EECS 318 CAD
Computer Aided Design

LECTURE 2:
DSP Architectures

Instructor: Francis G. Wolff
wolff@eecs.cwru.edu
Case Western Reserve University

This presentation uses powerpoint animation: please viewshow

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.

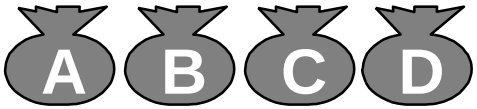
Pipelining and parallelism are 2 methods used to achieve concurrency.

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

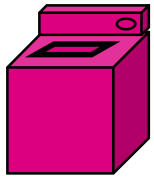
Parallelism is the use of multiple resources to increase concurrency.

Pipelining is Natural!

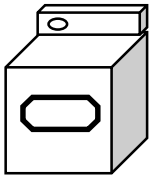
- Laundry Example
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold



- Washer takes 30 minutes



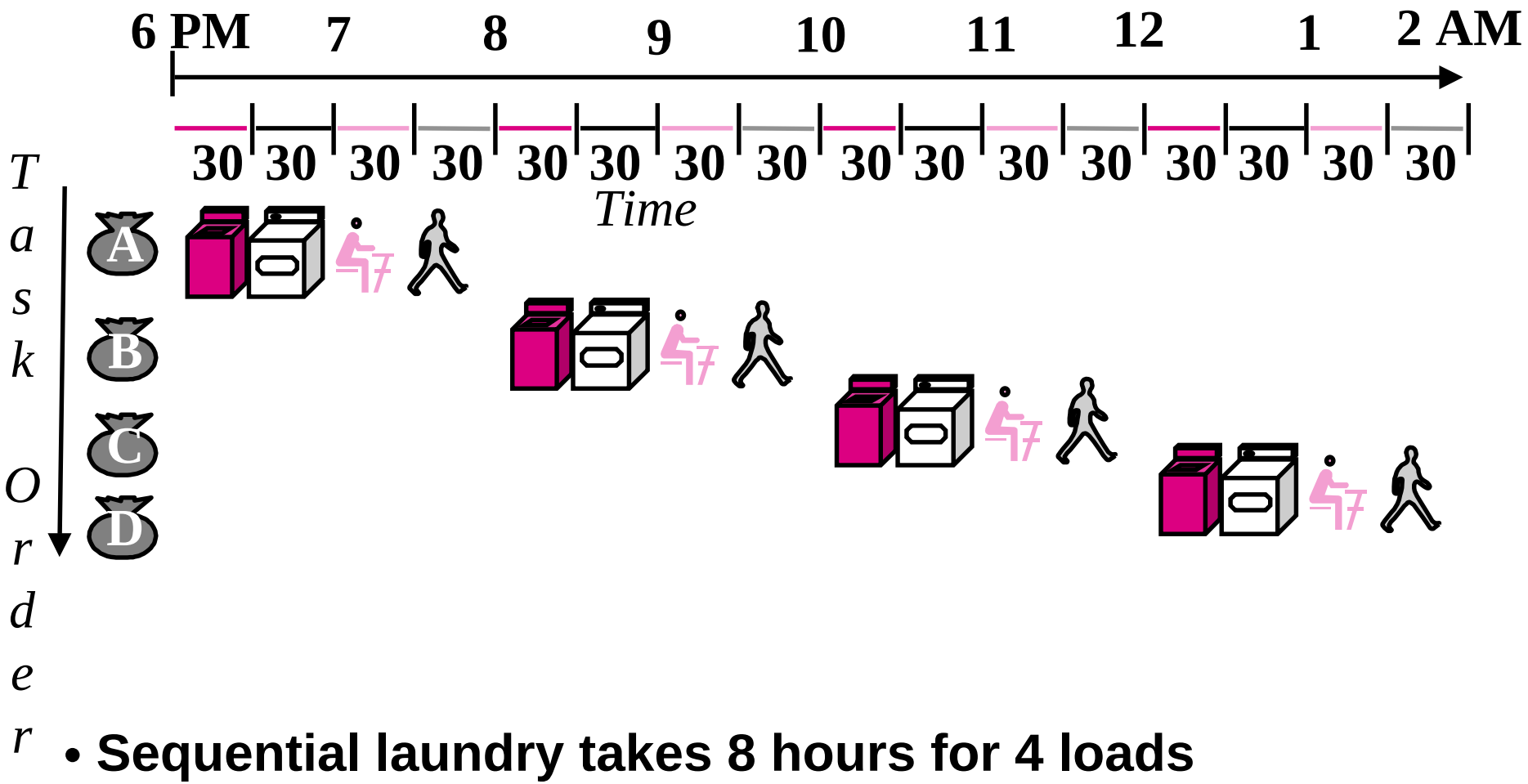
- Dryer takes 30 minutes



- "Folder" takes 30 minutes

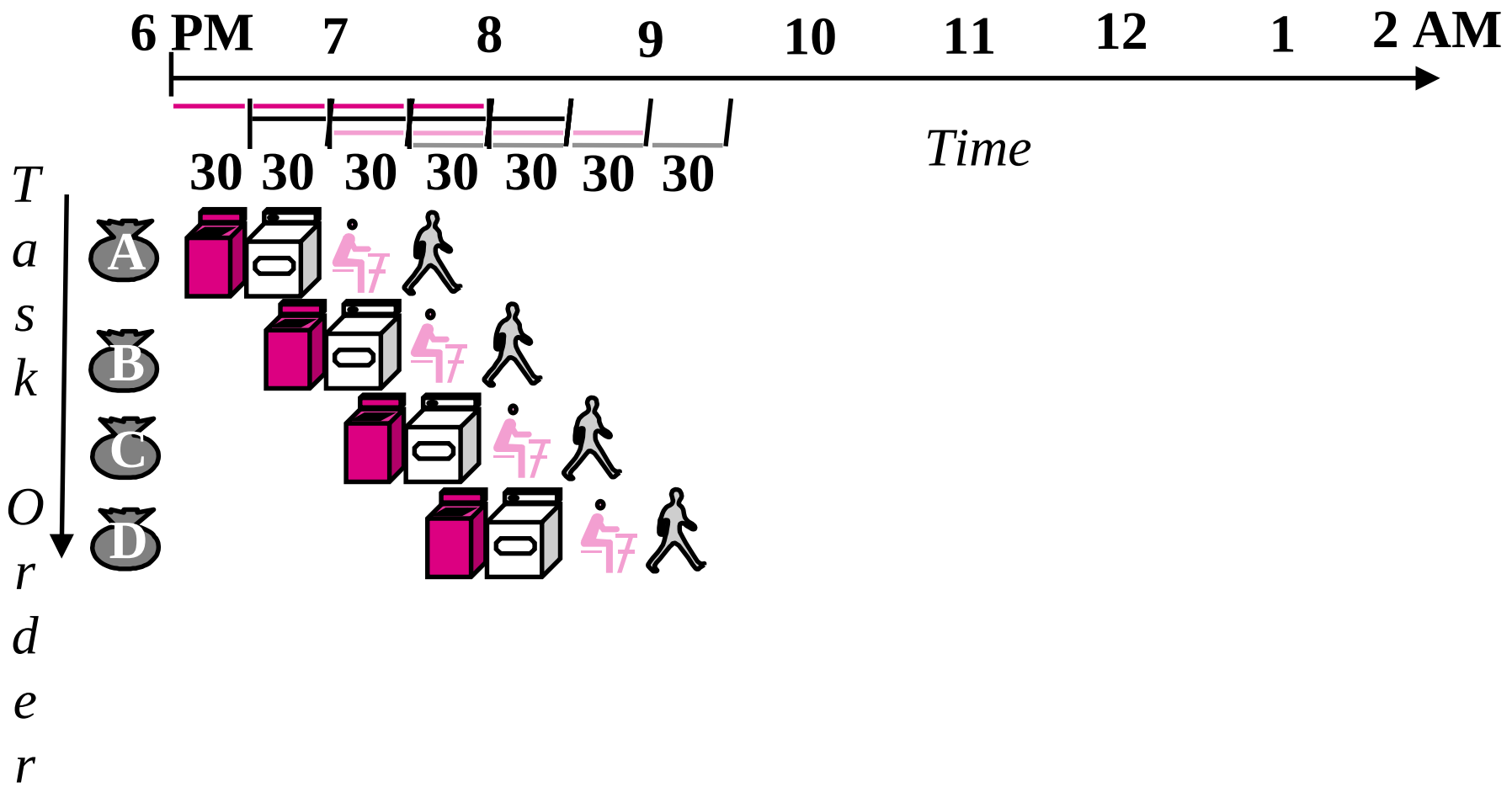
- "Stasher" takes 30 minutes to put clothes into drawers

Sequential Laundry



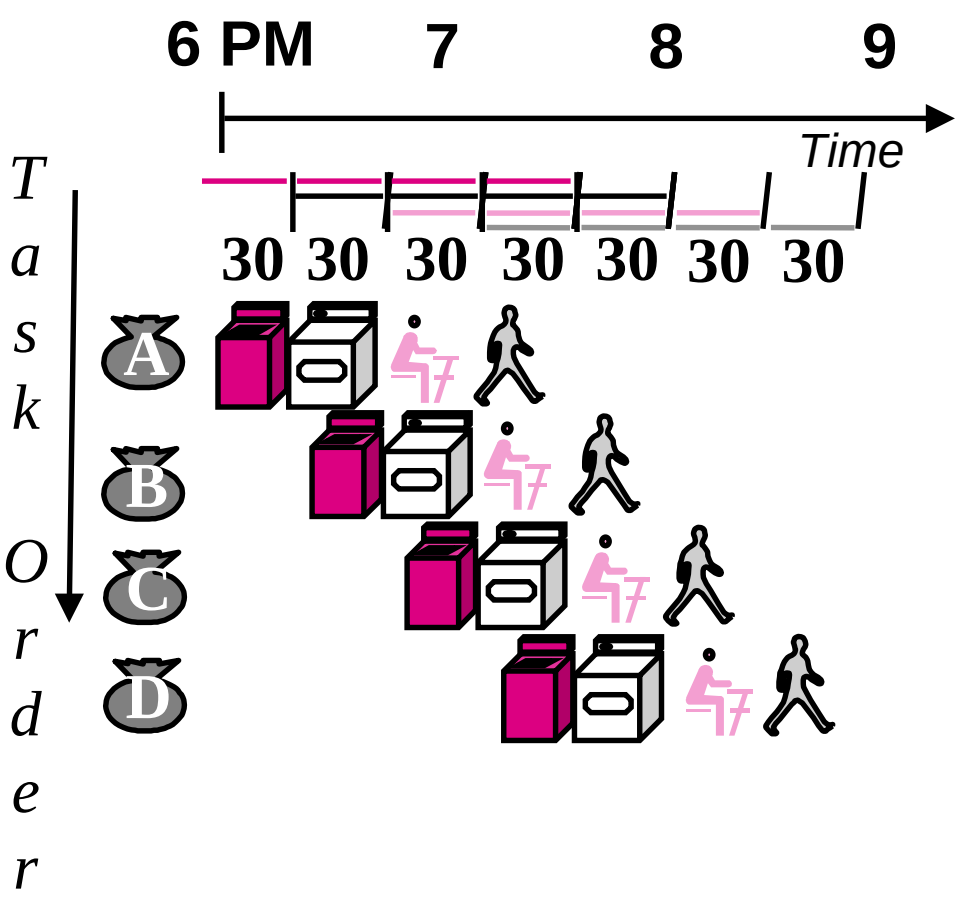
- **Sequential laundry takes 8 hours for 4 loads**
- **If they learned pipelining, how long would laundry take?**

Pipelined Laundry: Start work ASAP



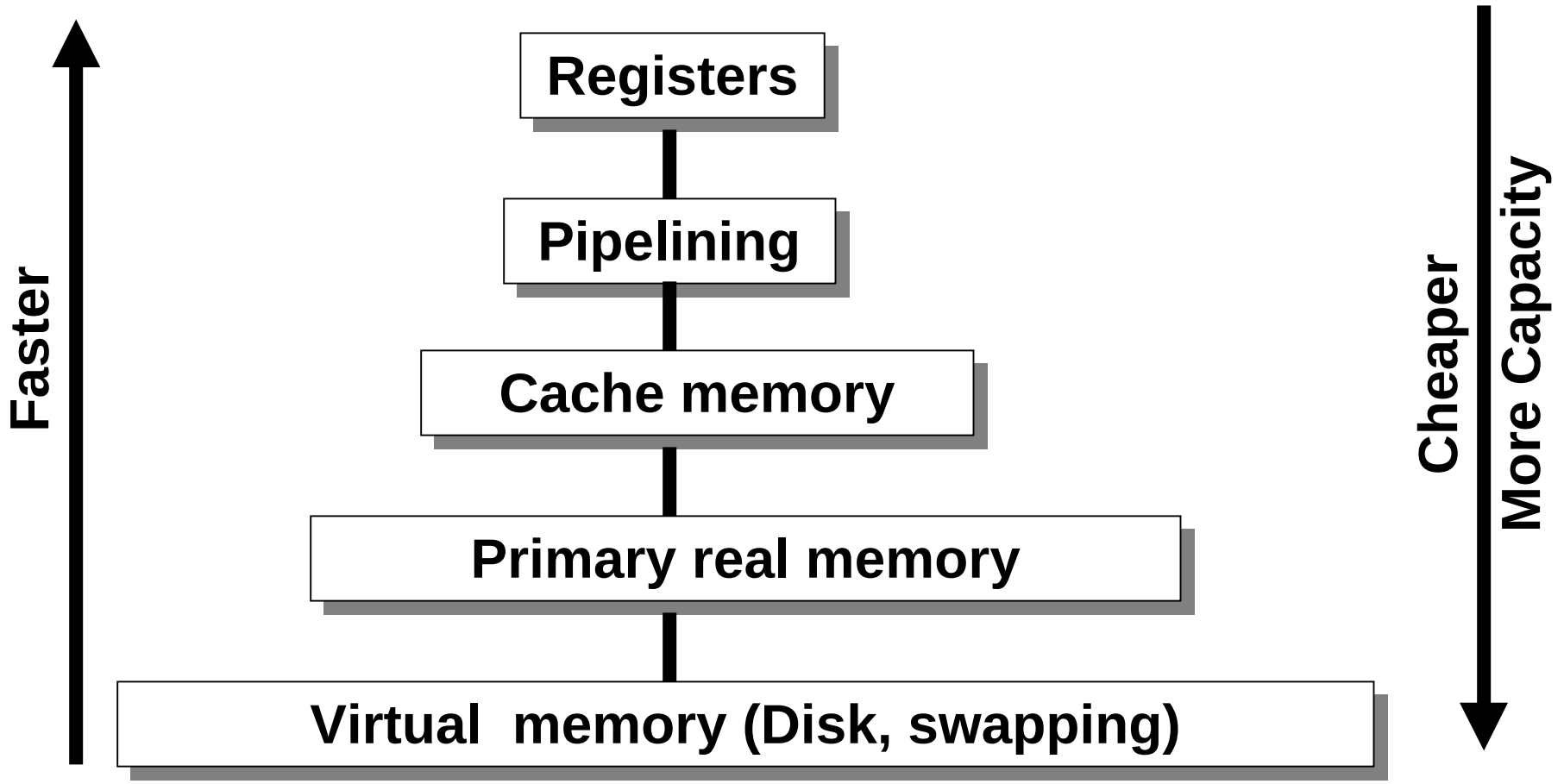
Pipelined laundry takes 3.5 hours for 4 loads!

Pipelining Lessons



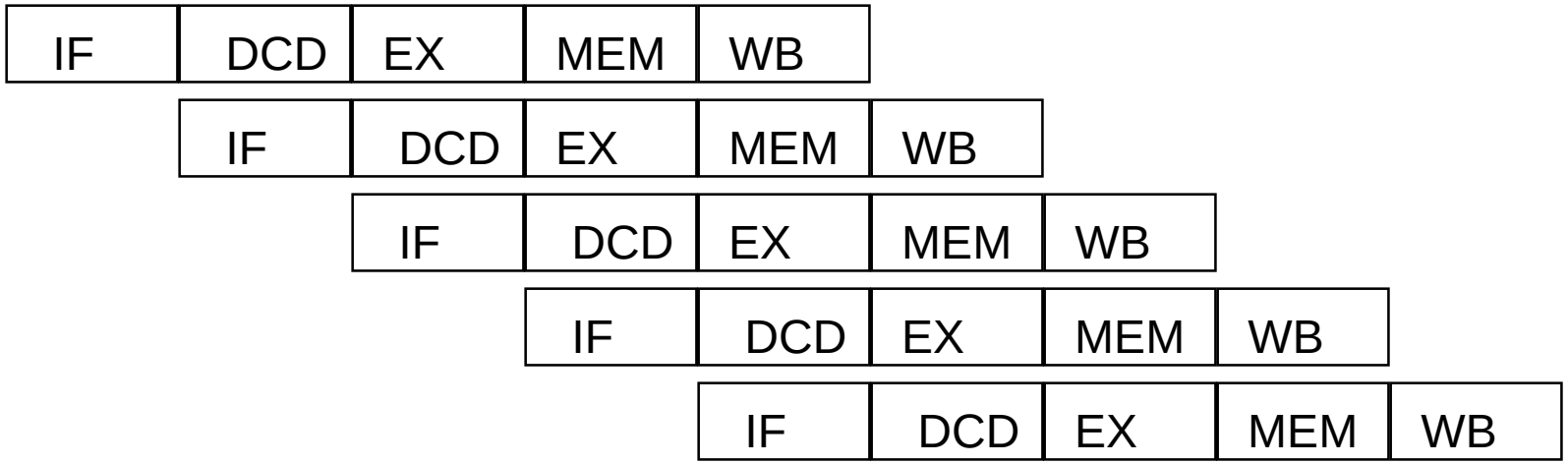
- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages reduces speedup
- 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

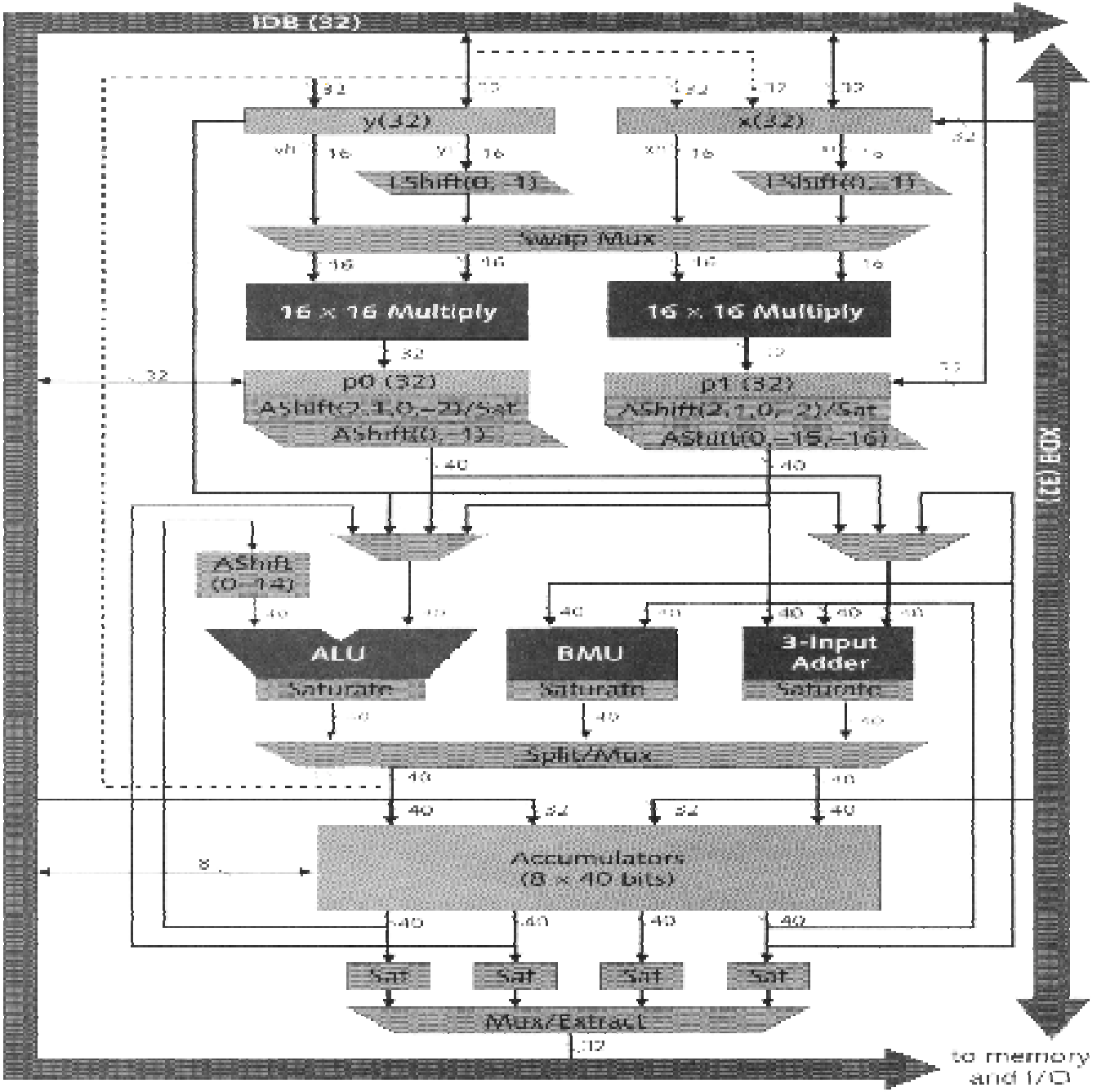
Enhanced Conventional DSPs

More parallelism via:

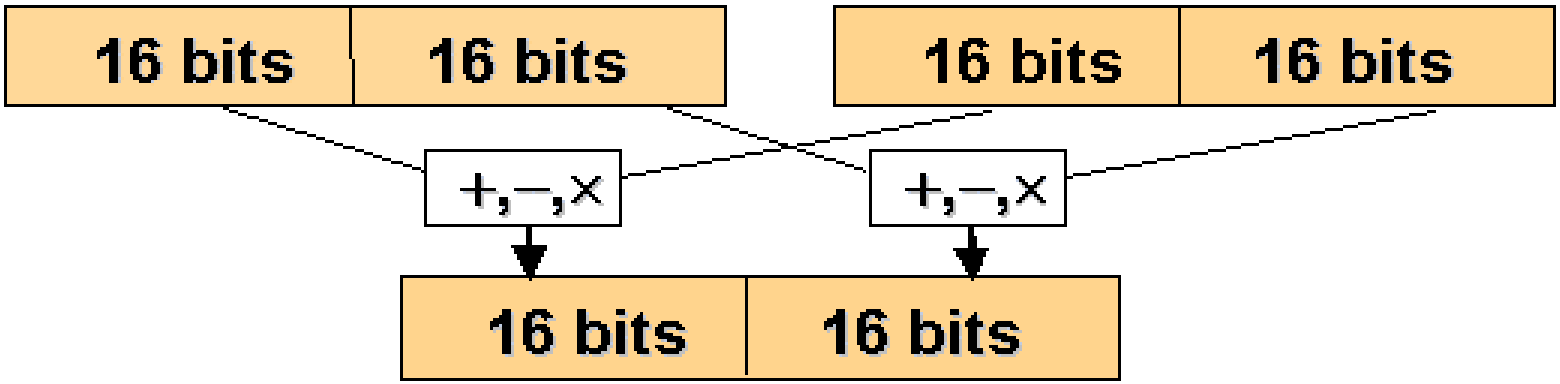
- Multi-operation data path
  - e.g., 2nd multiplier, adder
  - SIMD capabilities (ranging from limited to extensive)
- Highly specialized hardware in core
  - e.g., application-oriented data path operations
- Co-processors
  - Viterbi decoding, FIR filtering, etc.

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

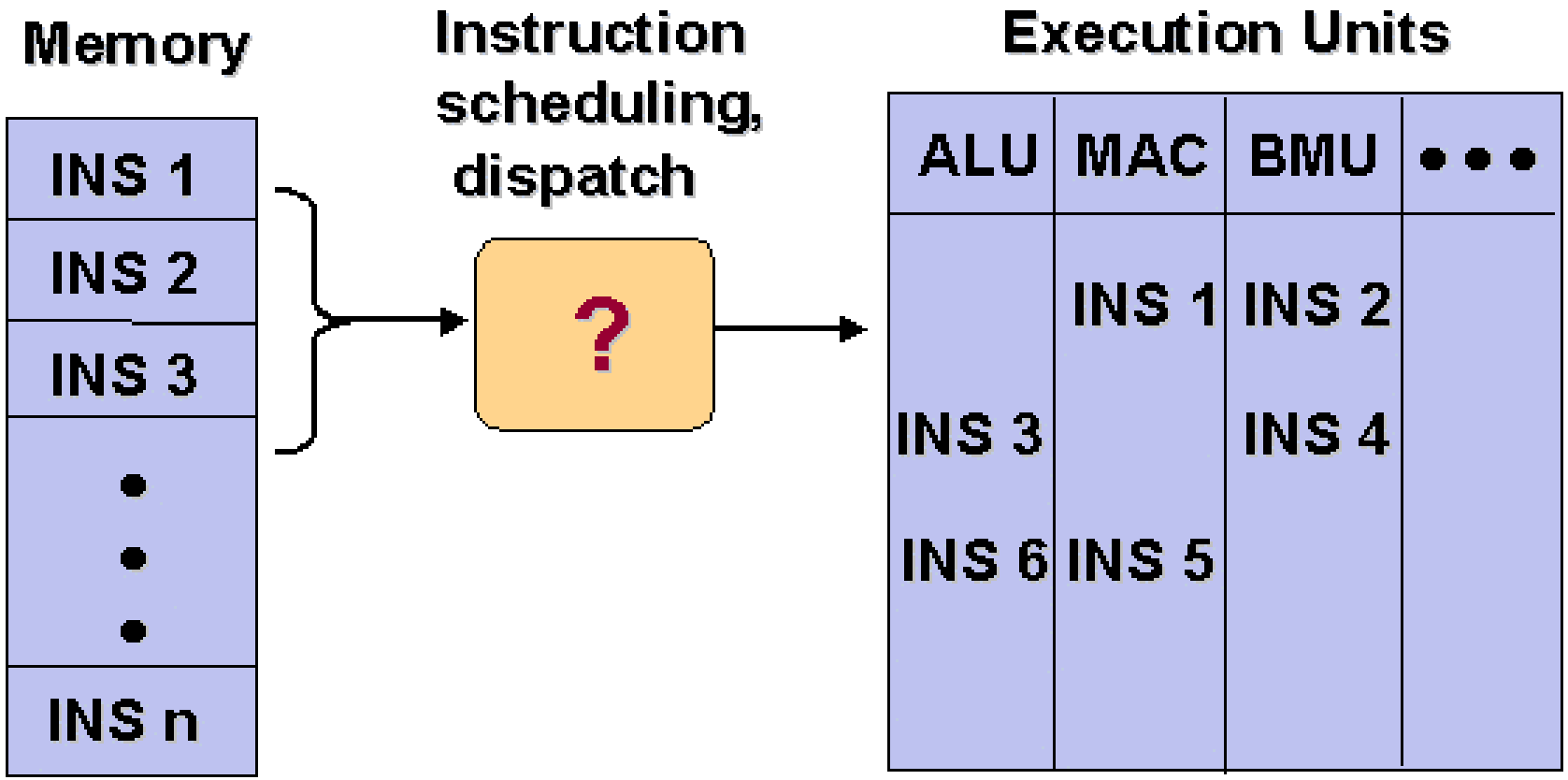
SIMD Extensions

- 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

Superscalar vs VLIW



VLIW (Very Long Instruction Word)

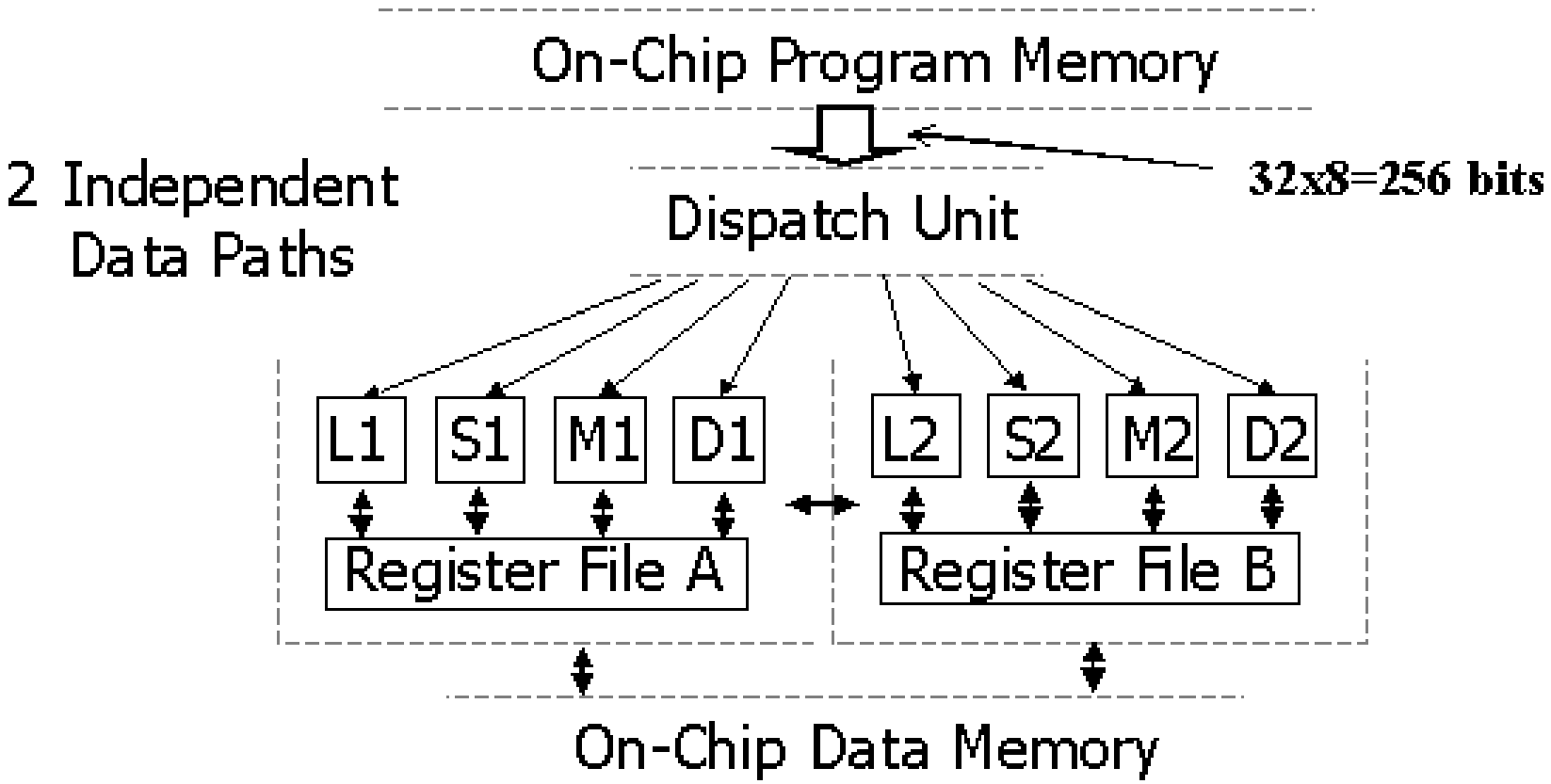
Examples of current & upcoming VLIW-based architectures for DSP applications:

- 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

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.

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