not an attractive solution on processors with few registers

Comparison instructions
- Perform subtraction for obtaining status bits without modifying any register
- Enhancement to normal compare: compare absolute value
Program Flow Control Instructions

Loop

- Repeated execution of a small number of arithmetic or multiplications instructions
  - The overhead imposed by instructions used to decrement and test a counter and branch to the start of the loop may be relatively large

- So, *hardware looping* instructions are provided
  - Allow a single instruction or block of instructions to be repeated a number of times without the overhead that would normally come from the decrement-test-branch sequence

Branches

- Conditional or Unconditional
- PC-relative or not
  - PC-relative branches are important in applications that require position-independent code
- Call/return or jump
  - In call, the return address is saved automatically in a particular storage
- Delayed, multicycle (nondelayed), or delayed with nullify
Conditional Instruction Execution

- Allow the programmer to specify that an instruction is to be executed if a specified condition is true
- Free the programmer from using branch instructions simply to execute a single instruction as part of *if-then-else* construct
- Useful on processors with deep pipelines

**Examples**
- Analog Devices ADSP-21xx and ADSP-210xx
  - Support conditional execution for most instructions
- AT&T DSP16xx and DSP Group PineDSPCore
  - Provide conditional execution of far fewer instructions
- TI TMS320C5x and TMS320C54x
  - A special conditional instruction XC
    - If the condition specified as XC’s argument is true, the next one or two single-word instructions (or the next two-word instructions) are executed
    - Otherwise, NOPs are executed instead
Special Function Instructions

- A variety of specialized instructions are provided on some DSPs
- Block-floating-point instructions
- Bit manipulation instructions
- Other special instructions
  - An iterative division instruction
    - Used to perform division one bit at a time
  - Square root seed instruction
    - Used as a basis for finding the square root of a number
    - Analog Devices ADSP-210xx and Texas Instruments TMS320C4x
  - Specialized stack operations such as push and pop
  - Interrupt enable and disable instructions
Block Floating-Point Instructions

- A form of floating-point arithmetic that is sometimes used on fixed-point processors
  - Used in applications such as speech coding and some implementations of the FFT
  - Makes heavy use of two operations
    - **Exponent detection**
      - Determines the number of redundant bits (the number of leading one or zero)
      - Sometimes used separately from normalization to determine the maximum exponent in a block of numbers
    - **Normalization**
      - Exponent detection combined with a left shift that scales the data value so that it contains no redundant sign bits
Three Basic Categories in Exponent Detection and Normalization

**Block Exponent detection**
- Used to determine the maximum exponent in a block of data
- Useful for manipulating arrays of blocks floating-point data
- Analog Devices ADSP-21xx and Zoran ZR3800x

**Exponent detection**
- Computes exponent of a single data value in one instruction cycle
- ADSP-21xx, OakDSPCore, μPD7701x, SGS-Thomson D950-CORE, TMS320C54x, and ZR3800

**Normalization**
- A single-cycle normalize instruction
  - AT&T DSP16xx
- Two-cycle normalization using exponent detect instruction and a shift instruction
  - DSP processors supporting single-cycle exponent detection
- Iterative normalization
  - Normalize a data value one bit at a time
  - PineDSPCore, DSP5600x, DSP561xx, TMS320C2x, and TMS320C5x
Bit Manipulation Instructions

- Useful for both decision processing and error correction coding

- Single-bit manipulation instructions
  - *bit set*, *bit clear*, *bit toggle*, and *bit test*
  - Motorola DSP5600x was one of the first to support bit manipulation instructions (two-cycle execution)

- *branch-if-bit-set* and *branch-if-bit clear*
  - Motorola processors

- Multi-bit (bit-field) manipulation instructions
  - Operate on several bits at one

- Examples
  - Logical operation (*and*, *or*, *not*, *exclusive-or*) on a specified memory (TI TMS320C5x)
  - bit-field test, set, and clear instructions (Motorola DSP561xx)
  - Bit-field *extract* and *replace* instructions (AT&T DSP16xx)
Registers

- Registers are closely coupled to the instruction set
- Instructions typically specify registers as source or destination registers, or use them to generate addresses

- Abundance of registers
- Makes the processor easier to program
- But increases the instruction width and die size

- Types of Registers
  - Accumulators
  - Address registers
  - General-and special-purpose registers
  - Other registers
    - Stack pointer, program counter, loop registers
    - Status register, control register, link register
Accumulators and Address Registers

Accumulators

- A register that is at least wide enough to hold the largest ALU or multiplier result produced by the data path
- Used as a source or destination for arithmetic operations
- At least one accumulator exists
- A single primary accumulator
  - TI TMS320c1x, TMS320C2x, and TMS320C5x
- Two or more accumulators
  - Simplify coding of algorithms that use complex numbers
  - DSP16xx, PineDSPCore, OakDSPCore, DSP5600x, etc.
  - Some floating-point processors provide a large number of extended-precision registers
    - Some of them can be used as accumulators

Address registers

- Used to generate addresses for register-indirect addressing
- Also used to manage circular buffers
- From 2 (on TI TMS20C1x) to 22 (AT&T DSP32xx) registers
General- and Special-Purpose Registers

General-purpose registers
- A bank of registers for general purpose use (often called a register file)
- Used for most arithmetic and multiplication instructions
- Preferred, but more expensive than special-purpose
- ADSP-2100x, DSP32xx, μPD7701x TMS3203x, etc.

Special registers
- Dedicate registers to certain execution units
- Examples
  - ADSP-21xx
    - The multiplier, ALU and shifter each has its own dedicated input and output registers
  - DSP16xx, Z893xx, and PineDSPCore
    - Dedicated multiplier output registers
- Execution unit input/output registers also used as general-purpose registers
  - Multiplier input registers (X0, X1, Y0, Y1) in DSP5600x
Parallel Move Support

- Ability to perform multiple memory accesses in a single instruction cycle
  - One of the most important features
  - Often called parallel moves because they often occur in parallel with an arithmetic or multiplication operation

- Two forms
  - depending on whether or not the data values being moved are directly related to the operands of the ALU or multiply instruction being executed
    - Operand-related
    - Operand-unrelated

- TI TMS320C1x
  - Allows only one data memory access per instruction cycle
Operand-Related Parallel Moves

- Only instructions that have multiple source operands can make multiple data memory accesses per instruction cycle
- Accesses are limited to those required by the arithmetic or multiplication operation
- Example
  
  \[ \text{MPY} \ (R0), (R4) \]

- Related to data-stationary assembly coding styles

DSP processors

- AT&T DSP32C and DSP32xx
- DSPGroup PineDSPCore and OakDSPCore
- TI TMS320C2x and TMS320C5x
- Zilog Z893xx
Operand-Unrelated Parallel Moves

In parallel with a multiply or ALU operation, the processor can access multiple memory locations that may be unrelated to the operands of the ALU instructions.

- Allow the programmer to more efficiently use the memory bandwidth.
- Useful in algorithms that require frequent, nonsequential memory accesses, such as the FFT.

Example:

```
MPY  X0, Y0, A   X:(R0)+, X0  Y1, Y:(R4)+
```

Related to time-stationary assembly coding styles.

DSP processors

- All Motorola DSP processors
- NEC μPD7701x, SGS-Thomson D950-CORE, and Zoran ZR3800xx
- TI TMS320C3x, TMS320C4x, and TMS320C54x
  - Support both (related and unrelated) for different kinds of instructions.
Orthogonality

The extent to which a processor’s instruction set is consistent
- The more orthogonal, the easier to program
  - Fewer inconsistencies and special cases
- Subjective topic

Two major areas that most influence the orthogonality
- The consistency and completeness of its instruction set
  - Processor with add but not subtract is nonorthogonal
- The degree to which operands and address modes are uniformly available with different operations
  - Register-indirect addressing with add but not with subtract is nonorthogonal

Larger instruction word widths -> more orthogonal
- Independent encodings of operations and operands within an instruction word
- Increase required bus and memory width (system cost)
- Need for a smaller, less orthogonal, harder to program, but more memory-efficient instruction set
Example of a Fully Independent Instruction Encoding on a Hypothetical Processor

- 32 instructions (operation types) with three operands
  - Each operand can be one of eight registers
- Two parallel data moves
  - Eight address registers and three address register update modes

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Instruction Set Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction code</td>
<td>One of 32 instructions</td>
<td>5</td>
</tr>
<tr>
<td>Operands</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One of 8 primary source operands</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>One of 8 secondary source operands</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>One of 8 destination operands</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>First parallel move</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One of 8 primary register-indirect address registers</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>One of 3 register-indirect update modes</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>One of 8 source or destination registers</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Second parallel move</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One of 8 secondary register-indirect address registers</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>One of 3 register-indirect update modes</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>One of 8 source or destination registers</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>
Approaches for Squeezing More Functionality into a Smaller Instruction Word Width

- Reduced number of operations
  - Supporting fewer operations frees instruction word bits
  - AT&T DSP16xx does not have a rotation instruction

- Reduced number of addressing modes
  - A few addressing modes
  - Limit update modes in register-indirect addressing
  - Limit the allowable combinations of operations and addressing modes

- Restrictions on source/destination operands
  - By implied addressing

- Use of mode bits
  - Mode bits determine what an instruction does
  - TI TMS320C5x
    - Mode bit determines whether the shift instruction is arithmetic or logical

- Increase programming difficulty but reduce cost
Assembly Language Format

- **opcode-operand style**
  - Expresses instructions in terms of an instruction mnemonic and its operands
    - MPY X0, Y0
    - ADD P, A
    - MOV (R0), X0
    - JMP LOOP

- **C-like, or algebraic style**
  - Capitalizes on the syntax of the C programming language and its arithmetic shorthand
    - P = X0 * Y0
    - A = P + A
    - X0 = *R0
    - GOTO LOOP

- A single processor can have both types of assemblers