# High Performance Computing

Lecture 5: Hardware Issues

Nayda G. Santiago

August 18, 2008

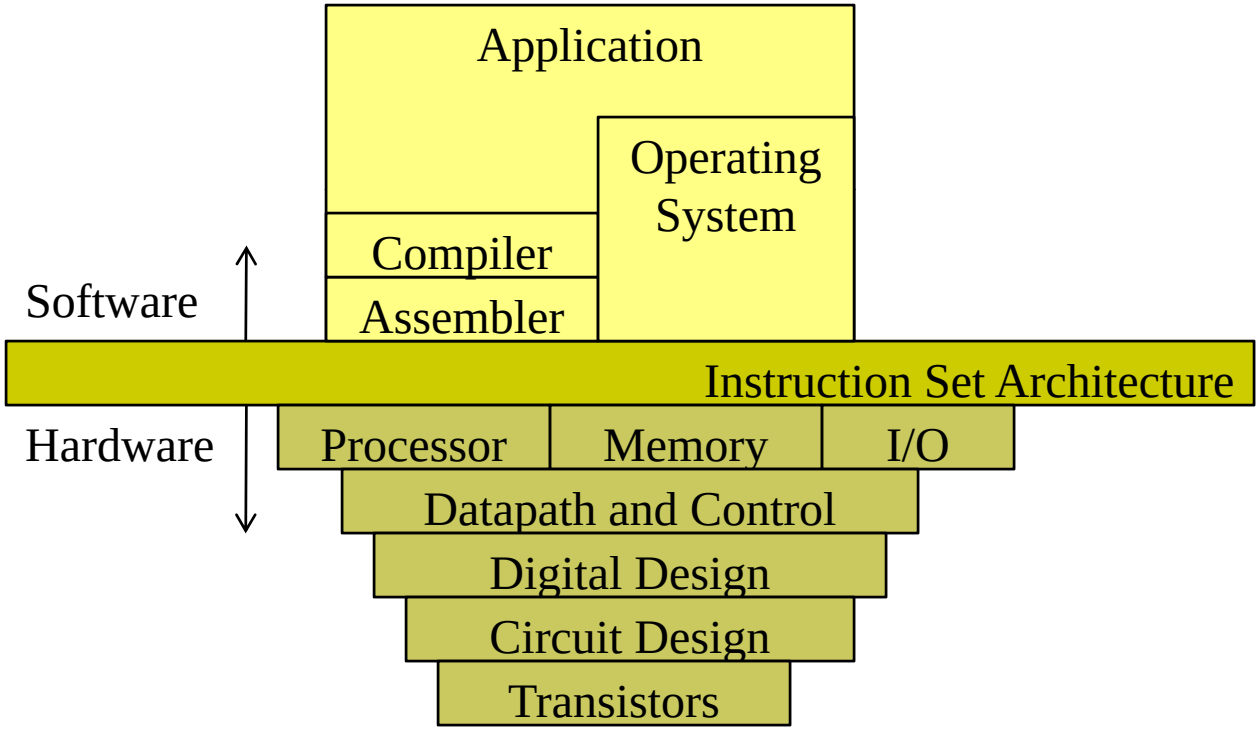
# Administrative Details

- Make up class for  Monday Aug 11, 2008 class
  - Thursday, Aug 21, 2008, 5:30pm to 6:30pm, S203
- We need to schedule additional make up classes – TODAY!!!
  - Traveling from Sept 30 to Oct 4, Colorado – 2 lectures
  - Traveling from Oct 9 to Oct 12, Houston – 1 lecture
  - Traveling from Oct 20 to Oct 22, Ann Arbor – 2 lectures
  - Traveling from Oct 27 to Oct 29, Boston – 2 lectures

# Units of High Performance Computing

| Abbreviation | Name | Represents |
|---|---|---|
| 1 Mflop/s | 1 Megaflop/s | $10^6$ Flop/sec |
| 1 Gflop/s | 1 Gigaflop/s | $10^9$ Flop/sec |
| 1 Tflop/s | 1 Teraflop/s | $10^{12}$ Flop/sec |
| 1 Pflop/s | 1 Petaflop/s | $10^{15}$ Flop/sec |
| 1 MB | 1 Megabyte | $10^6$ Bytes |
| 1 GB | 1 Gigabyte | $10^9$ Bytes |
| 1 TB | 1 Terabyte | $10^{12}$ Bytes |
| 1 PB | 1 Petabyte | $10^{15}$ Bytes |

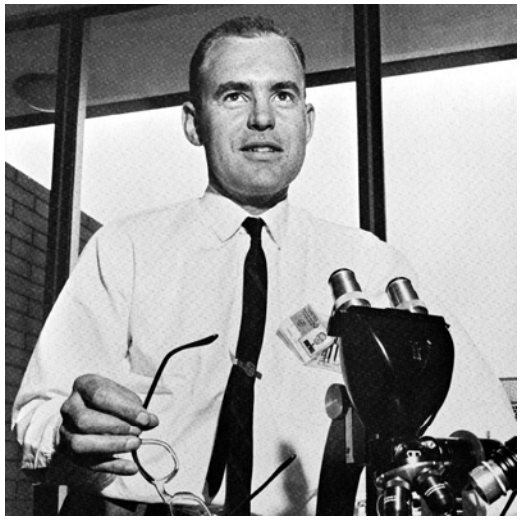
# Computer System




Taken from David Petterson, Intro to Hw/Sw talk

# Moore's Law

- Gordon Moore
  - Co-founder of Intel
  - Electronics Magazine
  - April 1965
  - Number of devices/chip doubles every 18 months
  - The trend has continued for more than half a century and is not expected to stop for another decade at least.



# Moore's Law

- Something doubles every 18 – 24 months
  - Originally transistors
  - Performance
- Moore's law is an exponential
- Has hold true for approx 30 years
- Empirical

# In today's processors

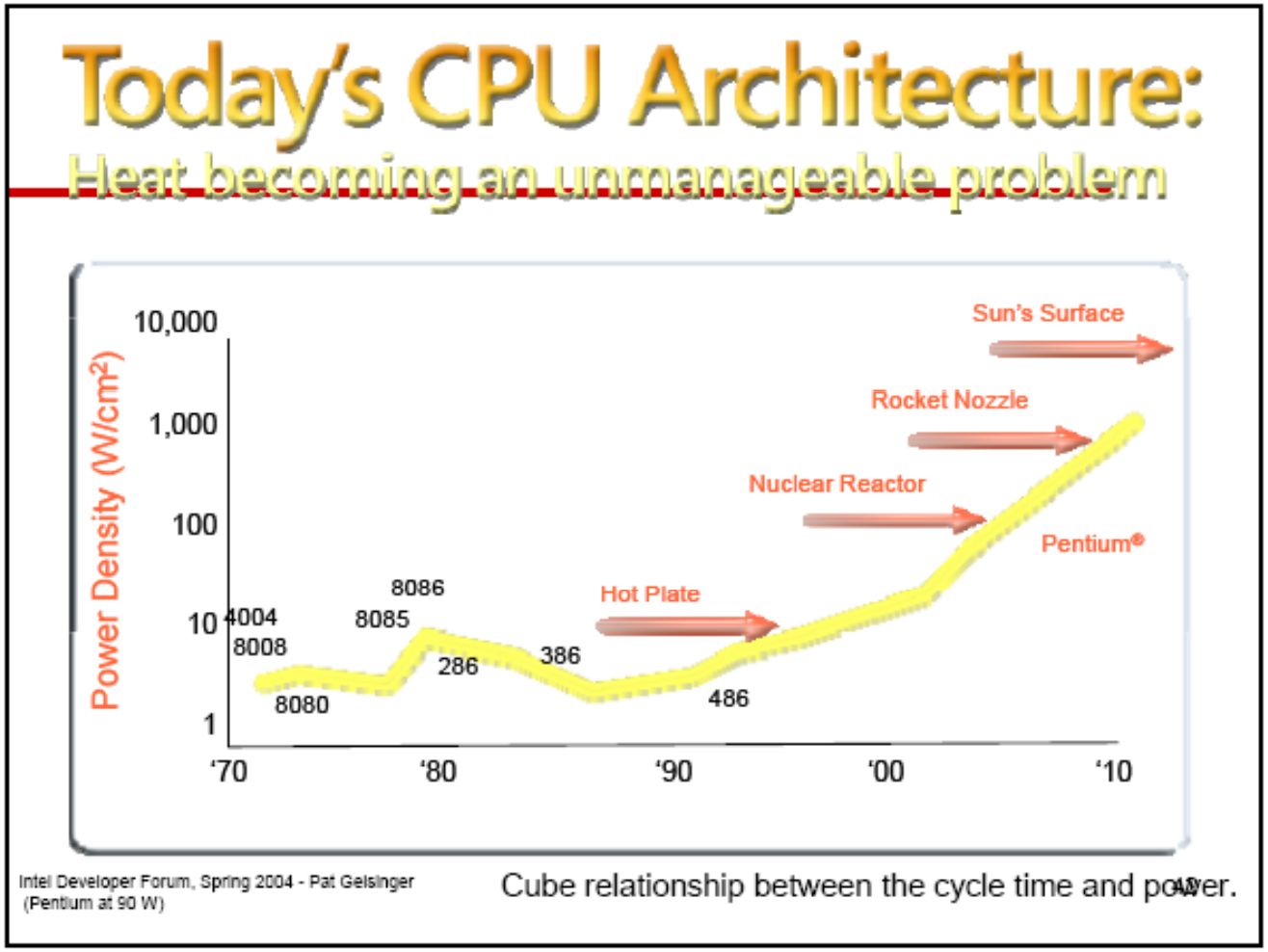
- Increase the number of gates and decrease the cycle time of the processor
  - Increase transistor density and clock rate
    - HEAT becomes an unmanageable problem!!!
    - Power is proportional to
      - Frequency
      - Voltage level

- Some Intel Processors: power consumption 100 W!!!

# Intel Itanium 2

| Clock Speed | Power |
|---|---|
| 1.66 GHz | 122W |
| 1.6 GHz | 122 W |
| 1.5 GHz | 107W |

# Heat – a problem



In order to hold the trend

# CLEVER DESIGN TO IMPROVE PERFORMANCE

# So how to hold the trend

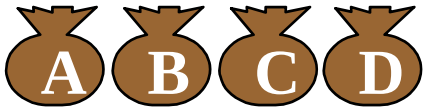
- Clever design techniques
  - Pipelining
    - Prediction
  - Out of order execution
  - Superscalar
    - Multiple functional units
  - Memory
    - Hierarchical memory design
    - Multilevel caches

# So how to hold the trend

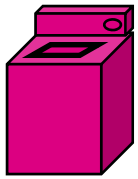
- Clever design techniques
  - Pipelining
    - Prediction
  - Out of order execution
  - Superscalar
    - Multiple functional units
  - Memory
    - Hierarchical memory design
    - Multilevel caches

# Remember the Pipelining Example?

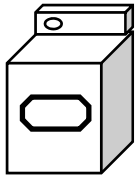
- **Laundry Example**
- **Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, fold, and put away**




- **Washer takes 30 minutes**



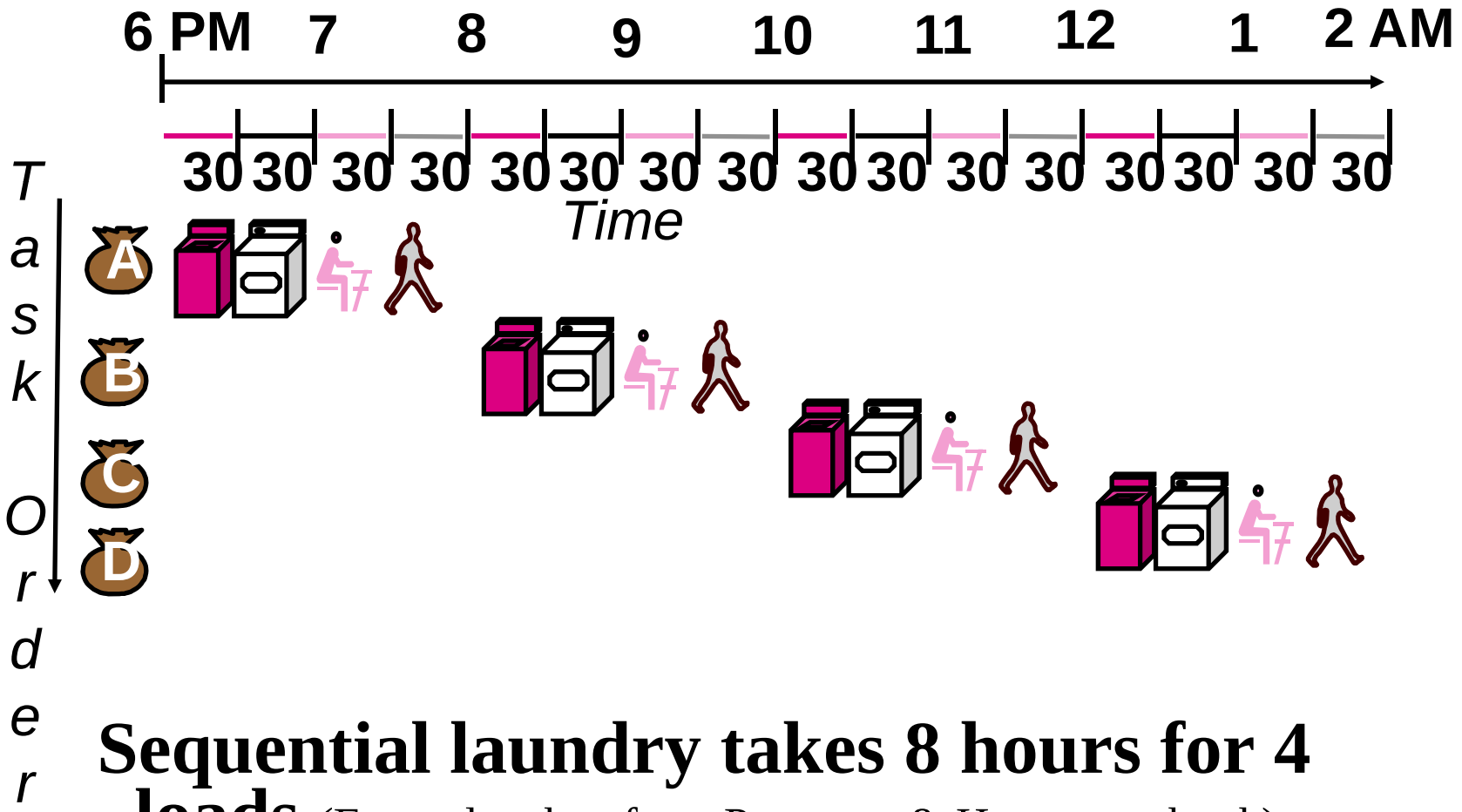
- **Dryer takes 30 minutes**



- **“Folder” takes 30 minutes**

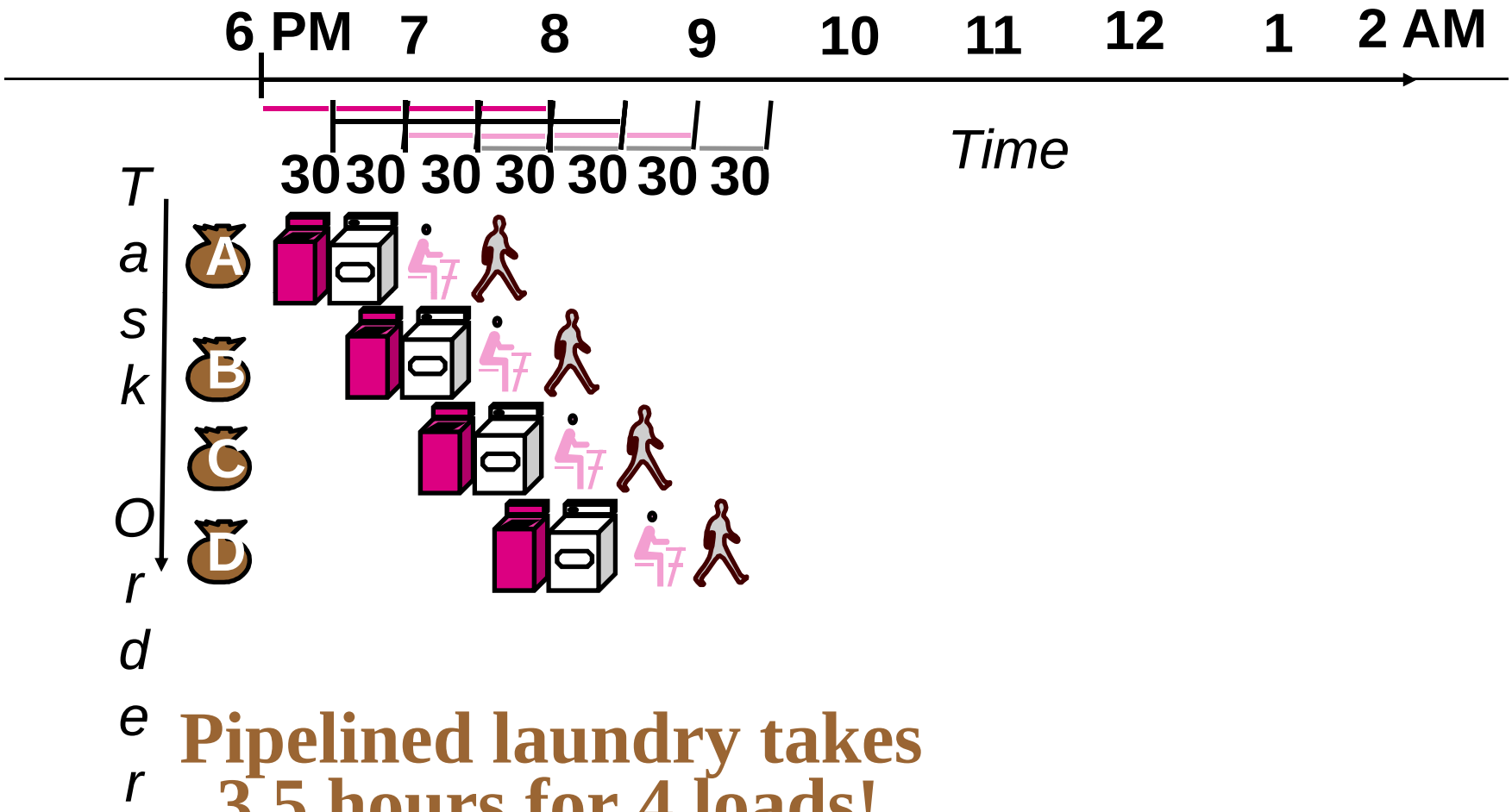
- **“Stasher” takes 30 minutes to put clothes into drawers**

# Sequential Laundry



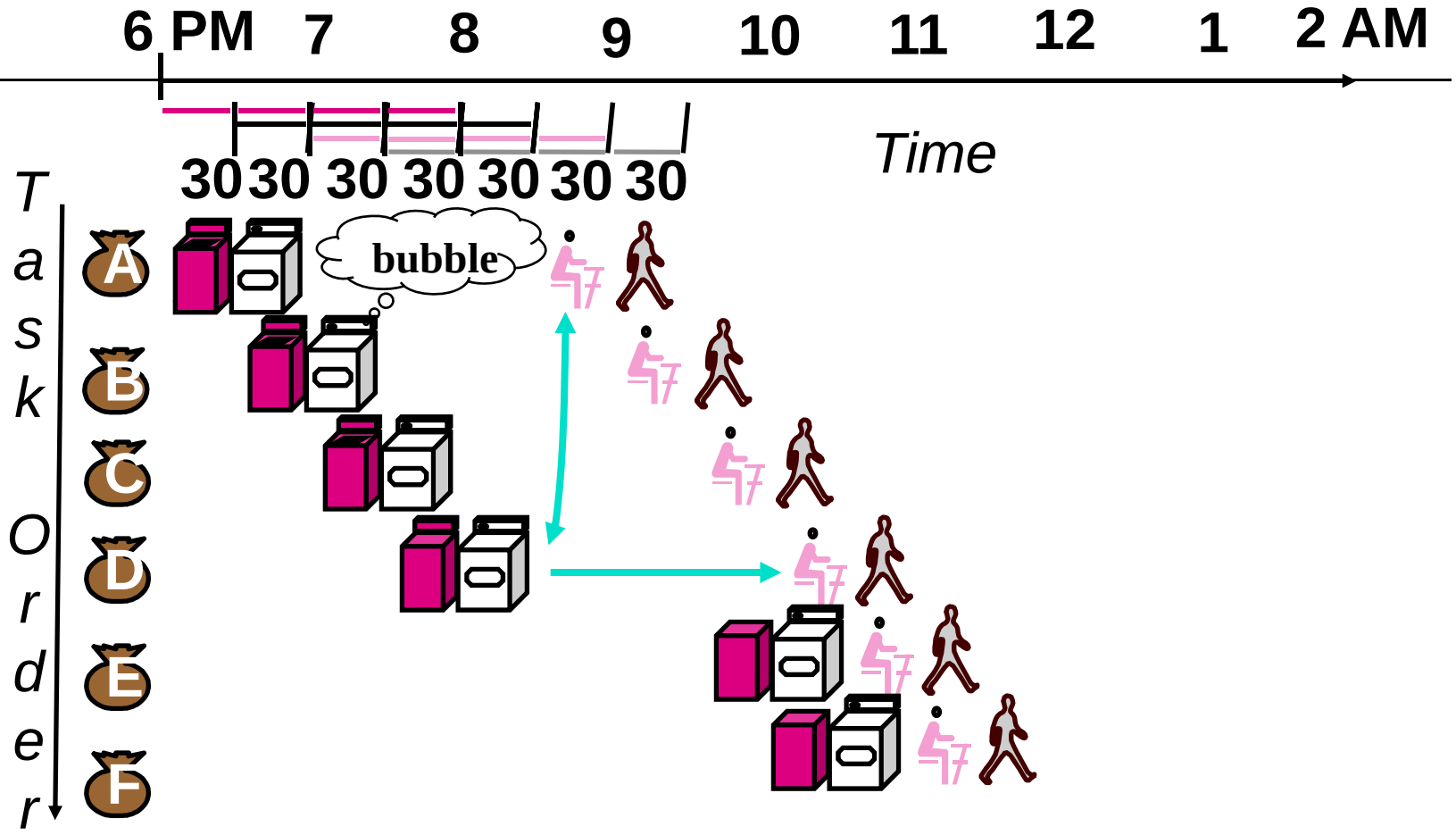

**Sequential laundry takes 8 hours for 4 loads** (Example taken from Patterson & Hennessey book)

# Pipelined Laundry: Start work ASAP




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

# Pipeline Hazard: Stall




A depends on D; stall since folder tied up

# Prediction

- There really are three different kinds of branches:
  - Forward conditional branches
    - based on a run-time condition
  - Backward conditional branches
    - The branch is based on some condition, such as branching backwards to the beginning of a program loop when a test at the end of the loop states the loop should be executed again.
  - Unconditional branches
    - this includes jumps, procedure calls and returns that have no specific condition.

# Static Branch Prediction

- Forward branches dominate backward branches by about 4 to 1 (whether conditional or not).
  - About 60% of the forward conditional branches are taken
  - Approximately 85% of the backward conditional branches are taken (because of the prevalence of program loops).
- Backward branches will be predicted to be taken, since that is the most common case.

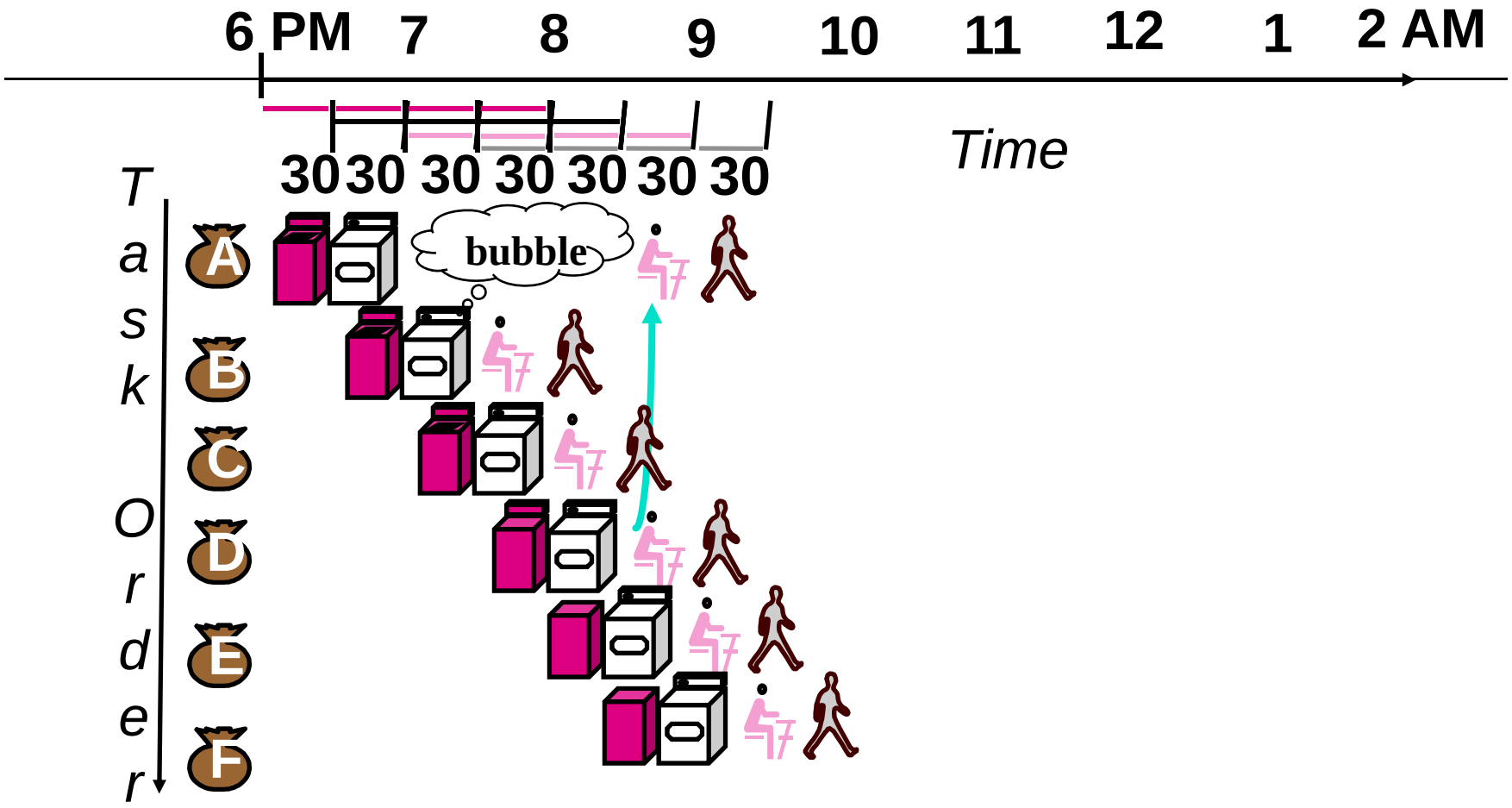
# Dynamic Branch Prediction

- Hardware-based schemes that utilize run-time behavior of branches to make dynamic predictions:
  - Information about outcomes of previous occurrences of branches are used to dynamically predict the outcome of the current branch.
  - Why? Better branch prediction accuracy and thus fewer branch stalls.

# So how to hold the trend

- Clever design techniques
  - Pipelining
    - Prediction
  - Out of order execution
  - Superscalar
    - Multiple functional units
  - Memory
    - Hierarchical memory design
    - Multilevel caches

# Out-of-Order Laundry: Don't Wait

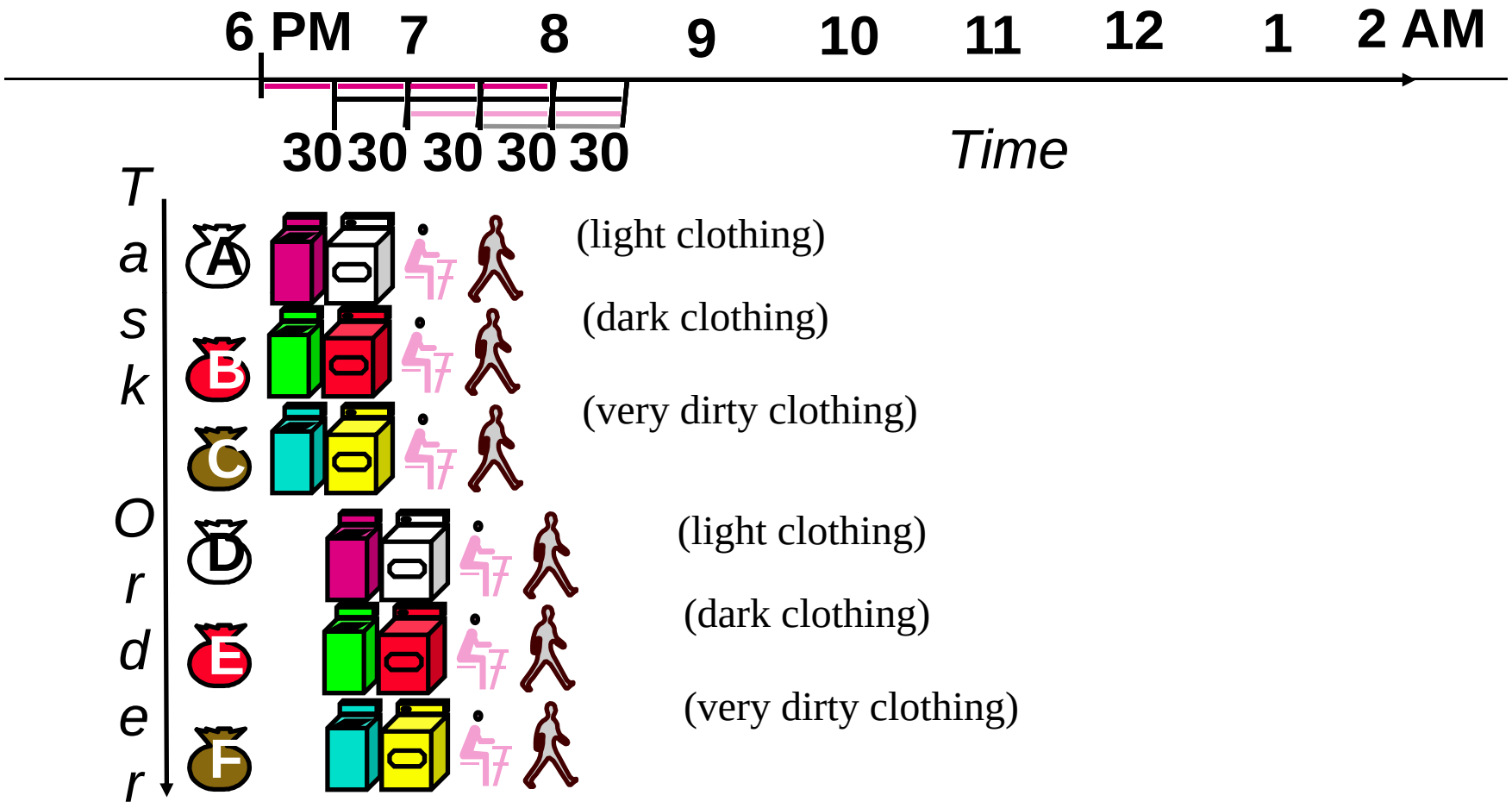



A depends on D; rest continue; need more resources to allow out-of-order

# So how to hold the trend

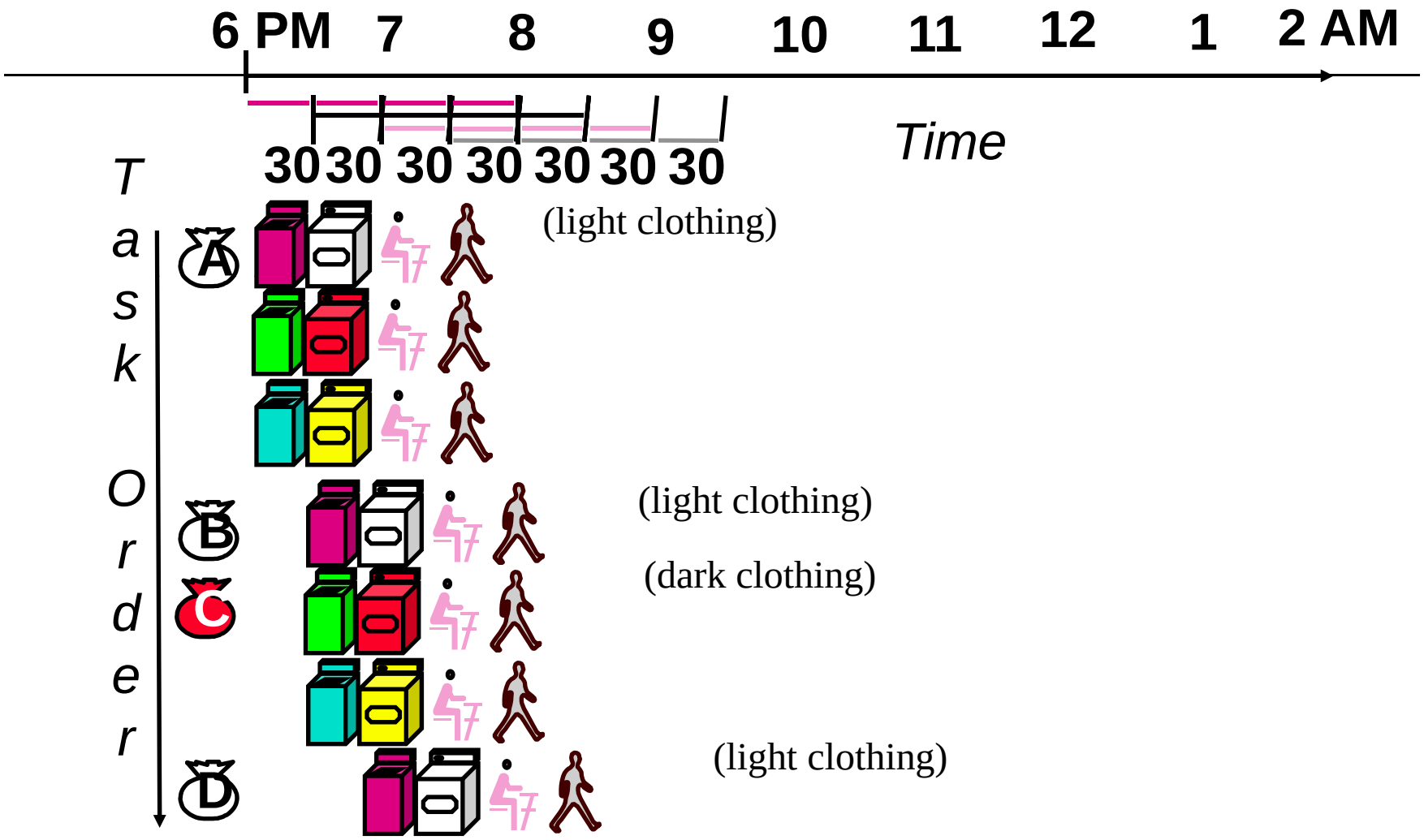
- Clever design techniques
  - Pipelining
    - Prediction
  - Out of order execution
  - Superscalar
    - Multiple functional units
  - Memory
    - Hierarchical memory design
    - Multilevel caches

# Superscalar Laundry: Parallel per stage




More resources, HW match mix of parallel tasks?

# Superscalar Laundry: Mismatch Mix



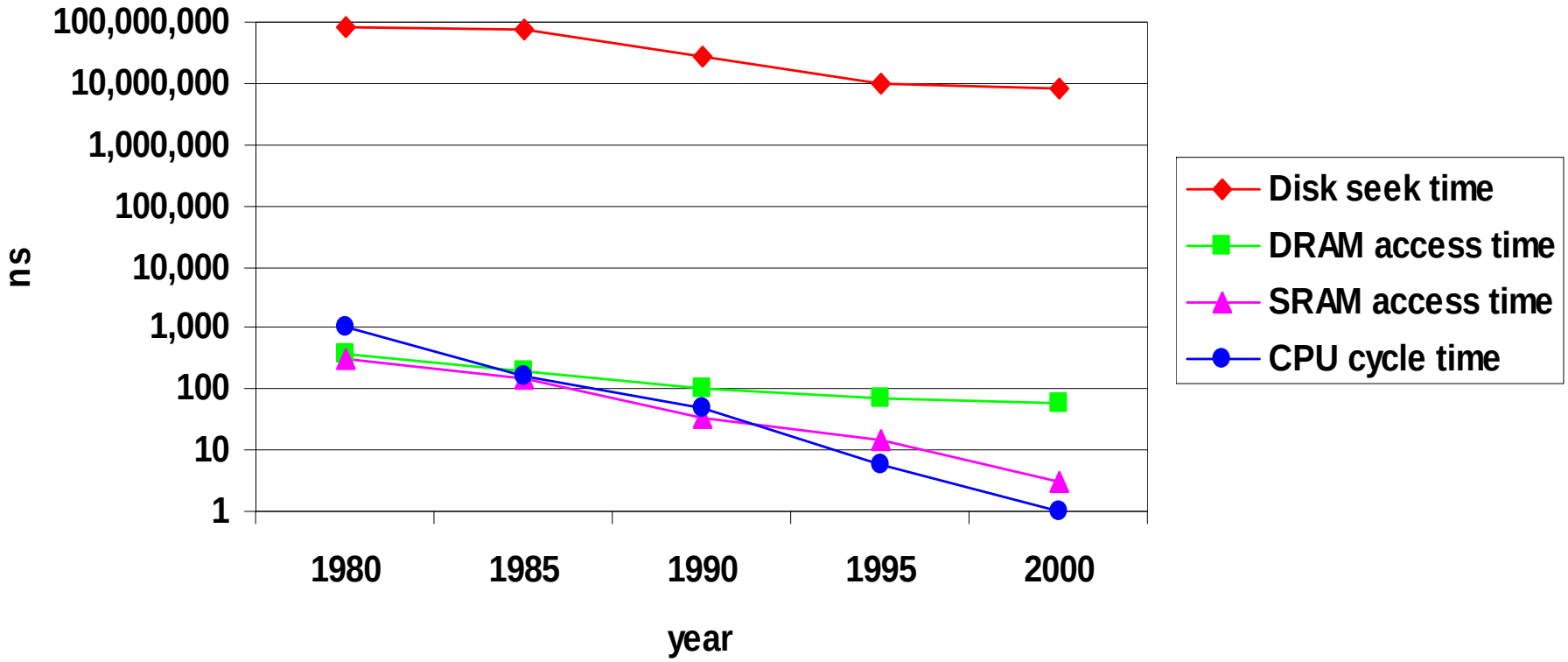

Task mix underutilizes extra resources

# So how to hold the trend

- Clever design techniques
  - Pipelining
    - Prediction
  - Out of order execution
  - Superscalar
    - Multiple functional units
  - Memory
    - Hierarchical memory design
    - Multilevel caches

# The CPU-Memory Gap

- The increasing gap between DRAM, disk, and CPU speeds.



# Locality

- Principle of Locality:
  - Programs tend to reuse data and instructions near those they have used recently, or that were recently referenced themselves.
  - Temporal locality: Recently referenced items are likely to be referenced in the near future.
  - Spatial locality: Items with nearby addresses tend to be referenced close together in time.

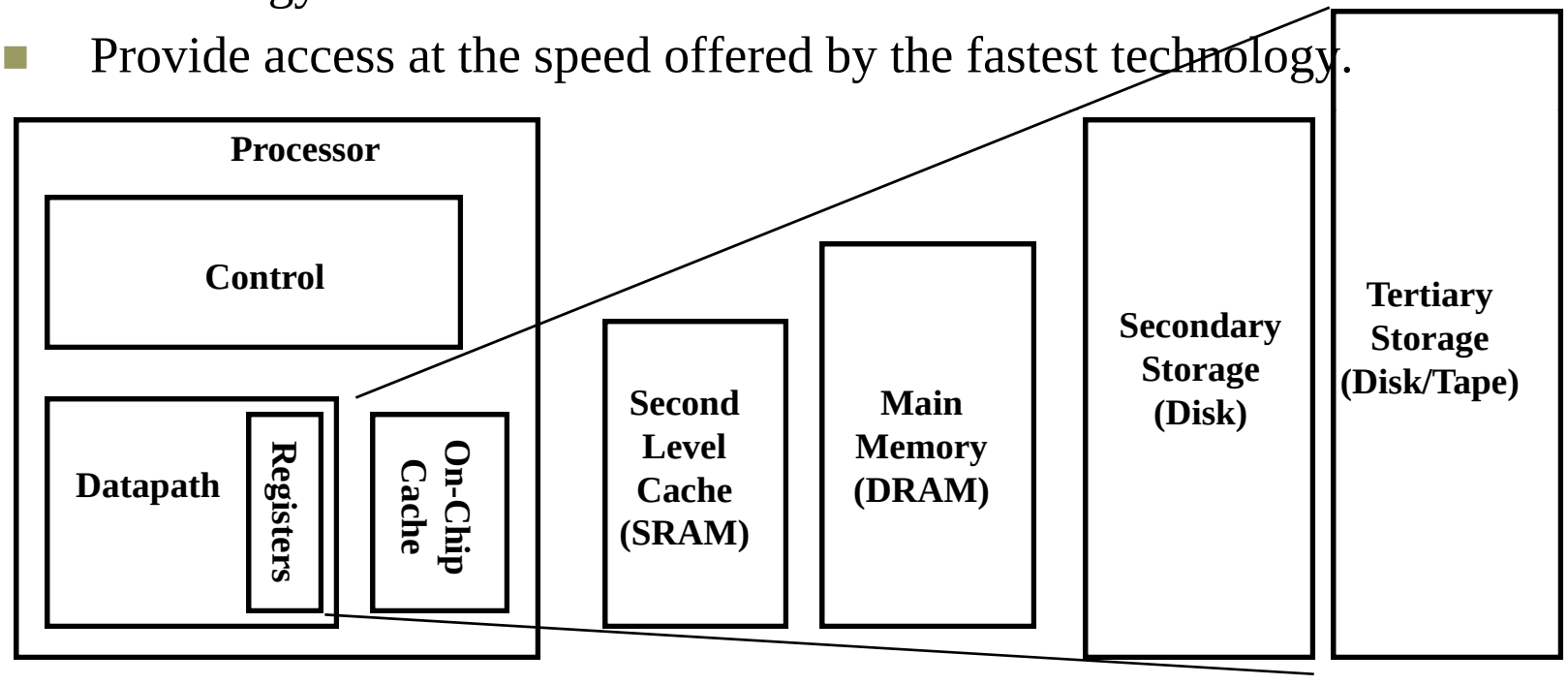
# Locality

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- Locality Example:
  - Data
    - Reference array elements in succession (stride-1 reference pattern): Spatial locality
    - Reference `sum` each iteration: Temporal locality
  - Instructions
    - Reference instructions in sequence: Spatial locality
    - Cycle through loop repeatedly Temporal locality

# A Modern Memory Hierarchy

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.




| | | | | | |
|---|---|---|---|---|---|
| **Speed (ns):** | 1s | 10s | 100s | 10,000,000s (10s ms) | 10,000,000,000s (10s sec) |
| **Size (bytes):** | 100s | Ks | Ms | Gs | Ts |

# References


- Book: ***Computer Architecture: A Quantitative Approach***, Fourth Edition, by John L. Hennessy and David A. Patterson, 2006.
- Book: ***The Sourcebook of Parallel Computing*** by Jack Dongarra, Ian Foster, Geoffrey C. Fox, and William Gropp, 2002.
- Dr. MAREK ANDRZEJ PERKOWSKI lecture for CS 252, Portland State University.
- Dr. John Kubiatowicz lecture for CS252, Berkeley University.
- Dr. Roberto Lopez lecture for ECE 684 lecture, New Jersey Institute of Technology.