High Performance Scientific computing
Lecture 3

S. Gopalakrishnan

Von Neumann Architecture

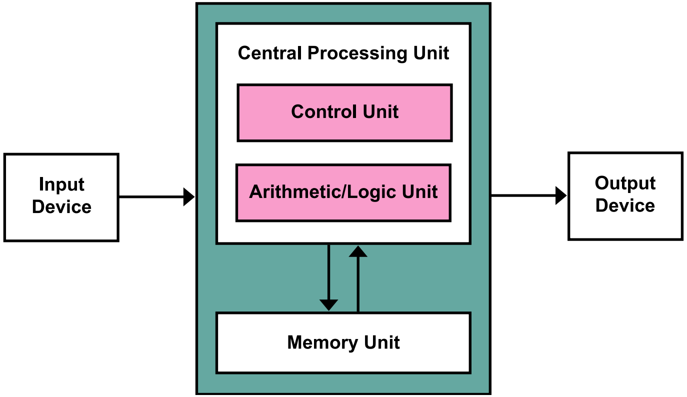


Image: Wikipedia

Program Code Levels

- High-level language
    - Level of abstraction closer to problem domain
    - Provides for productivity and portability

```
for( i=0; i<=NR+1; i++ )       /* Copy the values into told */
    for( j=0; j<=NC+1; j++ )
      told[i][j] = t[i][j];
```

Program Code Levels

- High-level language
    - Level of abstraction closer to problem domain
    - Provides for productivity and portability

```
for( i=0; i<=NR+1; i++ )       /* Copy the values into told */
    for( j=0; j<=NC+1; j++ )
      told[i][j] = t[i][j];
```

- Assembly language
    - Textual representation of instructions.
    - Compilation and linking of program will yield this

```
 je else
 jmp endif
else:
endif:
 sarl $1, %edx
 movl %edx, %eax
 addl %eax, %edx
 addl %eax, %edx
 addl $1, %edx
 addl $1, %ecx
```

- Hardware representation

Program Code Levels

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability

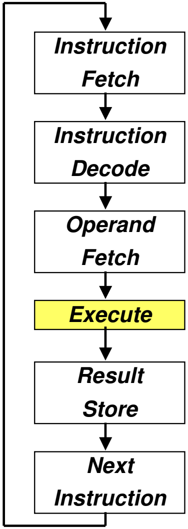
```
for( i=0; i<=NR+1; i++ )       /* Copy the values into told */
    for( j=0; j<=NC+1; j++ )
      told[i][j] = t[i][j];
```

- Assembly language
  - Textual representation of instructions.
  - Compilation and linking of program will yield this

```
 je else
 jmp endif
else:
endif:
 sarl $1, %edx
 movl %edx, %eax
 addl %eax, %edx
 addl %eax, %edx
 addl $1, %edx
 addl $1, %ecx
```

- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data

Execution cycle



Obtain instruction from program storage

Determine required actions and instruction size

Locate and obtain operand data

Compute result value or status

Deposit results in storage for later use

Determine successor instruction

Credit: Based on notes from EC232 UMass – Amherst

Instruction Pipelining

Increases instruction throughput, i.e number of instructions executed per second. But time for each instruction remains the same.

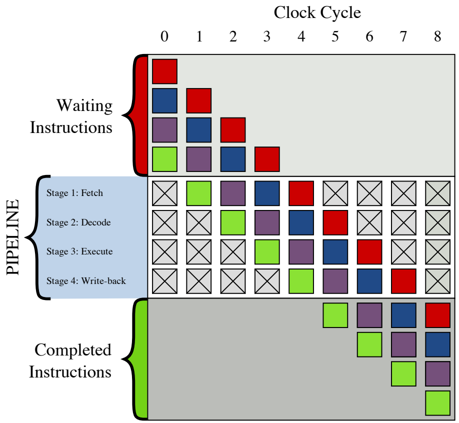


Image: Wikipedia

Pipelining Issues

Data dependencies can result in slowing of throughput.

```
for( i=0; i<=LAST; i++ )
a[i]=b[i]+c[i]
```

No deadlock in previous sample, pipelining will work like a charm.

Pipelining Issues

Data dependencies can result in slowing of throughput.

```
for( i=0; i<=LAST; i++ )
a[i]=b[i]+c[i]
```

No deadlock in previous sample, pipelining will work like a charm.

```
for( i=0; i<=LAST; i++ )
a[i]=b[i]+a[i-1]
```

This is a deadlock situation since $a[i]$ depends on $a[i-1]$

Pipelining Issues

Bubbles in pipelines due to dependencies.

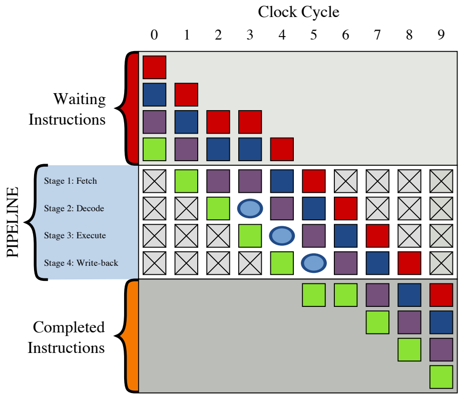
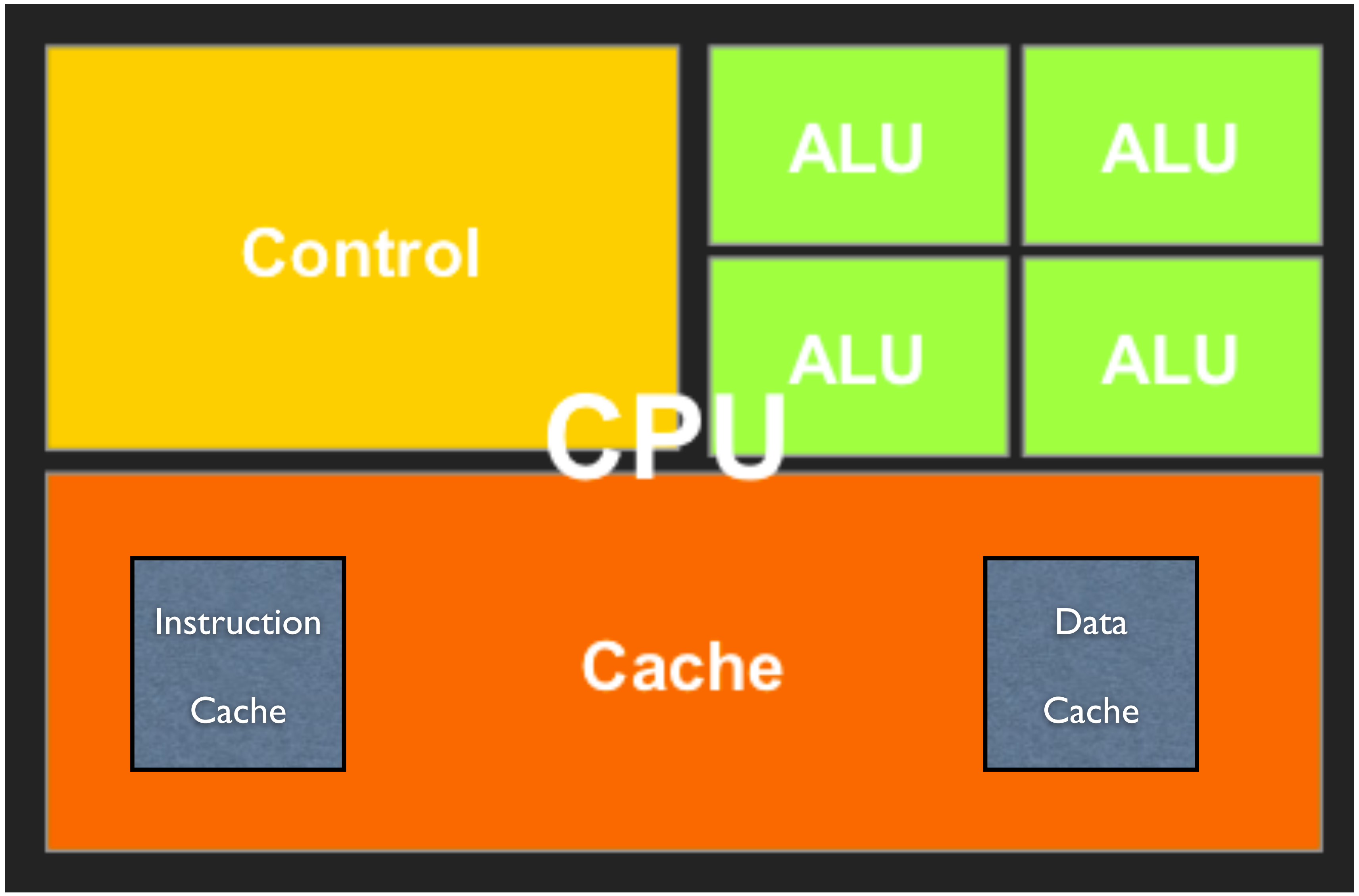


Image: Wikipedia

Memory Issues

CPU Block Diagram



Source:Prof. Wen-mei W. Hwu UIUC

Memory Basics

Users want large and fast memories!

Fact
- Large memories are slow
- Fast memories are small

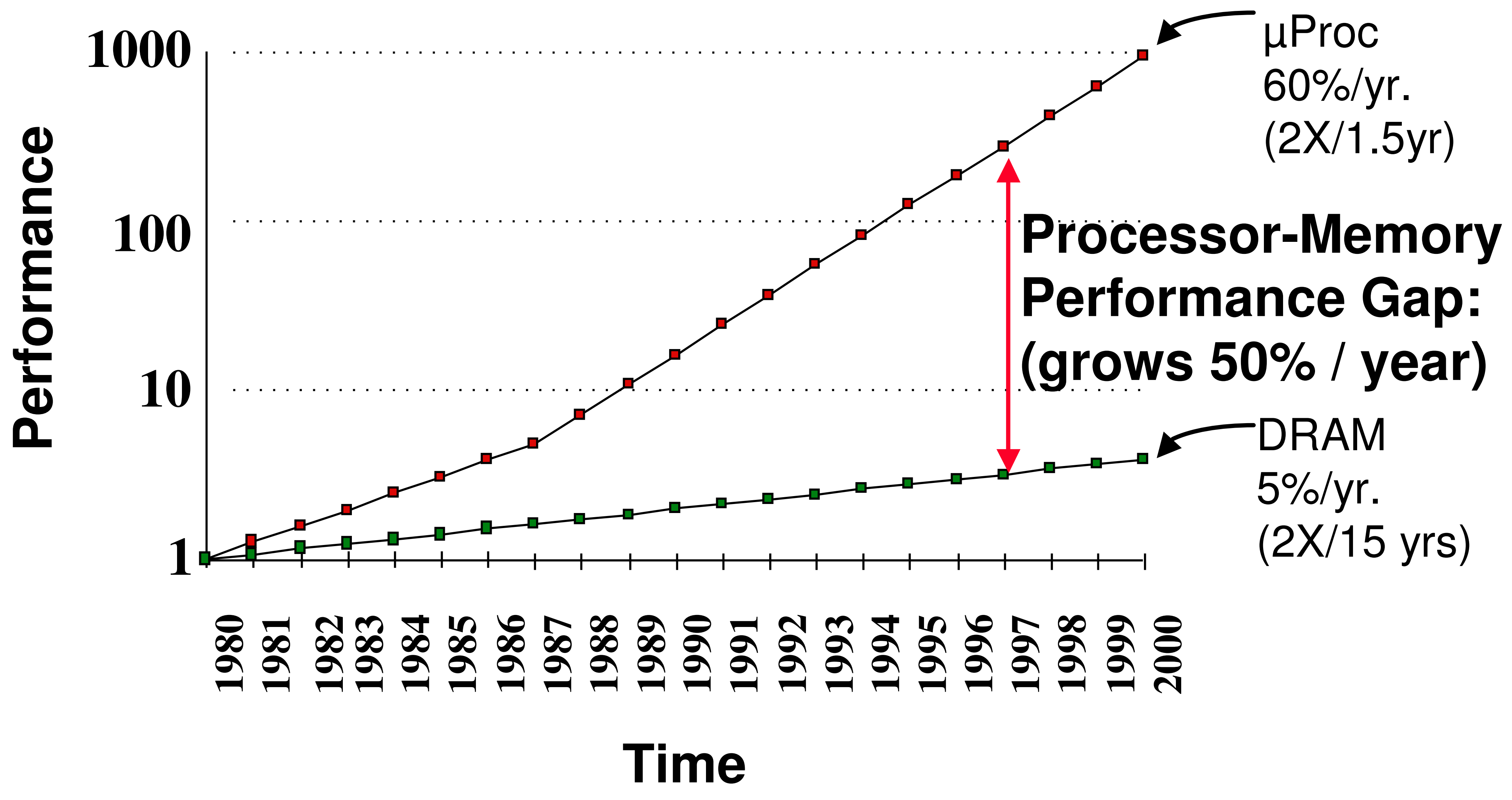
Assume CPU runs at 3GHz
Every instruction requires 4B of instruction and at least one memory access (4B of data)

- 3 * 8 = 24GB/sec

Memory bandwidth and access time is a performance bottleneck

Source: ECE232 Umass-Amherst

Processor-DRAM Memory Performance Gap
Motivation for Memory Hierarchy



Source:Ece 232 Umass-Amherst

Memory Basics

Small memories are fast

--So just write small programs

"640 K of memory should be enough for anybody" -- Bill Gates, 1981

Today's programs require large memories

- Powerpoint 2003 – 25 megabytes
- Scientific applications may require Gigabytes of memory

How do we create a memory that is large, cheap and fast (most of the time)?

Strategy: Provide a Small, Fast Memory which holds a subset of the main memory – called cache

- Keep frequently-accessed locations in fast cache
- Cache retrieves more than one word at a time
- Sequential accesses are faster after first access

Source: ECE232 Umass-Amherst

Memory Basics

Hierarchy of Levels

- Uses smaller and faster memory technologies close to the processor
- Fast access time in highest level of hierarchy
- Cheap, slow memory furthest from processor
- The aim of memory hierarchy design is to have access time close to the highest level and size equal to the lowest level

Basic Philosophy

Move data into 'smaller, faster' memory

Operate on it

Move it back to 'larger, cheaper' memory

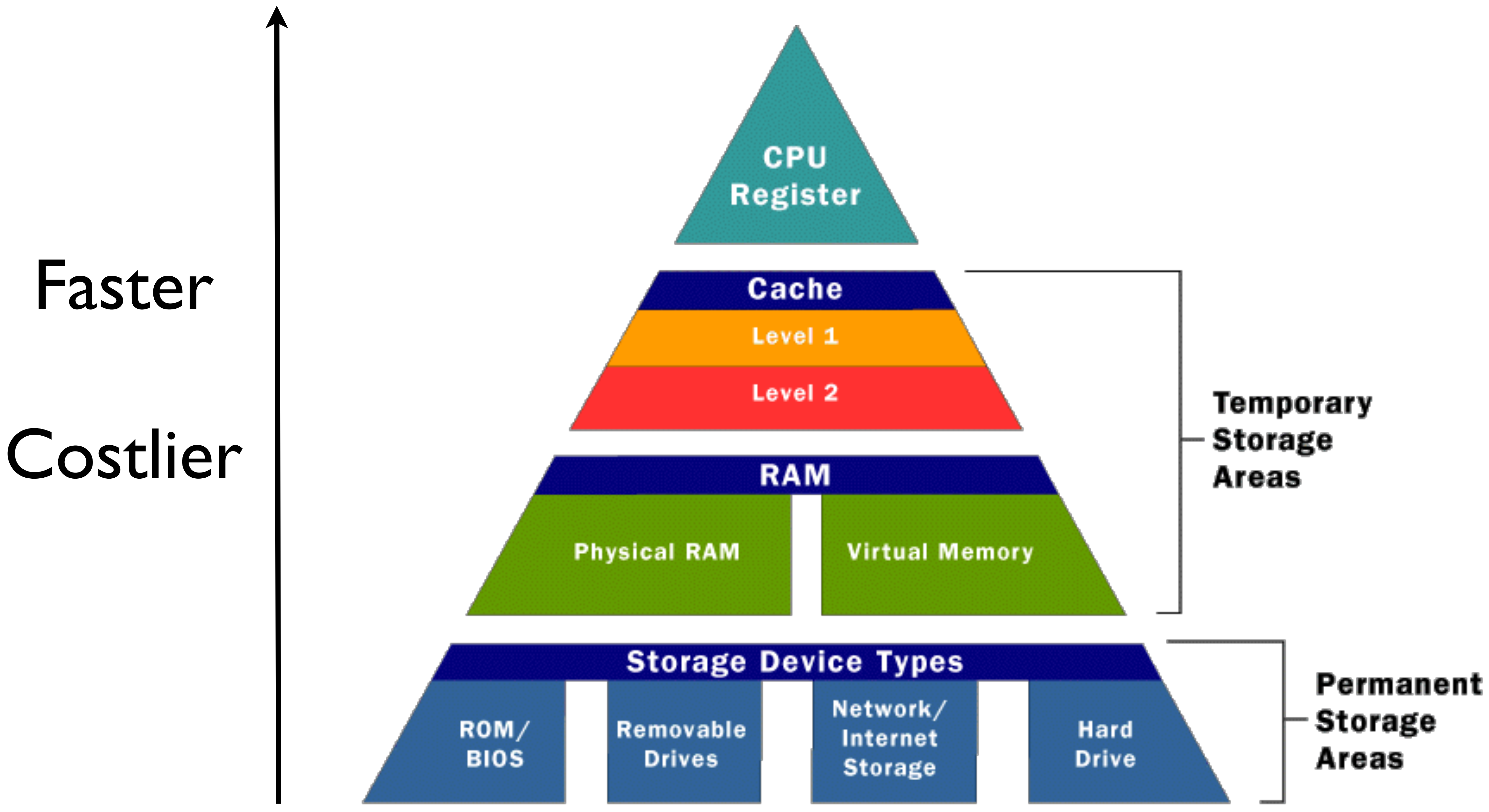
• How do we keep track if changed

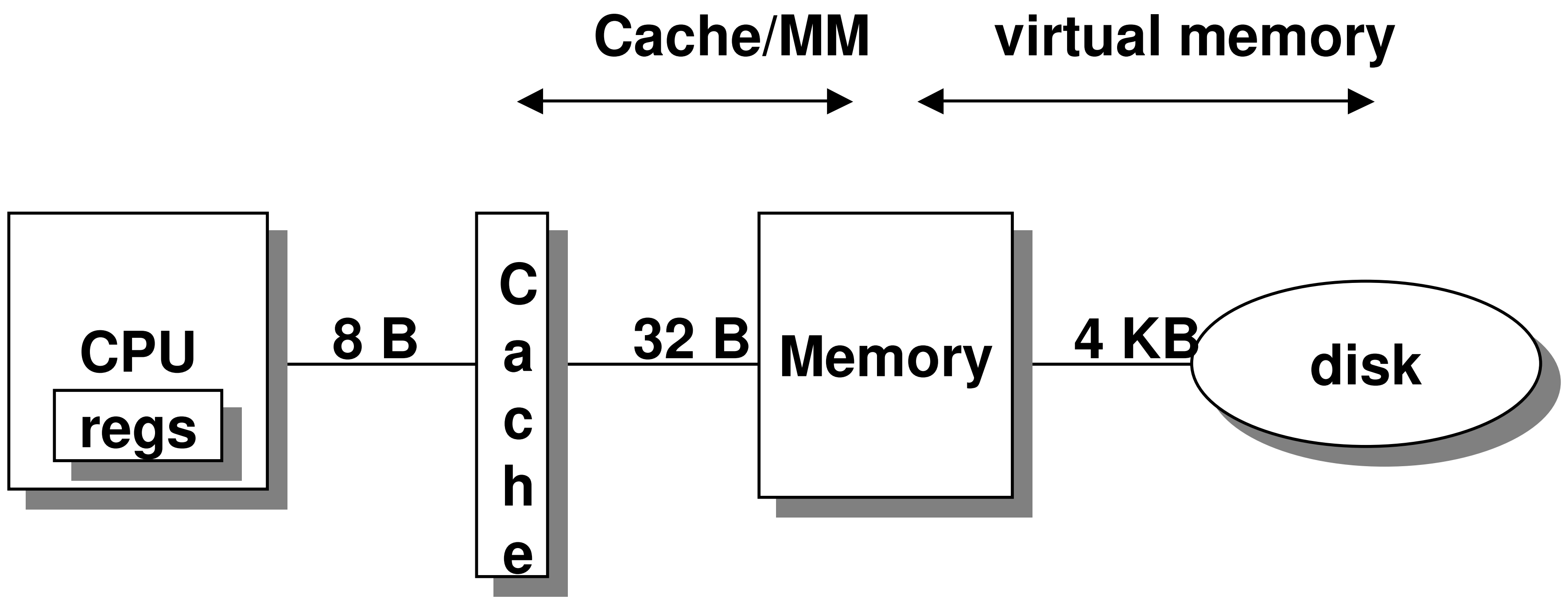
What if we run out of space in 'smaller, faster' memory?

Important Concepts: Latency, Bandwidth

Source: ECE232 Umass-Amherst

Memory hierarchy

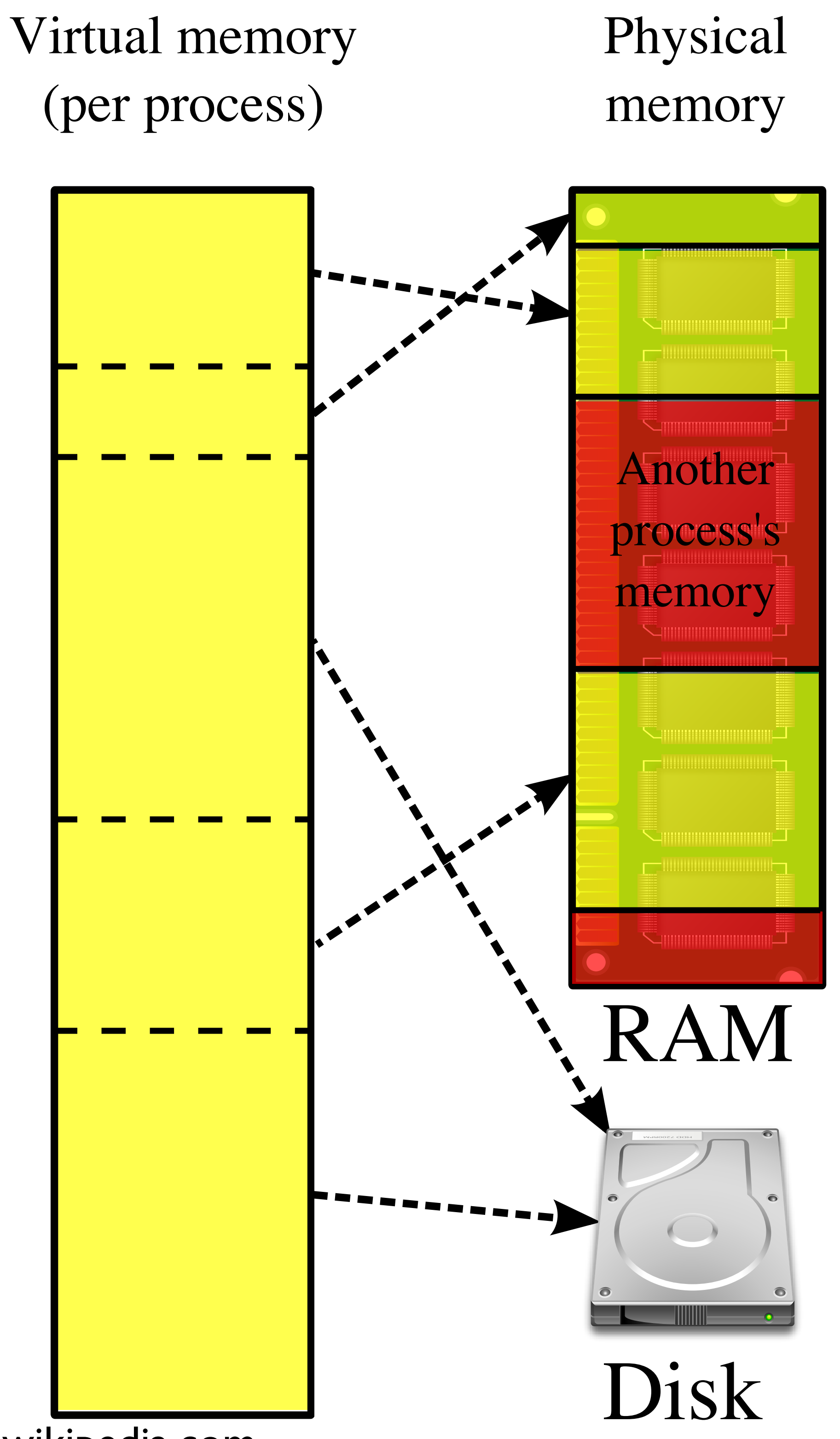




- Notice that the data width is changing
  - Why?
- Bandwidth: Transfer rate between various levels
  - CPU-Cache: 24 GBps
  - Cache-Main: 0.5-6.4GBps
  - Main-Disk: 187MBps (serial ATA/1500)

Source:Ece 232 Umass-Amherst

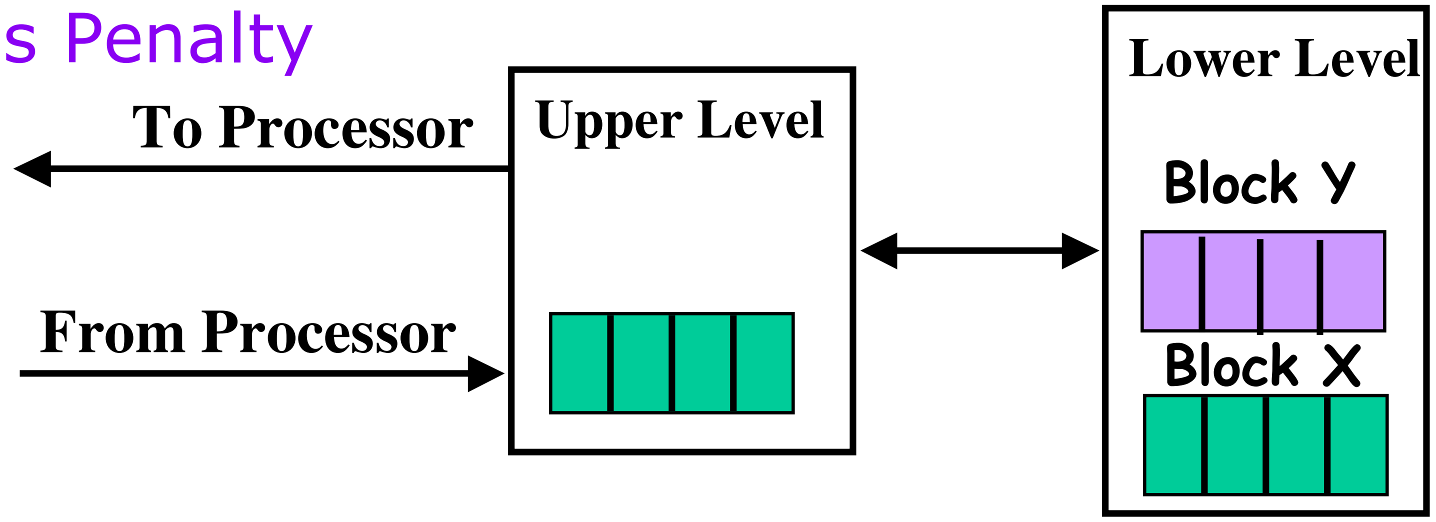
Virtual Memory and Paging



Source: www. wikipedia.com

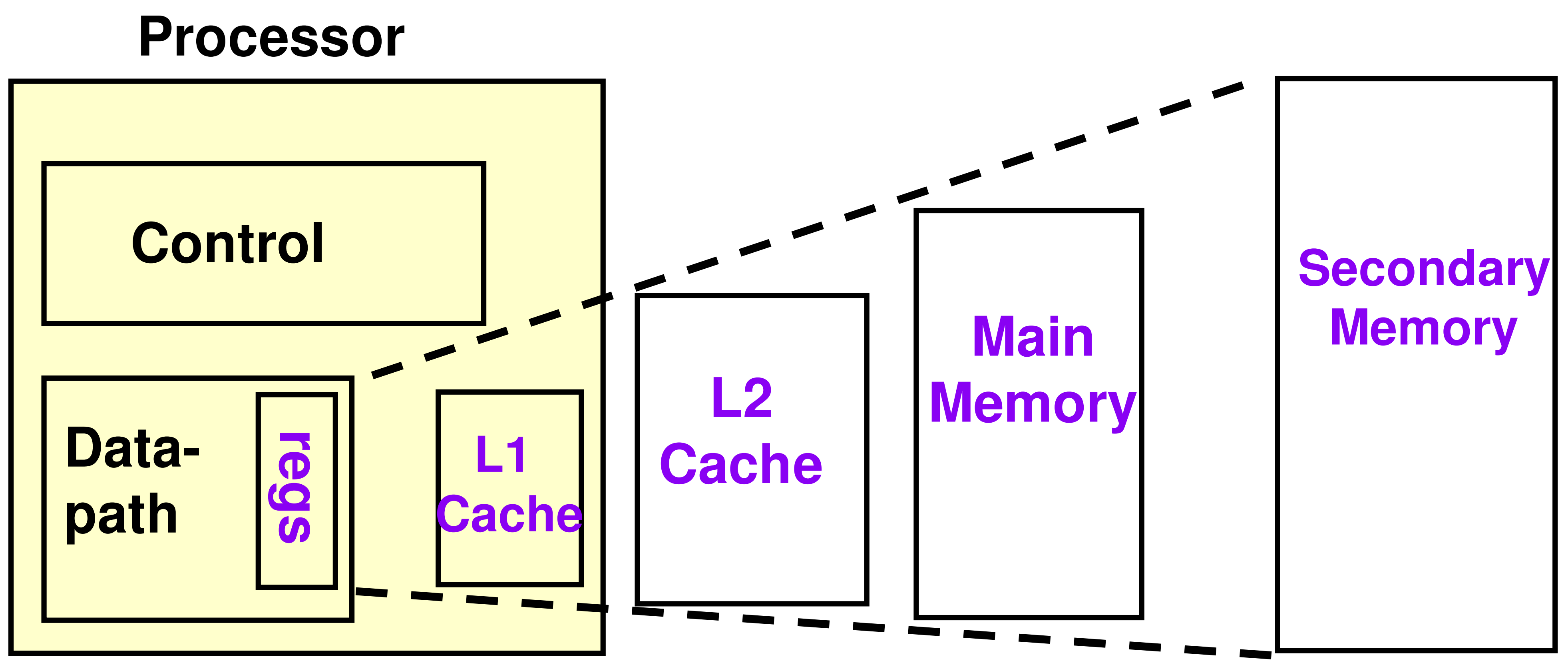
Memory Hierarchy Terminology

- Hit: data appears in upper level in block X
- Hit Rate: the fraction of memory accesses found in the upper level
- Miss: data needs to be retrieved from a block in the lower level (Block Y)
- Miss Rate = 1 - (Hit Rate)
- Hit Time: Time to access the upper level which consists of Time to determine hit/miss + upper level access time
- Miss Penalty: Time to replace a block in the upper level + Time to deliver the block to the processor
- Note: Hit Time << Miss Penalty



Source: ECE 232 Umass-Amherst

Current Memory Hierarchy



Speed(ns):	1ns	2ns	6ns	100ns	10,000,000ns
Size (MB):	0.0005	0.1	1-4	1000-6000	500,000
Cost ($/MB):	--	$10	$3	$0.01	$0.002
Technology:	Regs	SRAM	SRAM	DRAM	Disk

- Cache - Main memory: Speed
- Main memory – Disk (virtual memory): Capacity

Source:Ece 232 Umass-Amherst