A Short Introduction to DSP Microprocessor Architecture
What makes a DSP chip a DSP?

- Conventional microprocessors use the Von Neumann architecture: program and data all in a single memory. Address and data buses are \textit{shared} between instruction and data fetches.
• Von Neumann architecture is inexpensive, simple, and effective, BUT there are performance problems:
  – Von Neumann “bottleneck”: fetch for next instruction collides with data fetch/store
  – Buses may be idle during instruction decode
  – DSP algorithms often have “multiply-accumulate” requirements: \( \text{coef}[n] \times \text{data}[n] \), where two operands must be fetched

• Most DSP chips use Harvard architecture: separate memory space(s) for program and data
Harvard Architecture

CPU/ALU

Program Memory

Data Memory #1

Data Memory #2

P Address

Instr.

D1 Address

D1 Data

D2 Address

D2 Data
DSP Architectural Features

- ALU typically centered around Multiply-Accumulate (MAC) structure with large accumulator
  - Digital filters require accumulated sum-of-products
- Multiple address generators to handle separate memory spaces
  - Address units handle modulo buffer arithmetic
DSP Data Representation

- Numerical values represented as *binary fractions*: $-1.0 \leq \text{value} < 1.0$
Why a fractional representation?

- The product of two fractional numbers is also a fractional number
- Normalized representation is convenient
- Coefficients from digital filter designs are typically already in fractional form
DSP Architecture: Accumulator

- Accumulator register holds intermediate results (n-bit number \( \times \) n-bit number yields 2n-1 bit number)
- Accumulator typically has extra “guard bits” or “extension register” for overflow
Accumulator Example

Motorola 56xxx has two 56-bit accumulators (48-bit result with 8 guard bits)
Digital Filter Example

- Simple FIR filter is given by

\[ y[n] = \sum_{i=0}^{N-1} b_i \cdot x[n-i] \]

- Current output is *sum* of product of *coefficients* and *past input* values.
Filter example (cont.)

\[ x[n] \rightarrow Z^{-1} \rightarrow x[n-1] \rightarrow Z^{-1} \rightarrow x[n-2] \rightarrow Z^{-1} \rightarrow x[n-3] \rightarrow + \rightarrow y[n] \]

\[ b_0, b_1, b_2, b_3 \]
Filter example (cont.)

- Procedure:
  - Clear accumulator
  - Fetch coefficient and data
  - MAC
  - Repeat fetch & MAC until done
DSP Support for Parallel Moves

- Need to fetch next coefficient and next stored value at each step in the filter
- DSPs generally support a *parallel move* or *fetch* operation while MAC is computed
- This design avoids idle ALU and data buses

Ex:

```plaintext
mac x0, y0, a   x:(r0)+,x0   y:(r4)+,y0
```
Summary

- DSP chips use the Harvard architecture: separate program and data memory spaces
- ALU is centered around the multiply-accumulate (MAC) function
- DSPs typically use a fractional number representation
- Address computation generally supports modulo buffer address arithmetic
- DSPs avoid idle cycles by allowing parallel actions