DSP markets

Industry analysts expect the Internet audio market to grow to more than 30 million units by 2003.

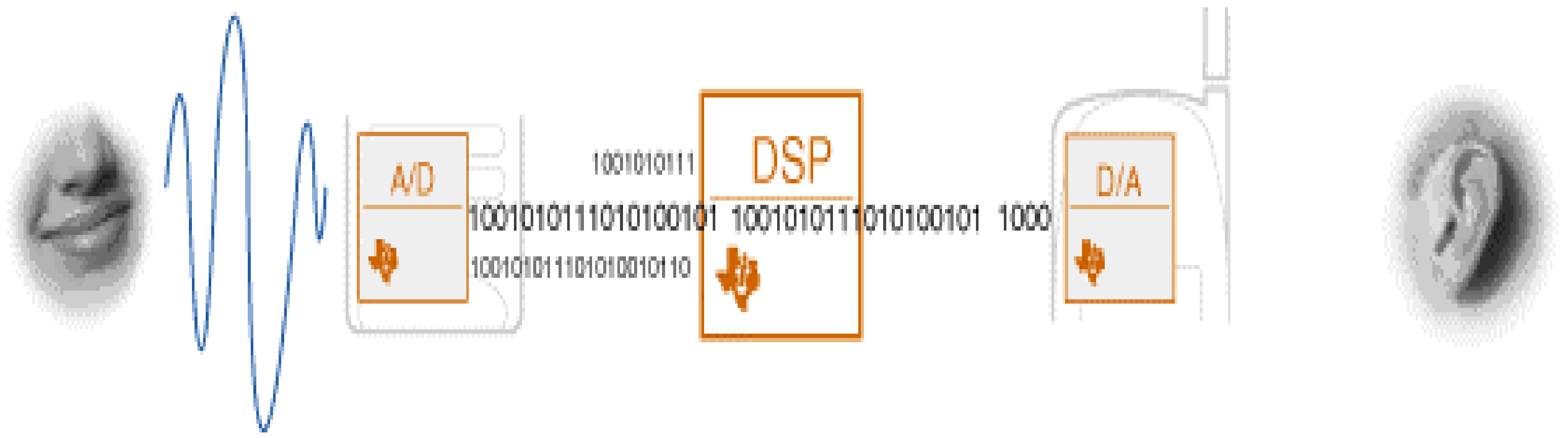
Analysts expect Internet telephony to be a $4 billion market by 2002.

DSP markets

Eight of the world's top 10 wireless network equipment providers have chosen TI DSPs.

TI forecasts that within five years, advanced wireless devices will reach 15% of the total digital cellular phone market.

Typical DSP application



When you speak, your voice is picked up by an analog sensor in the cell phone's microphone.

An analog-to-digital converter chip converts your voice, which is an analog signal, into digital signals, represented by 1s and 0s.

The DSP compresses the digital signals and removes any background noise.

In the listener's cell phone, a digital-to-analog converter chip changes the digital signals back to an analog voice signal.

Your voice exits the phone through the speaker.

TI DSP History: Modem applications

1982 TMS32010, TI introduces its first programmable general-purpose DSP to market

- Operating at 5 MIPS.
- It was ideal for modems and defense applications.

1988 TMS320C3x, TI introduces the industry's first floating-point DSP.

- High-performance applications demanding floating-point performance include voice/fax mail, 3-D graphics, bar-code scanners
  and video conferencing audio and visual systems.
- TMS320C1x, the world's first DSP-based hearing aid uses TI's DSP.

TI DSP History: Telecommunications applications

1989 TMS320C5x, TI introduces highest performance fixed-point DSP generation in the industry, operating at 28 MIPS.

- The ‘C5x delivers 2 to 4 times the performance of any other fixed-point DSP.

- Targeted to the industrial, communications, computer and automotive segments, the ‘C5x DSPs are used mainly in
    - cellular and cordless telephones,
    - high-speed modems,
    - printers and
    - copiers.

TI DSP History: Automobile applications

1992 DSPs become one of the fastest growing segments within the automobile electronics market.

The math-intensive, real-time calculating capabilities of DSPs provide future solutions for

- **active suspension,**
- **closed-loop engine control systems,**
- **intelligent cruise control radar systems,**
- **anti-skid braking systems and**
- **car entertainment systems.**

Cadillac introduces the 1993 model Allante featuring a TI DSP-based road sensing system for a smoother ride, less roll and tighter cornering.

TI DSP History: Hard Disk Drive applications

1994 DSP technology enables the first uniprocessor DSP hard disc drive (HDD) from Maxtor Corp.

the 171-megabyte PCMCIA Type III HDD.

- By replacing a number of microcontrollers, drive costs were cut by 30 percent while battery life was extended and storage capacity increased.
- In 1994, more than 95 percent of all high performance disk drives with a DSP inside contain a TI TMS320 DSP.

1996 TI's T320C2xLP cDSP technology enables Seagate, one of the world’s largest hard disk drive (HDD) maker, to develop the first mainstream 3.5-inch HDD to adopt a uniprocessor DSP design, integrating logic, flash memory, and a DSP core into a single unit.

TI DSP History: Internet applications

1999 Provides the first complete DSP-based solution, for the secure downloading of music off the Internet onto portable audio devices, with Liquid Audio Inc., the Fraunhofer Institute for Integrated Circuits and SanDisk Corp.

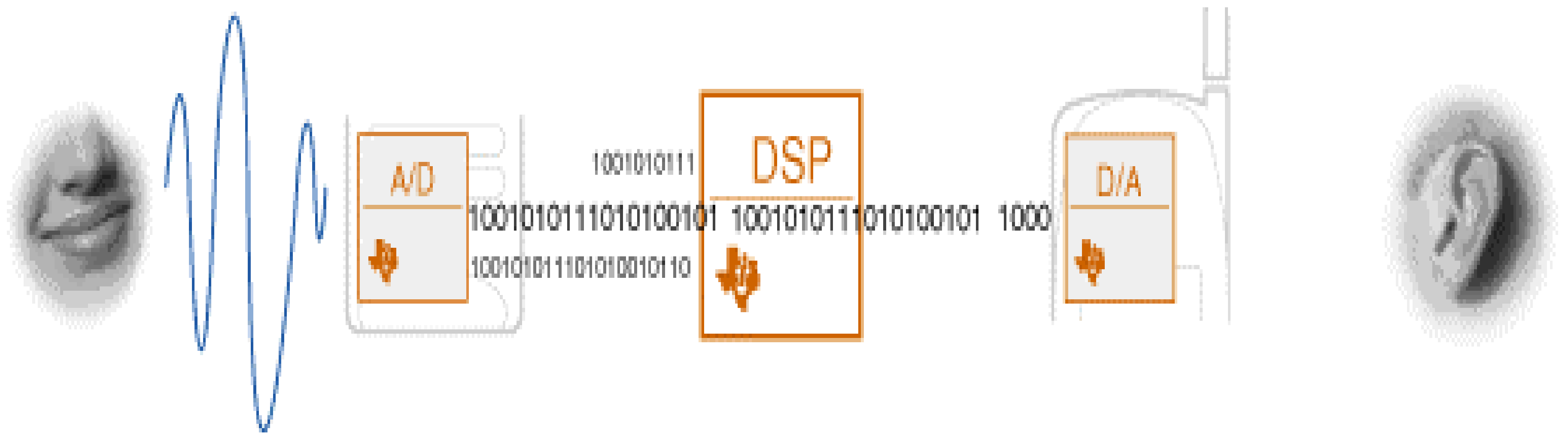
Announces that SANYO Electric Co., Ltd. will deliver the first Secure Digital Music Initiative (SDMI)-compliant portable digital music player based on TI's TMS320C5000 programmable DSPs and Liquid Audio's Secure Portable Player Platform (SP3).

Announces the industry's first 1.2 Volt TMS320C54x DSP that extends the battery life for applications such as cochlear implants, hearing aids and wireless and telephony devices.

Outline

- DSP architectural basics
- Improved performance through increased parallelism
  - Allowing more operations per instruction
    - Enhanced conventional DSPs
    - Single-instruction, multiple-data (SIMD)
  - Issuing multiple instructions per instruction cycle
    - VLIW DSPs
    - Superscalar DSPs
- CPUs with SIMD extensions
- DSP/microcontroller hybrids

Typical DSP application



When you speak, your voice is picked up by an analog sensor in the cell phone's microphone.

An analog-to-digital converter chip converts your voice, which is an analog signal, into digital signals, represented by 1s and 0s.

The DSP compresses the digital signals and removes any background noise.

In the listener's cell phone, a digital-to-analog converter chip changes the digital signals back to an analog voice signal.

Your voice exits the phone through the speaker.

DSP - Architecture Characteristics

DSP architecture is designed to solve one problem well

- Digital filters (FIR, IIR, and FFTs)
- In Real-Time

Architecture features added to speed up this problem

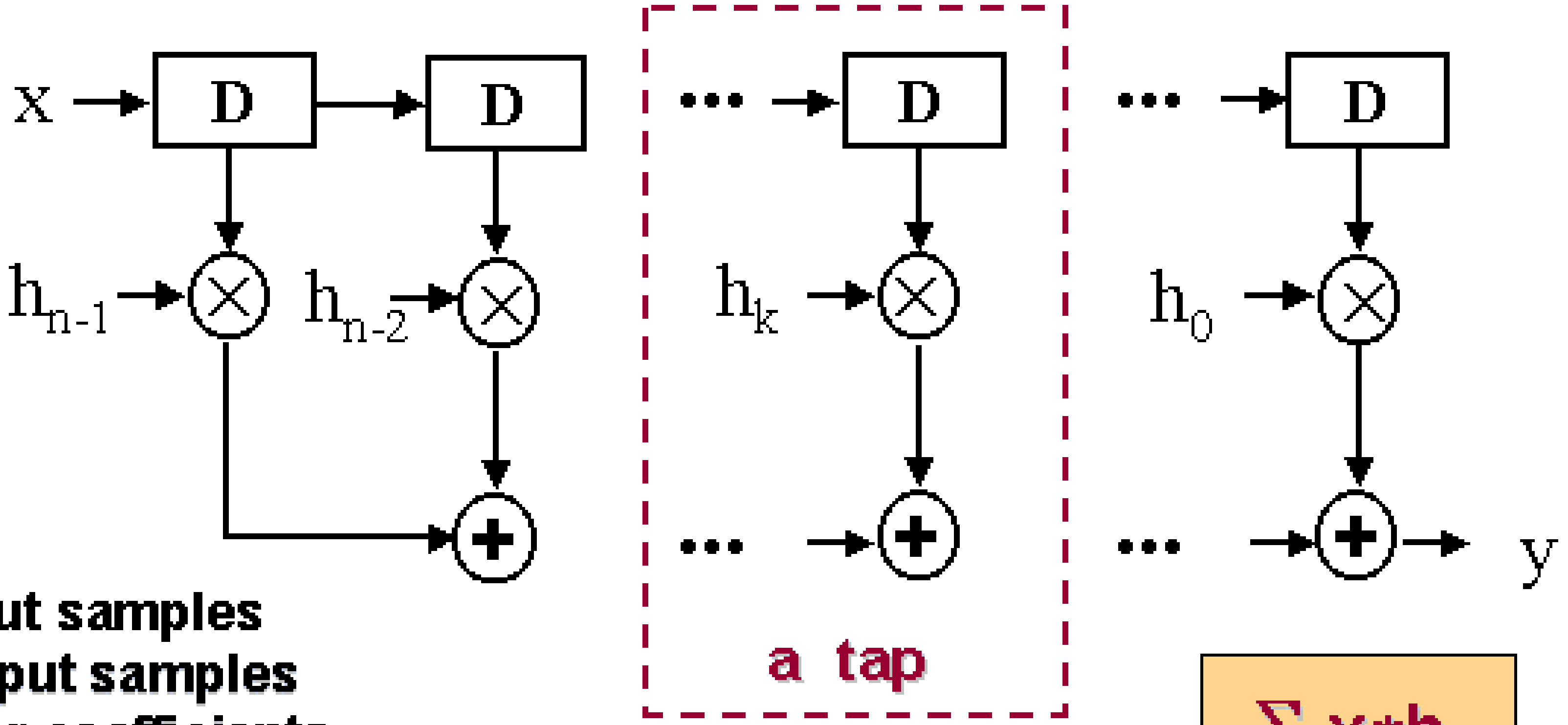
MAC: multiply & accumulator, speedup FIR tap
Circular buffer: speedup shifting FIR delay registers
RISC based: single clock per instruction
Harvard Architecture: separate instruction & data
Word orientated

Disadvantages (not a general purpose computer)

slow character processing
No multi-user operating system support
No virtual memory, no translate lookside tables
No memory page protection (Read, Write, Execute)

A Motivating Example:

FIR Filtering



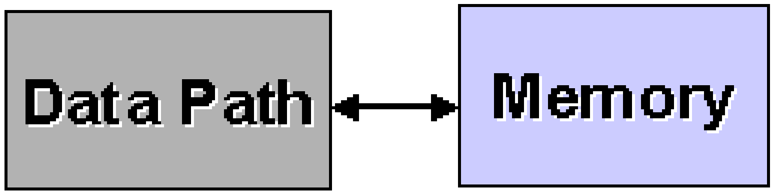
x = input samples
y = output samples
h = filter coefficients
D = unit time delay

Σ x*h

Filter characteristics governed by number of taps, selection of filter coefficients.

FIR Filter on a Typical GPP

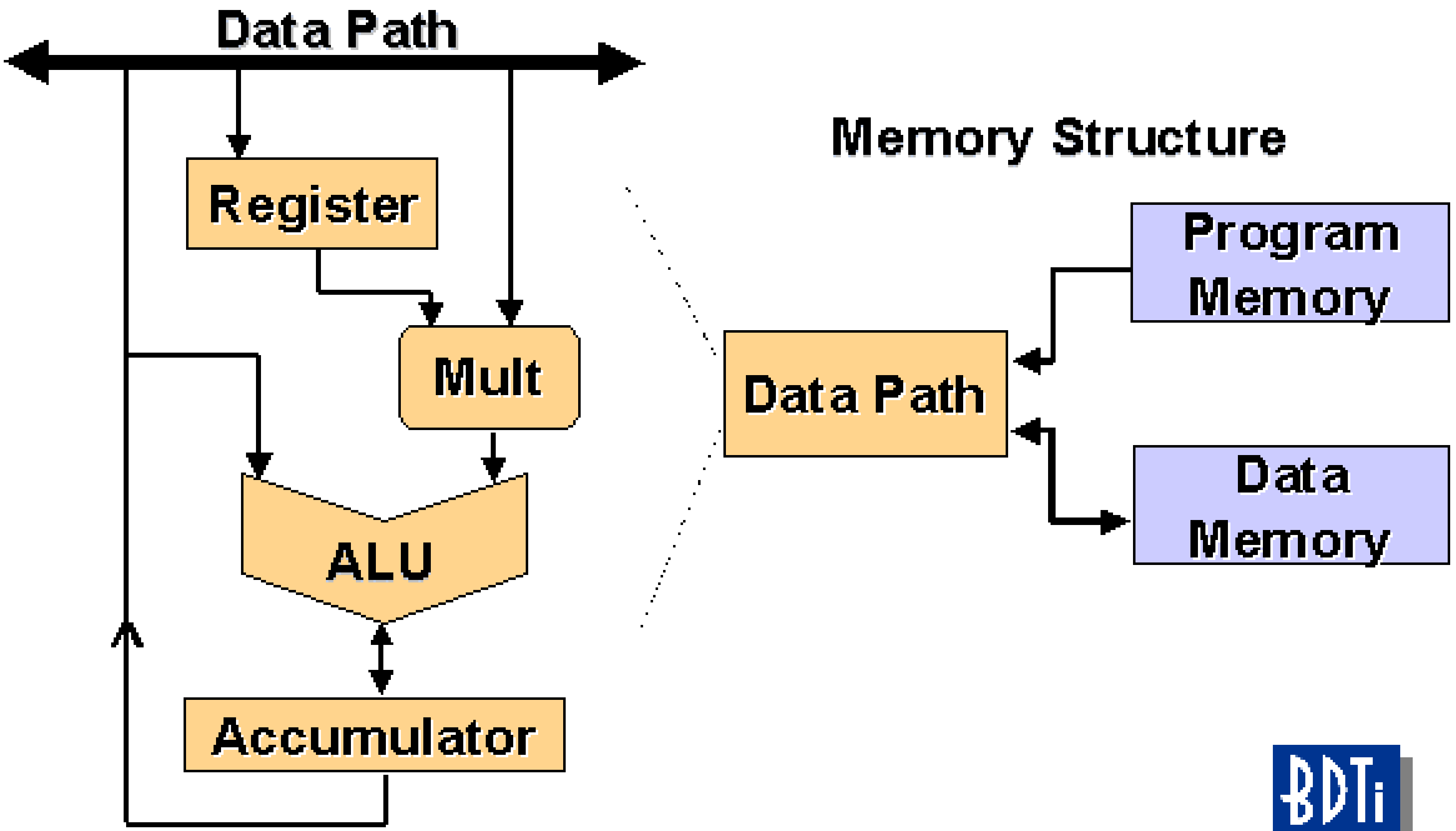
```
loop:
        mov     *r0,x0
        mov     *r1,y0
        mpy     x0,y0,a
        add     a,b
        mov     y0,*r2
        inc     r0
        inc     r1
        inc     r2
        dec     ctr
        tst     ctr
        jnz     loop
```



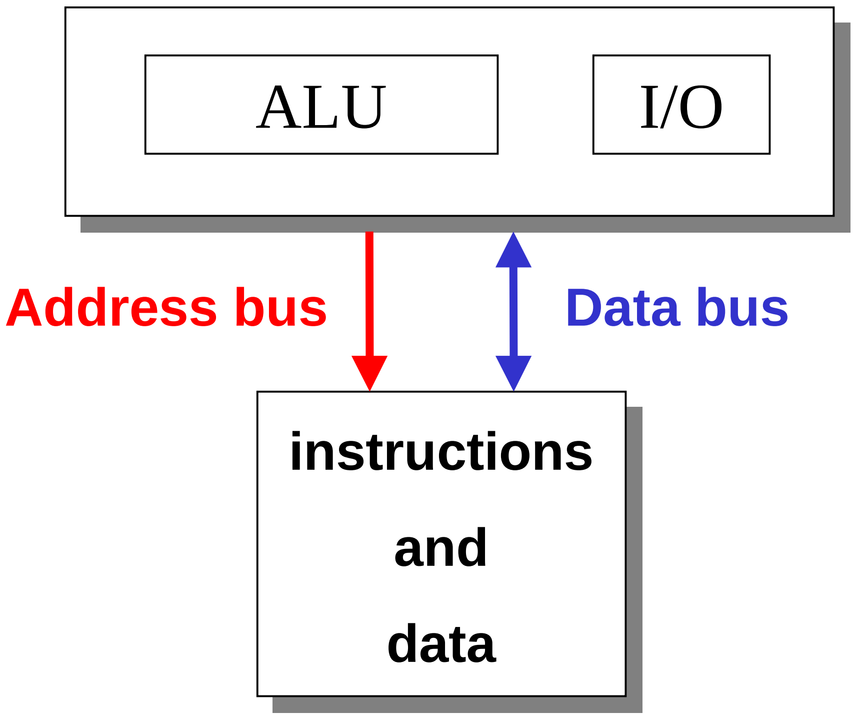
Problems:
- Memory bandwidth bottleneck
- Control code and addressing overhead
- Possibly slow multiply

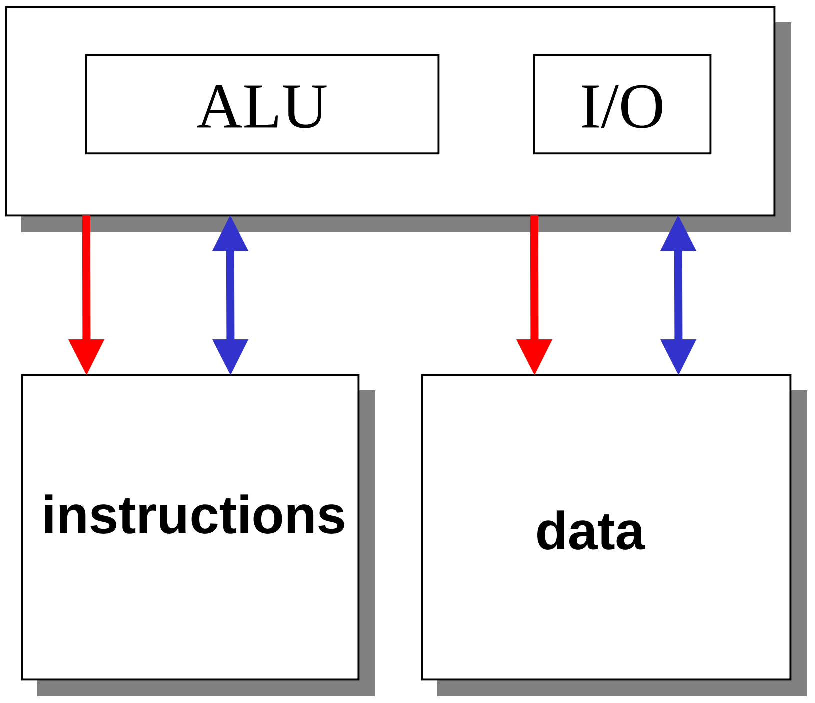
(Computes one tap per loop iteration)

Early DSP Architecture



Von Neuman & Harvard Architectures





<u>Von Neuman architecture</u>
Area efficient but requires higher bus bandwidth because instructions and data must compete for memory.

<u>Harvard architecture</u> **was coined to describe machines with separate memories. Speed efficient: Increased parallelism.**

FIR Filter on a Conventional DSP

```
 DO dotprod UNTIL CE;
dotprod:
  MR=MR+MX0*MY0(SS),MX0=DM(I0,M0),MY0=PM(I4,M4);
```

Baseline: "Conventional DSPs"

- Common attributes:
  - 16- or 24-bit fixed-point (fractional), or 32-bit floating-point arithmetic
  - 16-, 24-, 32-bit instructions
  - One instruction per cycle ("single issue")
  - Complex, "compound" instructions encoding many operations
  - Highly constrained, non-orthogonal architectures

Baseline: "Conventional DSPs"

- Common attributes (cont.):
  - Dedicated addressing hardware w/ specialized addressing modes
  - Multiple-access on-chip memory architecture
  - Dedicated hardware for loops and other execution control
  - Specialized on-chip peripherals and I/O interfaces
  - Low cost, low power, low memory usage

Increasing Parallelism

- Boosting performance beyond the increases afforded by faster clock speeds requires the processor to do more work in every clock cycle. How?
- By increasing the processors' parallelism in one of the following ways:
  - Increase the number of operations that can be performed in each instruction
  - Increase the number of instructions that can be issued and executed in every cycle

Reduced Instruction Set Computer

RISC - Reduced Instruction Set Computer

- By reducing the number of instructions that a processor supports and thereby reducing the complexity of the chip,
- it is possible to make individual instructions execute faster and achieve a net gain in performance
- even though more instructions might be required to accomplish a task.

RISC trades-off

instruction set complexity for instruction execution timing.

RISC Features

- Large register set: having more registers allows memory access to be minimized.
- Load/Store architecture: operating data in memory directly is one of the most expensive in terms of clock cycle.
- Fixed length instruction encoding: This simplifies instruction fetching and decoding logic and allows easy implementation of pipelining.

All instructions are register-to-register format
except Load/Store which access memory

All instructions execute in a single cycle
save branch instructions which require two.

Almost all single instruction size & same format.

Complex Instruction Set Computer

CISC - Complex Instruction Set Computer

Philosophy: Hardware is always faster than the software.

Objective: Instruction set should be as powerful as possible

With a power instruction set, fewer instructions needed to complete (and less memory) the same task as RISC.

- **CISC was developed at a time (early 60's), when memory technology was not so advanced.**
- **Memory was small (in terms of kilobytes) and expensive.**

But for embedded systems, especially Internet Appliances, memory efficiency comes into play again, especially in chip area and power.

Comparison

CISC	RISC
Any instruction may reference memory	Only load/store references memory
Many instructions & addressing modes	Few instructions & addressing modes
Variable instruction formats	Fixed instruction formats
Single register set	Multiple register sets
<u>Multi-clock</u> cycle instructions	<u>Single-clock</u> cycle instructions
Micro-program interprets instructions	Hardware (FSM) executes instructions
Complexity is in the micro-program	Complexity is in the compiler
Less to no pipelining	Highly pipelined
Program code size small	Program code size large

Which is better

RISC

Or

CISC

?

Analogy (Chakravarty, 1994)

Construct a 5 foot wall

Method A: a large amount of small concrete blocks.

Method B: a few large concrete blocks.

Which method is better?

The amount of work done in either method is equal

Method A: more blocks to stack but easier and faster to carry.

Method B: fewer blocks to stack but the process is slowed by the weight of each block.

The distinction is in the dispersal of the work done.

RISC versus CISC

RISC machines: SUN SPARC, SGI Mips, HP PA-RISC

CISC machines: Intel 80x86, Motorola 680x0

What really distinguishes RISC from CISC these days lies in the architecture and not in the instruction set.

CISC occurs whenever there is a disparity in speed

- between CPU operations and memory accesses
- due to technology or cost.

What about combining both ideas?

Intel 8086 Pentium P6 architecture
is externally CISC but internally RISC & CISC!

Intel IA-64 executes many instructions in parallel.

Pipelining (Designing…,M.J.Quinn, ‘87)

Instruction Pipelining is the use of pipelining to allow more than one instruction to be in some stage of execution at the same time.

Cache memory is a small, fast memory unit used as a buffer between a processor and primary memory

Ferranti ATLAS (1963):

- **Pipelining reduced the average time per instruction by 375%**
- **Memory could not keep up with the CPU, needed a cache.**

Pipelining versus Parallelism (Designing…,M.J.Quinn, ‘87)

Most high-performance computers exhibit a great deal of concurrency.

However, it is not desirable to call every modern computer a parallel computer.

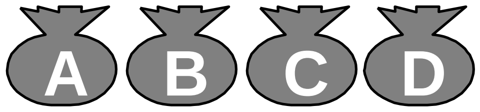
Pipelining and parallelism are 2 methods used to achieve concurrency.

Pipelining increases concurrency by dividing a computation into a number of steps.

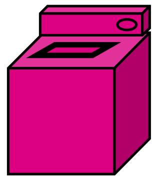
Parallelism is the use of multiple resources to increase concurrency.

Pipelining is Natural!

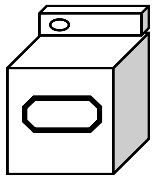
- Laundry Example
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold



- Washer takes 30 minutes

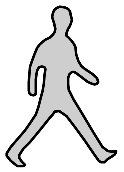


- Dryer takes 30 minutes

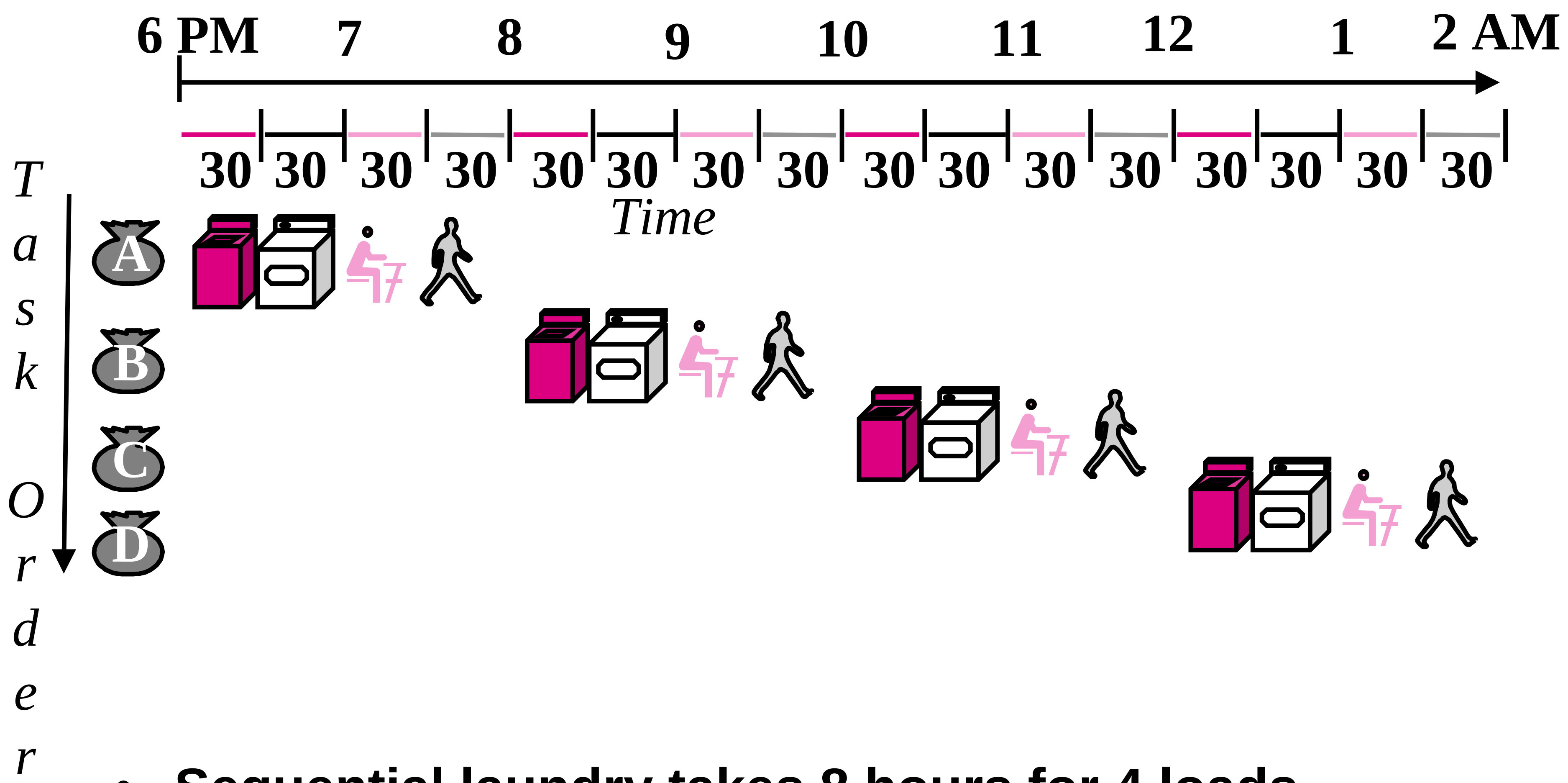


- "Folder" takes 30 minutes

- "Stasher" takes 30 minutes to put clothes into drawers

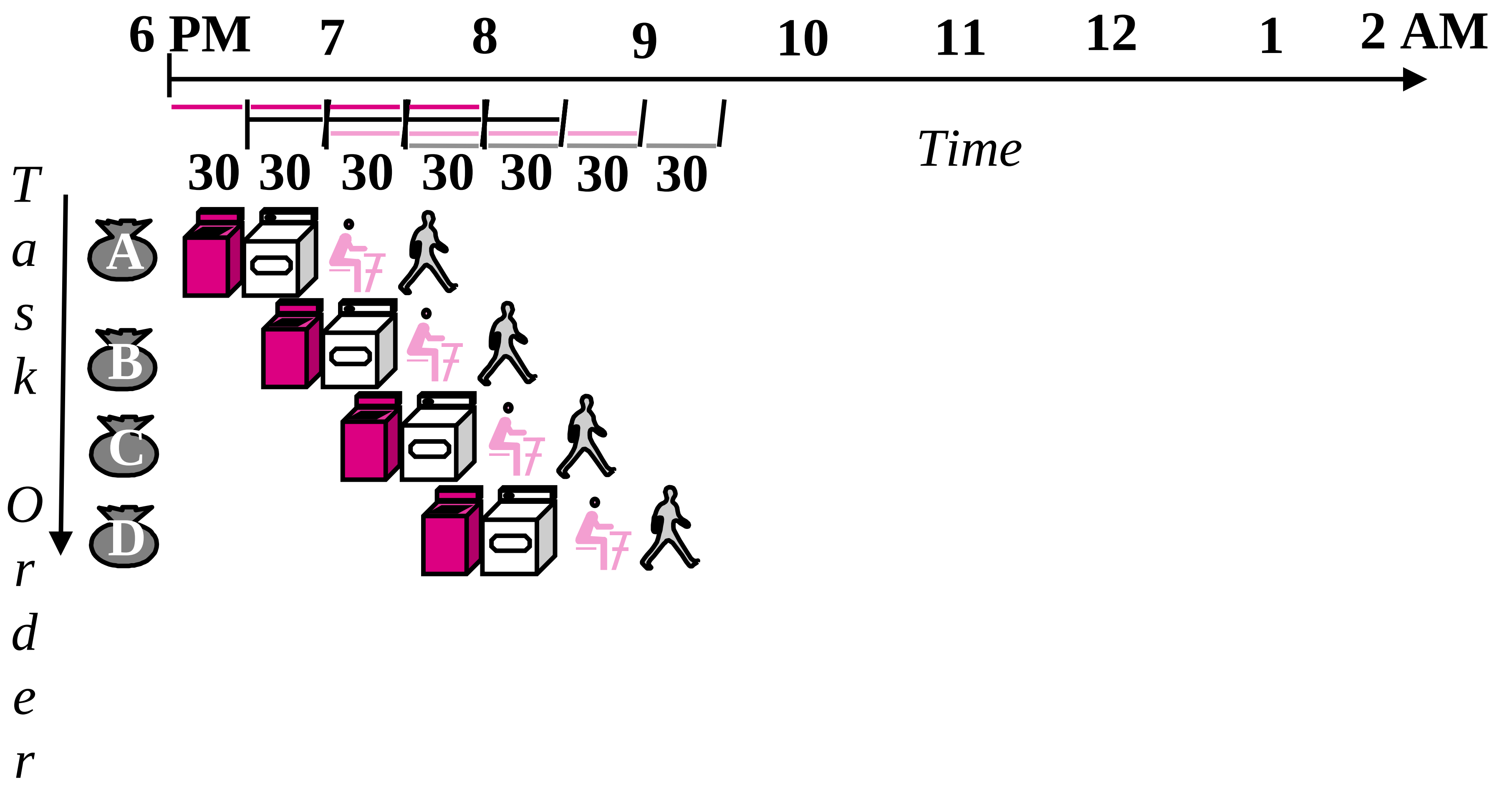


Sequential Laundry



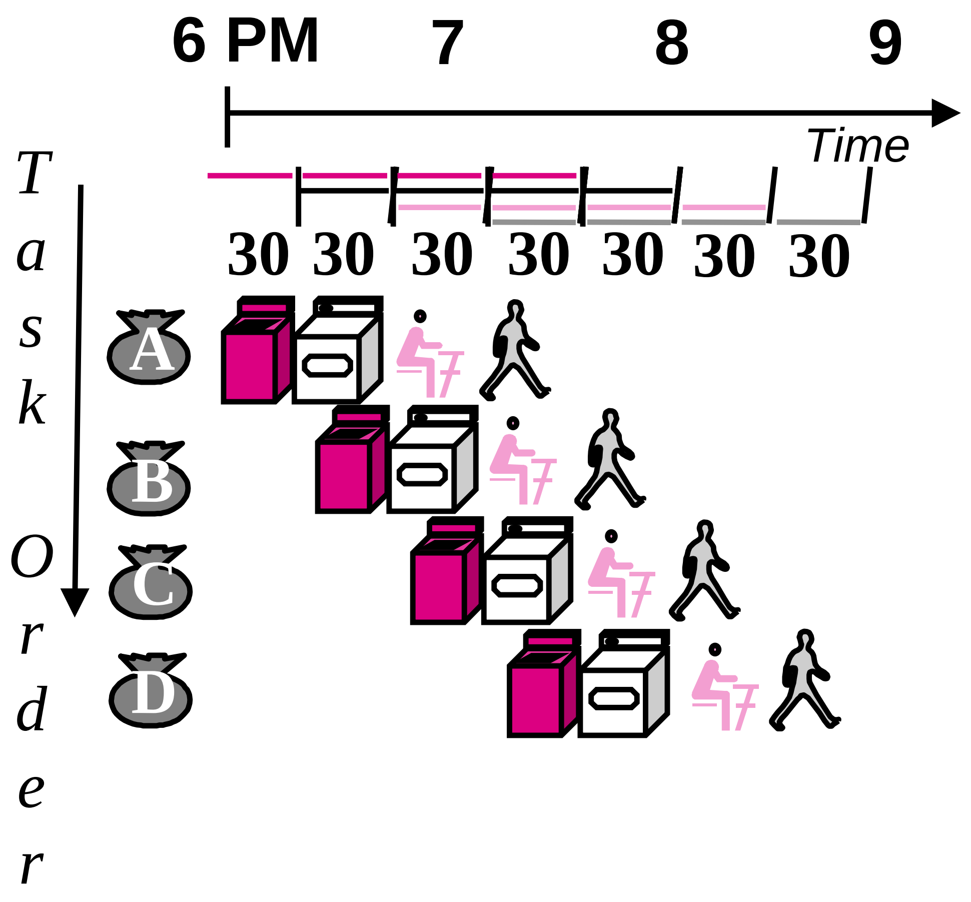
- **Sequential laundry takes 8 hours for 4 loads**
- **If they learned pipelining, how long would laundry take?**

Pipelined Laundry: Start work ASAP



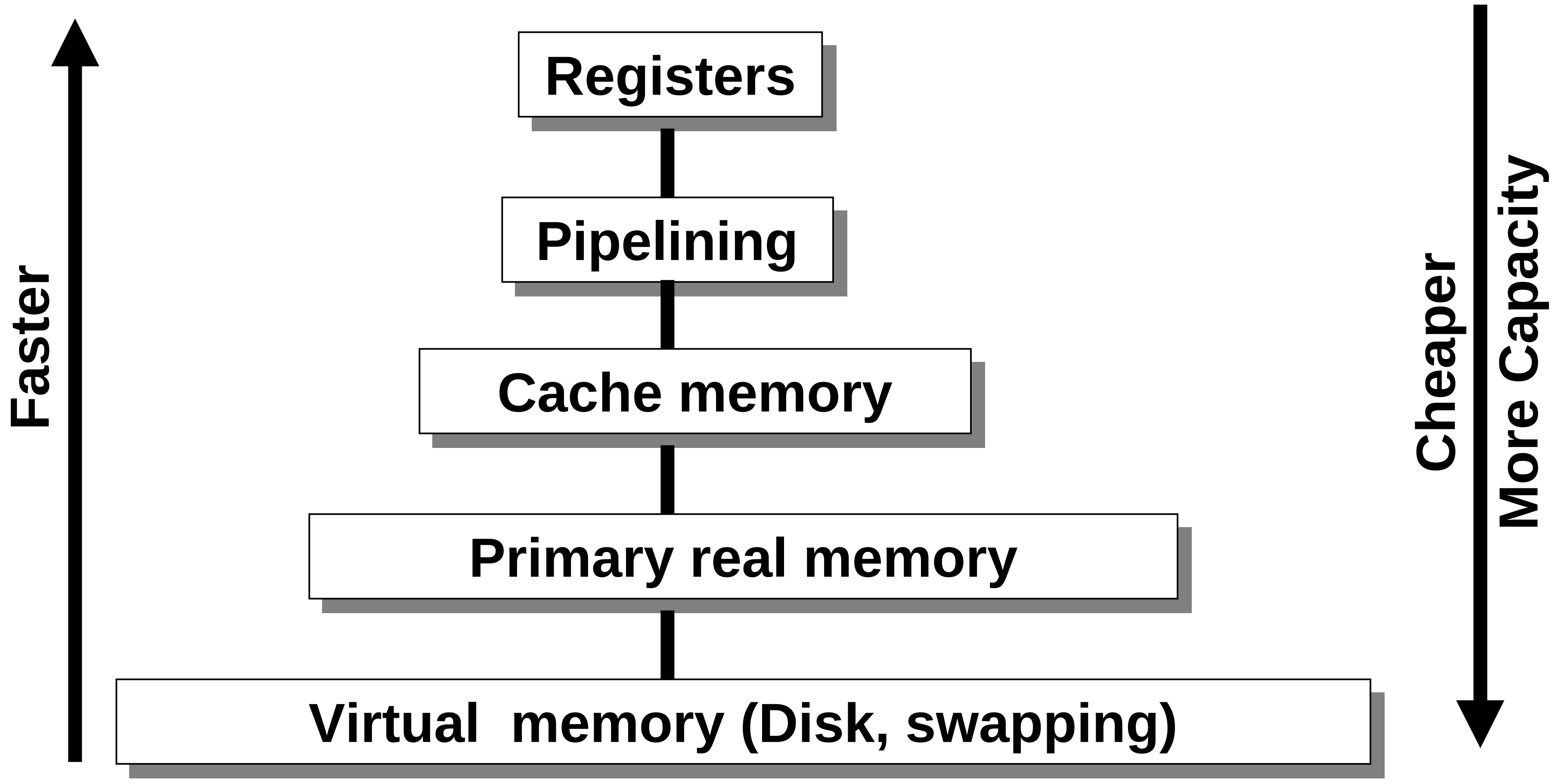
- **Pipelined laundry takes 3.5 hours for 4 loads!**

Pipelining Lessons



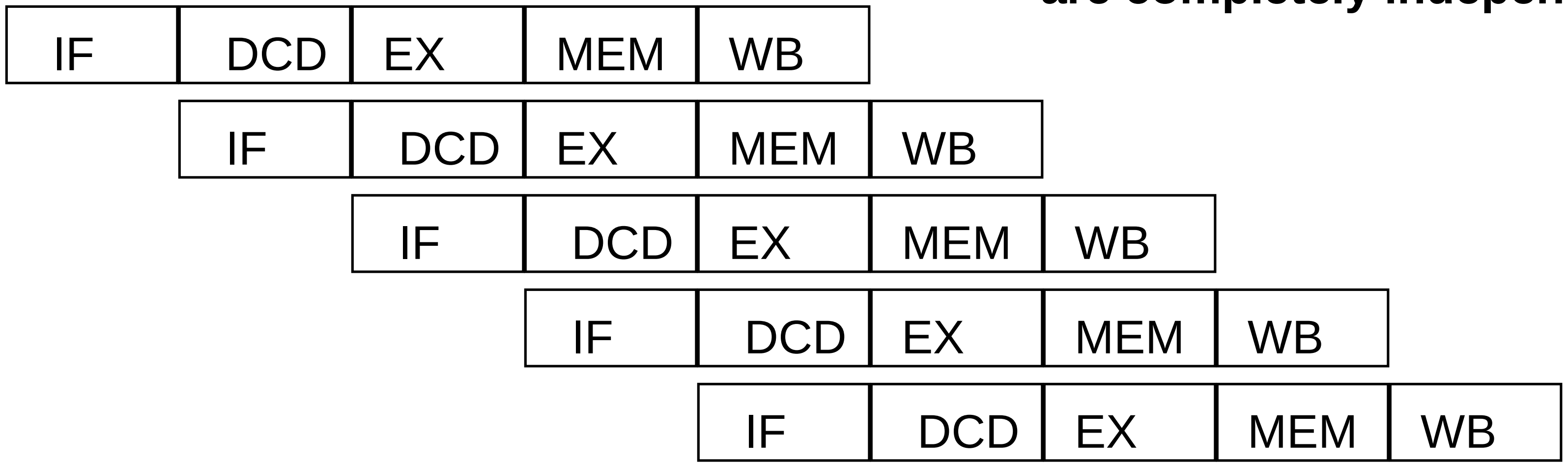
- Pipelining doesn't help **latency** of single task, it helps **throughput** of entire workload
- **Multiple** tasks operating simultaneously using different resources
- Potential speedup = **Number pipe stages**
- Pipeline rate limited by **slowest** pipeline stage
- Unbalanced lengths of pipe stages reduces speedup
- Time to "**fill**" pipeline and time to "**drain**" it reduces speedup
- Stall for Dependences

Memory Hierarchy



Ideal Pipelining

Assume instructions are completely independent!



Maximum Speedup ≤ Number of stages

$$\text{Speedup} \leq \frac{\text{Time for unpipelined operation}}{\text{Time for longest stage}}$$

More Operations Per Instruction

- How to increase the number of operations that can be performed in each instruction?

  - Add execution units (multiplier, adder, etc.)
    - Enhance the instruction set to take advantage of the additional hardware
    - Possibly, increase the instruction word width
    - Use wider buses to keep the processor fed with data

  - Add SIMD capabilities

More Instructions Per Clock Cycle

- How to increase the number of instructions that are issued and executed in every clock cycle?
  - Use VLIW techniques
  - Use superscalar techniques

- VLIW and superscalar architectures typically use simple, RISC-based instructions rather than the complex, compound instructions traditionally used in DSP processors

New Architectures for DSP

- Enhanced conventional DSPs
  - Examples: Lucent DSP16xxx, ADI ADSP-2116x
- VLIW (Very Long Instruction Word) DSPs
  - Examples: TI TMS320C6xxx, Siemens Carmel
- Superscalar DSPs
  - Example: ZSP ZSP164xx
- General-purpose processors, hybrids:
  - Examples: PowerPC with AltiVec, TriCore

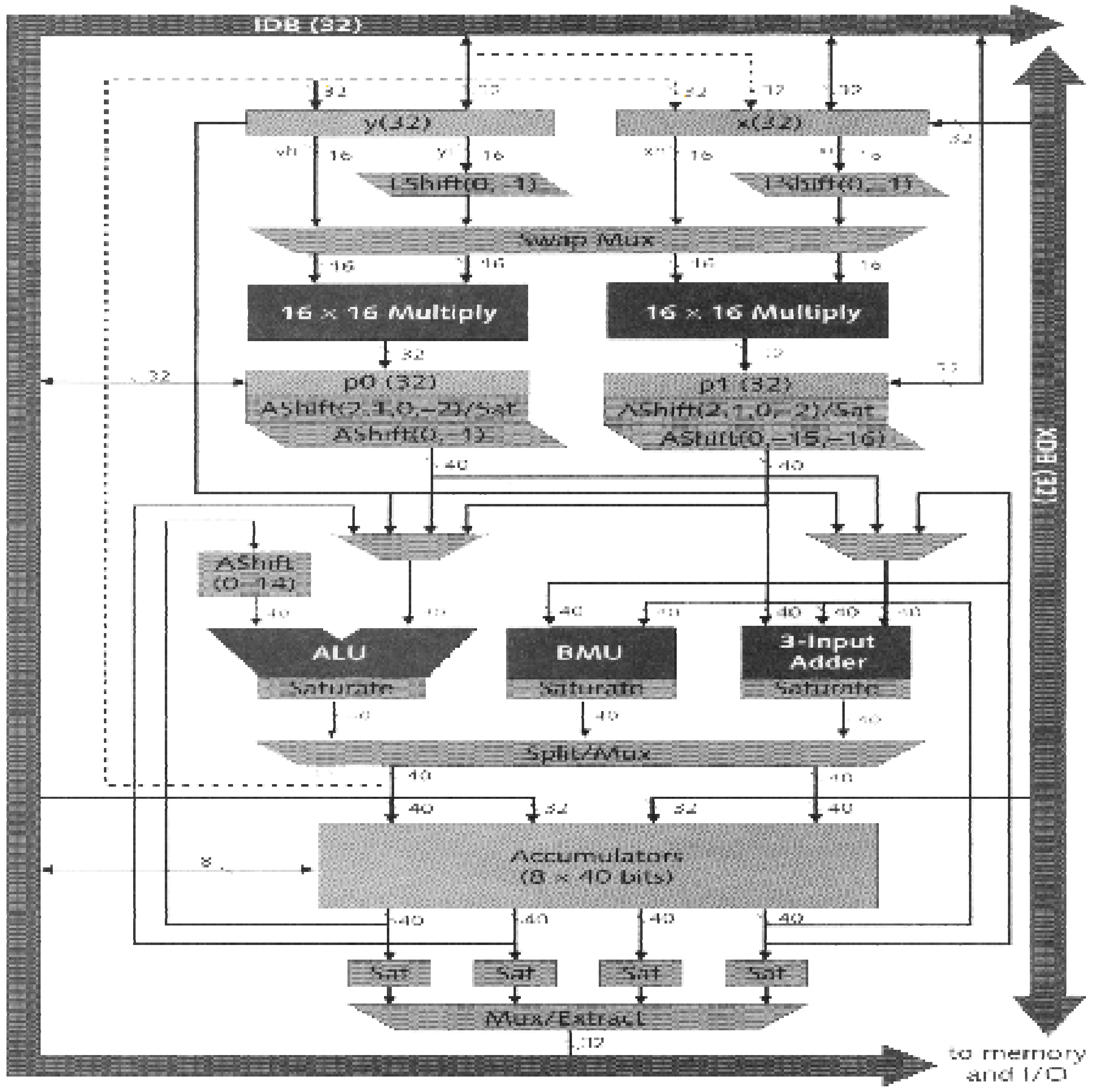
Enhanced Conventional DSPs

More parallelism via:

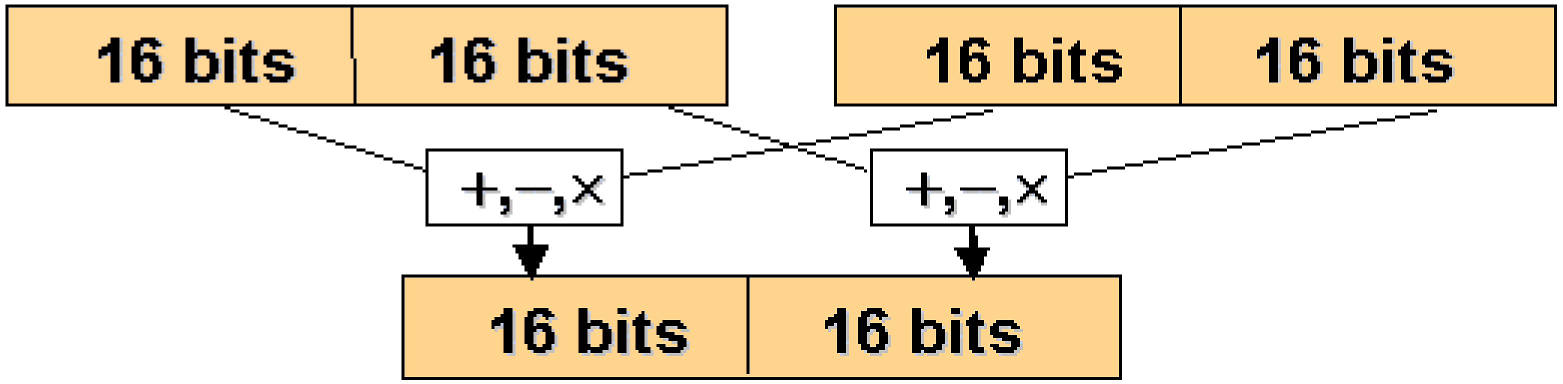
- Multi-operation data path
  - e.g., 2nd multiplier, adder
  - SIMD capabilities (ranging from limited to extensive)
- Highly specialized hardware in core
  - e.g., application-oriented data path operations
- Co-processors
  - Viterbi decoding, FIR filtering, etc.

Example: Lucent DSP16xxx, ADI ADSP-2116x

DSP16000 Data Path



SIMD



- Splits words into smaller chunks for parallel operations
- Some SIMD processors support multiple data widths (16-bit, 8-bit, ..)
- Examples: Lucent DSP16xxx, ADI ADSP-2116x

SIMD Extensions

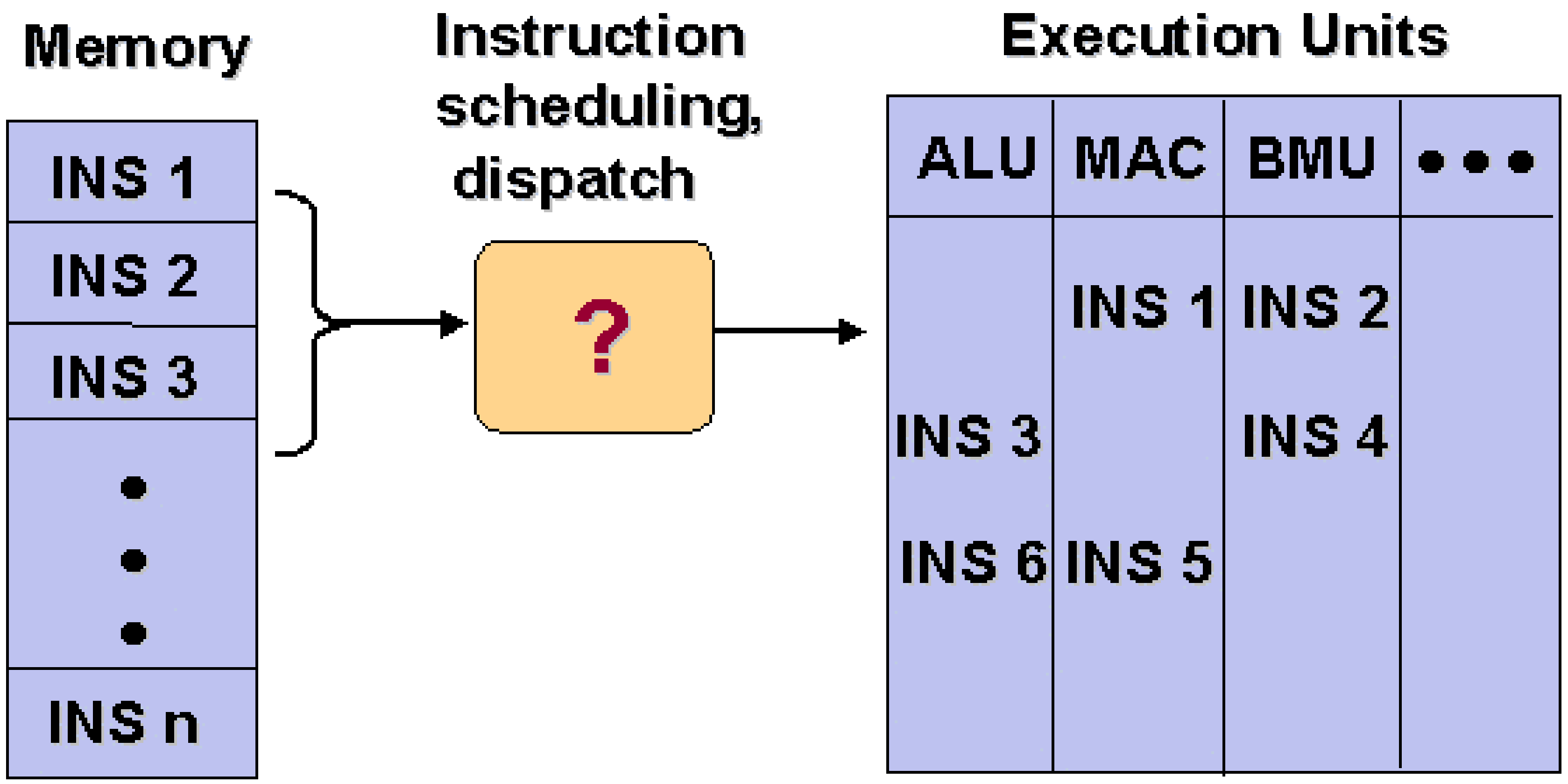
- SIMD is becoming more and more common in DSP processors
  - Limited SIMD capabilities on the DSP16xxx
  - Full SIMD capabilities (enabled by dual data paths) on ADI's ADSP-2116x
- SIMD extensions for CPUs are also common. Why?
  - Make good use of existing wide resources
    - Buses, data path
  - Significantly accelerate many DSP/image/video algorithms without a radical architectural change

SIMD Challenges

- Algorithms, data organization must be amenable to data-parallel processing
  - Programmers must be creative, and sometimes pursue alternative algorithms
  - Reorganization penalties can be significant
  - Most effective on algorithms that process large blocks of data

BDTi

Superscalar vs VLIW



VLIW (Very Long Instruction Word)

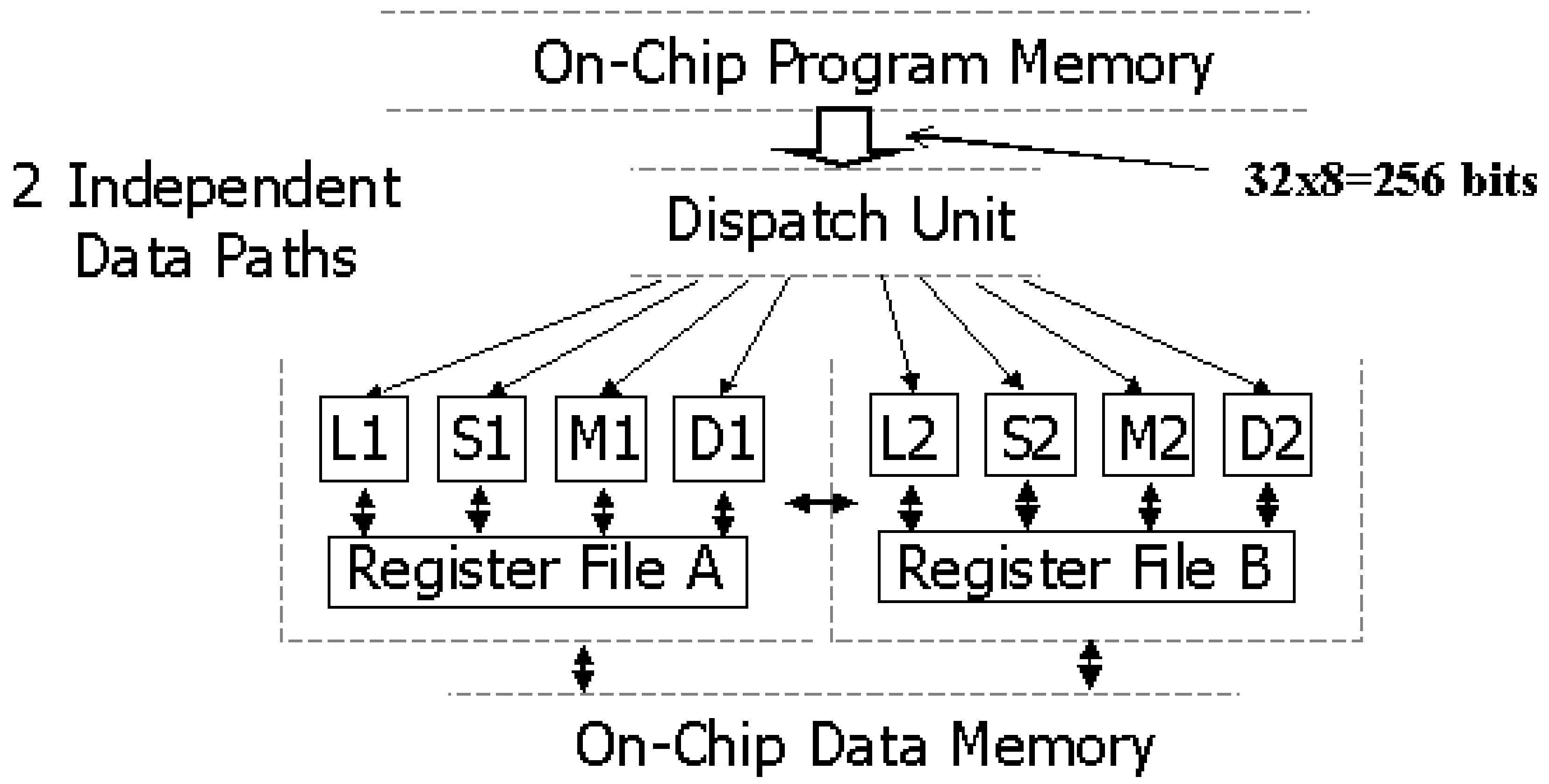
Examples of current & upcoming VLIW-based architectures for DSP applications:

- TI TMS320C6xxx, Siemens Carmel, ADI TigerSHARC, StarCore 140

Characteristics:

- Multiple independent operations per cycle, packed into single large "instruction" or "packet"
- More regular, orthogonal, RISC-like operations
- Large, uniform register sets

Example VLIW Data Path ('C6x)



FIR Filtering on the 'C6x

```
LOOP:
   ADD    .L1 A0,A3,A0
||ADD    .L2 B1,B7,B1
||MPYHL .M1X A2,B2,A3
||MPYLH .M2X A2,B2,B7
||LDW    .D2 *B4++,B2
||LDW    .D1 *A7--,A2
||[B0] ADD .S2 -1,B0,B0
||[B0] B .S1 LOOP
; LOOP ends here
```

VLIW Architectures

- Advantages:
  - Increased performance
  - More regular architectures
    - Potentially easier to program, better compiler targets
  - Scalable (?)

VLIW Architectures

- Disadvantages:
  - New kinds of programmer/compiler complexity
    - Programmer (or code-generation tool) must keep track of instruction scheduling
    - Deep pipelines and long latencies can be confusing, may make peak performance elusive
  - Code size bloat
    - High program memory bandwidth requirements
  - High power consumption

Superscalar Architectures

Current superscalar architectures for DSP apps:

- ZSP ZSP164xx, Siemens TriCore (DSP/μC hybrid)

Characteristics:

- Borrow techniques from high-end CPUs
- Multiple (usually 2-4) instructions issued per instruction cycle
  - Instruction scheduling handled in hardware, not by programmer
- RISC-like instruction set
- Lots of parallelism

FIR Filtering on the ZSP164xx

```
LOOP: LDDU        R4, R14, 2
      LDDU        R8, R15, 2
      MAC2.A      R4, R8
      AGN0        LOOP
```

(All four instructions execute in a single cycle)

Superscalar Architectures

- Advantages:
  - Large jump in performance
  - More regular architectures (potentially easier to program, better compiler targets)
  - Programmer (or code generation tool) isn't required to schedule instructions
    - But peak performance may be hard to achieve without hand-scheduling
  - Code size not increased significantly

Superscalar Architectures

- Disadvantages:
  - Energy consumption is a major challenge
  - Dynamic behavior complicates software development
    - Execution-time variability can be a hazard
    - Code optimization is challenging

Hybrid DSP/Microcontrollers

- GPPs for embedded applications are starting to address DSP needs
- Embedded GPPs typically don't have the advanced features that affect execution time predictability, so are easier to use for DSP

Hybrid DSP/Microcontrollers

Approaches

- Multiple processors on a die
  - e.g., Motorola DSP5665x
- DSP co-processor
  - e.g., ARM Piccolo
- DSP brain transplant in existing μC
  - e.g., SH-DSP
- Microcontroller tweaks to existing DSP
  - e.g., TMS320C27xx
- Totally new design
  - e.g., TriCore

Hybrid DSP/Microcontrollers

Advantages, Disadvantages

- Multiple processors on a die
  - Two entirely different instruction sets, debugging tools, etc.
  - Both cores can operate in parallel
  - No resource contention...
  - ..but probably resource duplication

BDTi

Hybrid DSP/Microcontrollers
Advantages, Disadvantages

- DSP brain transplant in existing μC, microcontroller tweaks to existing DSP
  - Simpler programming model than dual cores
  - Constraints imposed by "legacy" architecture

- Totally new design
  - Avoids legacy constraints
  - May result in a cleaner architecture
  - Adopting a totally new architecture can be risky

Conclusions

- The variety, performance range of processors for DSP is exploding
  - Better selection, flexibility,...
  - ...but harder to choose the "best" processor
- DSPs, microcontrollers, and CPUs are swapping architectural tricks
  - CPU, µC vendors recognize the need for DSP capabilities
  - DSP, µC vendors don't want to lose sockets to each other
  - What is good in a CPU may not be good in a DSP; be careful of issues such as execution-time predictability, programmability, etc.

BDTi

For More Information...

Free resources on BDTI's web site,

http://www.bdti.com

- *DSP Processors Hit the Mainstream* covers DSP architectural basics and new developments. Originally printed in IEEE Computer Magazine.
- *Evaluating DSP Processor Performance,* a white paper from BDTI.
- Numerous other BDTI article reprints, slides
- *comp.dsp* FAQ