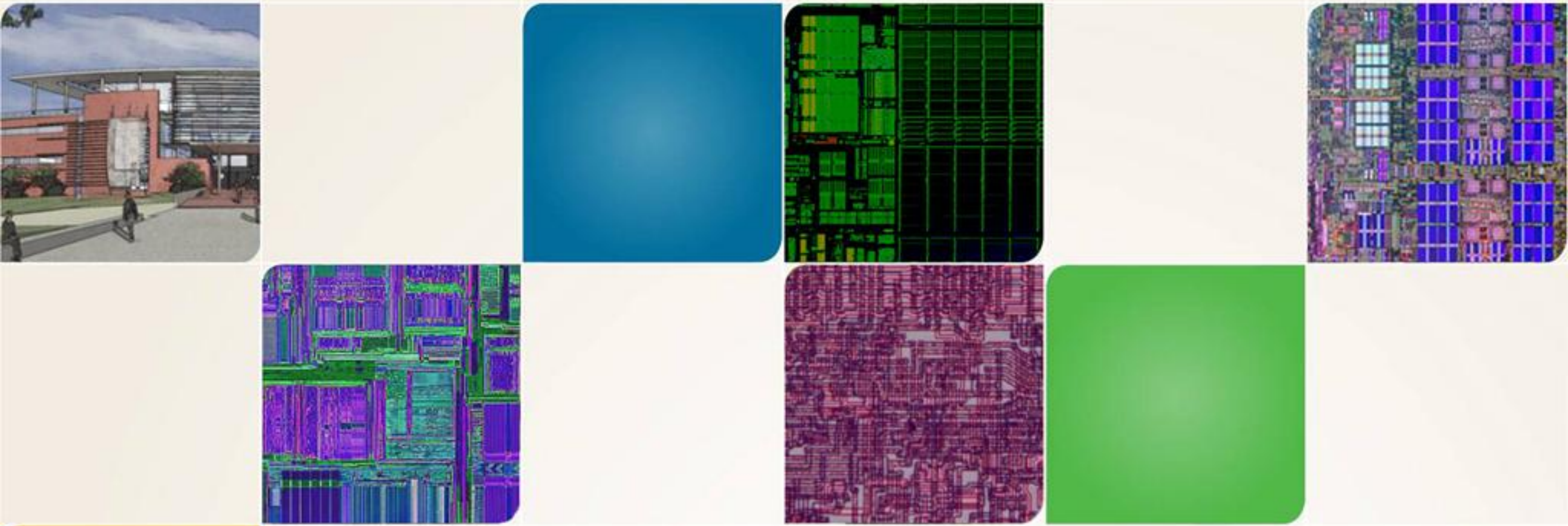


CS4803DGC Design Game Consoles

Spring 2010

Prof. Hyesoon Kim

Georgia Tech College of Computing

Thanks to Prof. Loh & Prof. Prvulovic

SIMD

Multiprocessing

- Flynn’s Taxonomy of Parallel Machines
  - How many Instruction streams?
  - How many Data streams?
- SISD: Single I Stream, Single D Stream
  - A uniprocessor
- SIMD: Single I, Multiple D Streams
  - Each “processor” works on its own data
  - But all execute the same instrs in lockstep
  - E.g. a vector processor or MMX, CUDA

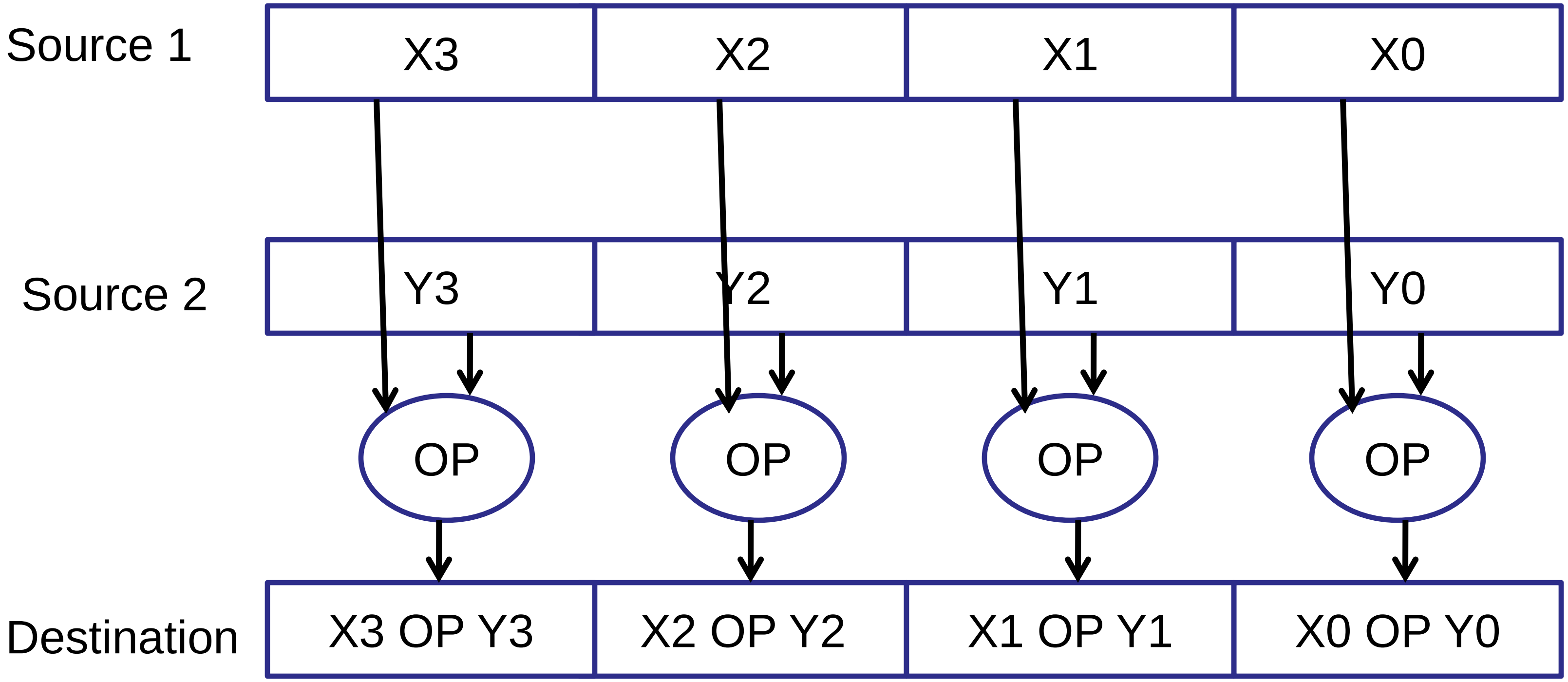
Flynn’s Taxonomy

- MISD: Multiple I, Single D Stream
  - Not used much
  - Stream processors are closest to MISD
- MIMD: Multiple I, Multiple D Streams
  - Each processor executes its own instructions and operates on its own data
  - This is your typical off-the-shelf multiprocessor (made using a bunch of “normal” processors)
  - Includes multi-core processors

SIMD Model

- Texas C62xx, IA32 (SSE), AMD K6, CUDA, Xbox..
- Early SIMD machines: e.g.) CM-2 (large distributed system)
  - Lack of vector register files and efficient transposition support in the memory system.
  - Lack of irregular indexed memory accesses
- Modern SIMD machines:
  - SIMD engine is in the same die

SIMD Execution Model



Source 1: X3 | X2 | X1 | X0

Source 2: Y3 | Y2 | Y1 | Y0

OP | OP | OP | OP

Destination: X3 OP Y3 | X2 OP Y2 | X1 OP Y1 | X0 OP Y0

```
for (ii = 0; ii < 4; ii++)
x[ii] = y[ii]+z[ii];
```

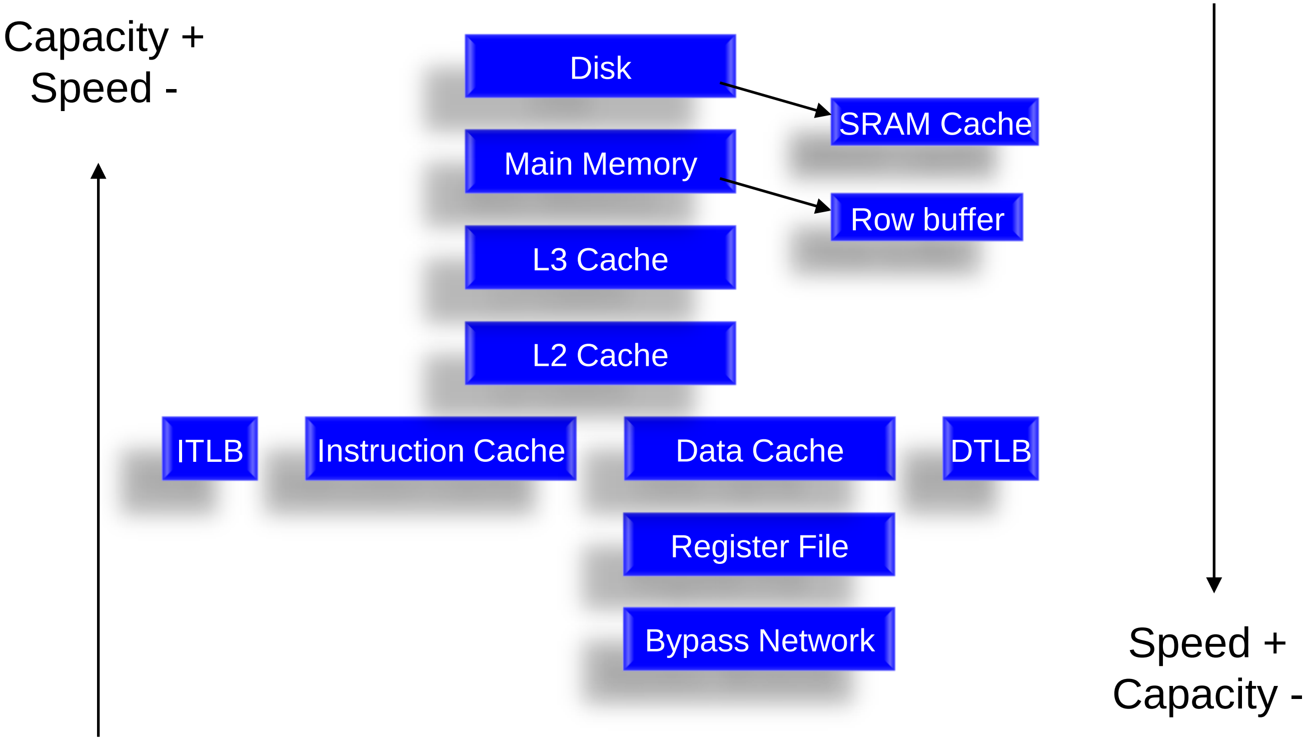
```
SIMD_ADD(X, Y, Z)
```

Locality and Caches

- Data Locality
  - Temporal: if data item needed now,
    it is likely to be needed again in near future
  - Spatial: if data item needed now,
    nearby data likely to be needed in near future
- Exploiting Locality: Caches
  - Keep recently used data
    in fast memory close to the processor
  - Also bring nearby data there

MEMORY SYSTEM

Storage Hierarchy and Locality



Capacity +
Speed -

Disk

SRAM Cache

Main Memory

Row buffer

L3 Cache

L2 Cache

ITLB

Instruction Cache

Data Cache

DTLB

Register File

Bypass Network

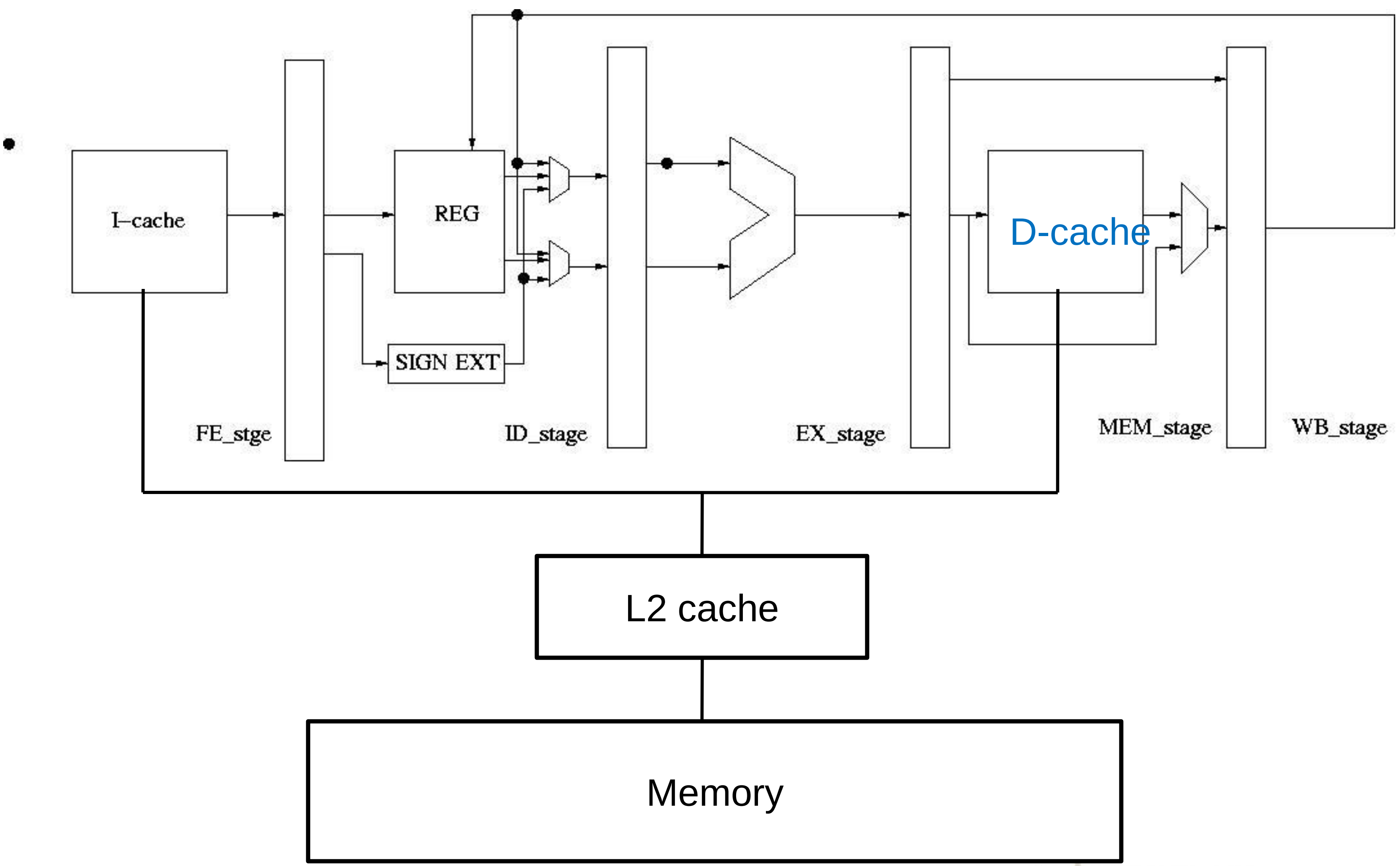
Speed +
Capacity -

Memory Latency is *Long*

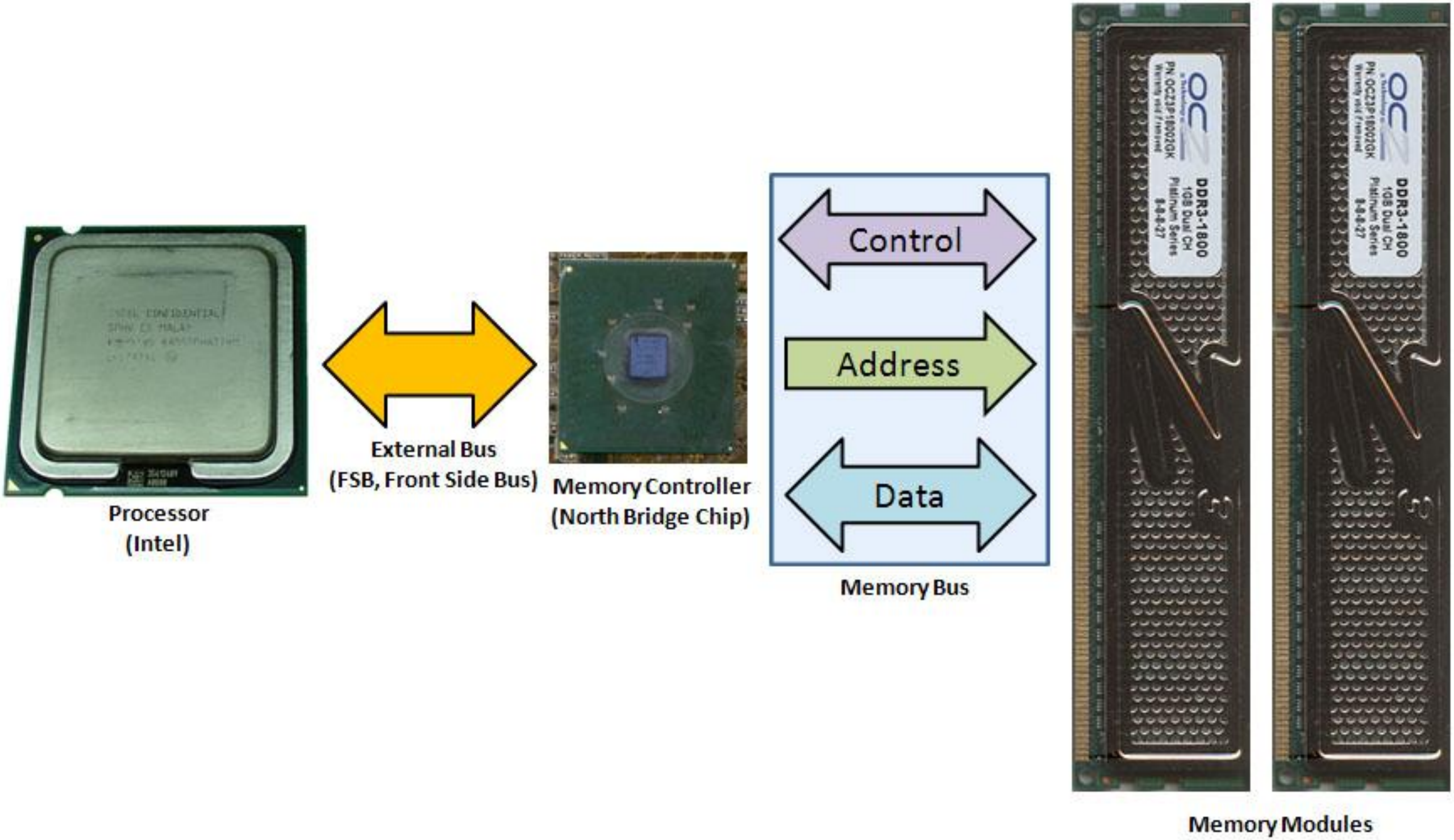
- 60-100ns not uncommon
- Quick back-of-the-envelope calculation:
  - 2GHz CPU
  - → 0.5ns / cycle
  - 100ns memory → 200 cycle memory latency!

- Solution: Caches

Cache


I–cache
REG
SIGN EXT
D-cache
FE_stge
ID_stage
EX_stage
MEM_stage
WB_stage
L2 cache
Memory

CPU-DRAM



Processor
(Intel)

External Bus
(FSB, Front Side Bus)

Memory Controller
(North Bridge Chip)

Control

Address

Data

Memory Bus

Memory Modules

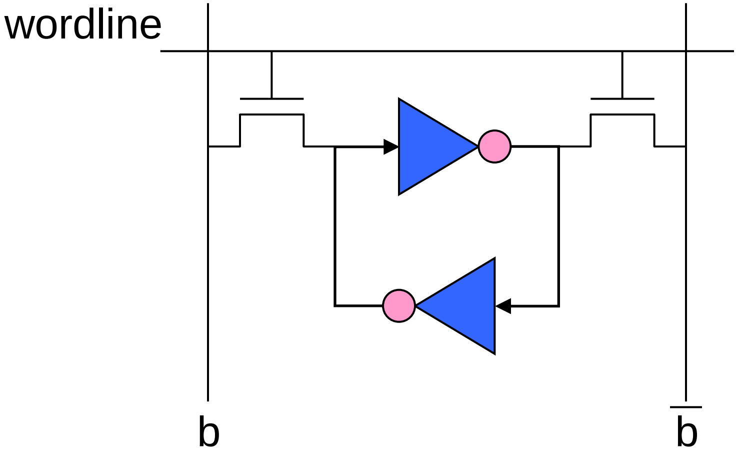
SRAM vs. DRAM

- DRAM = Dynamic RAM

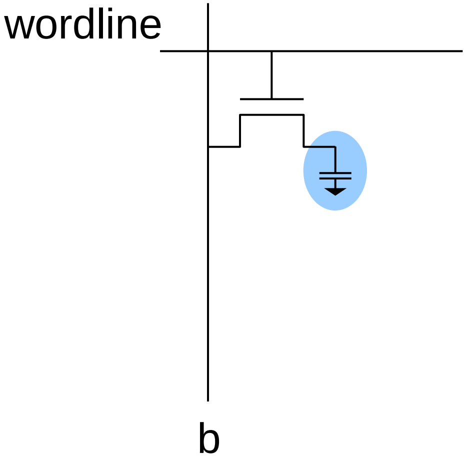
- SRAM: 6T per bit
  - built with normal high-speed CMOS technology
- DRAM: 1T per bit
  - built with special DRAM process optimized for density

Hardware Structures

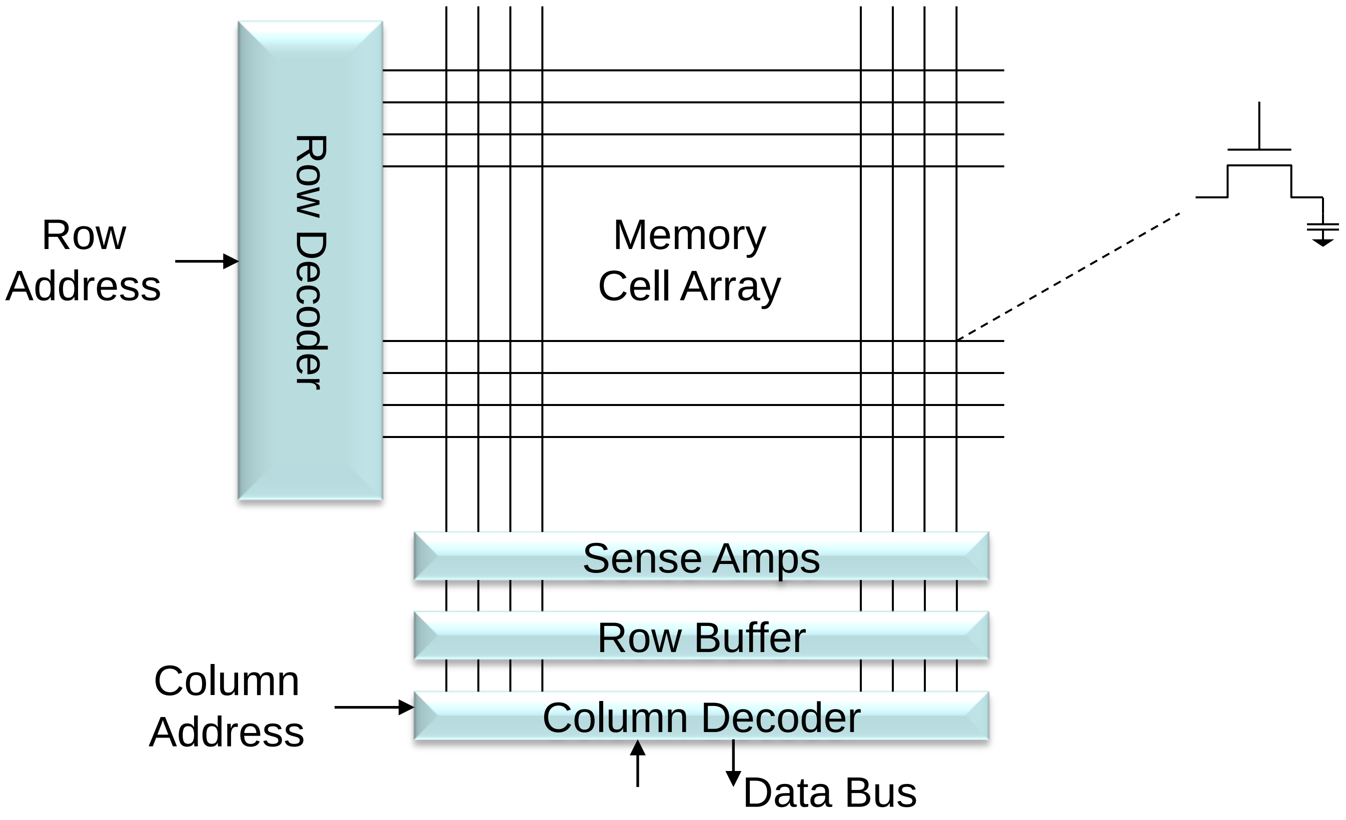
SRAM



DRAM



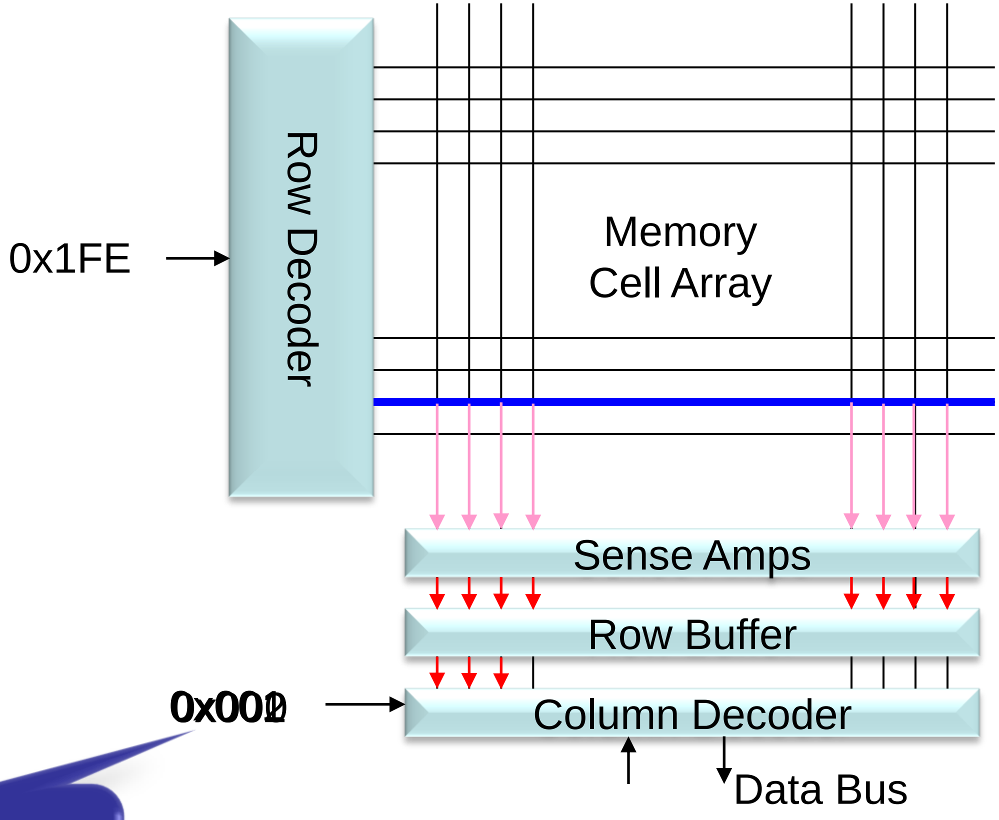
DRAM Chip Organization



DRAM Chip Organization (2)

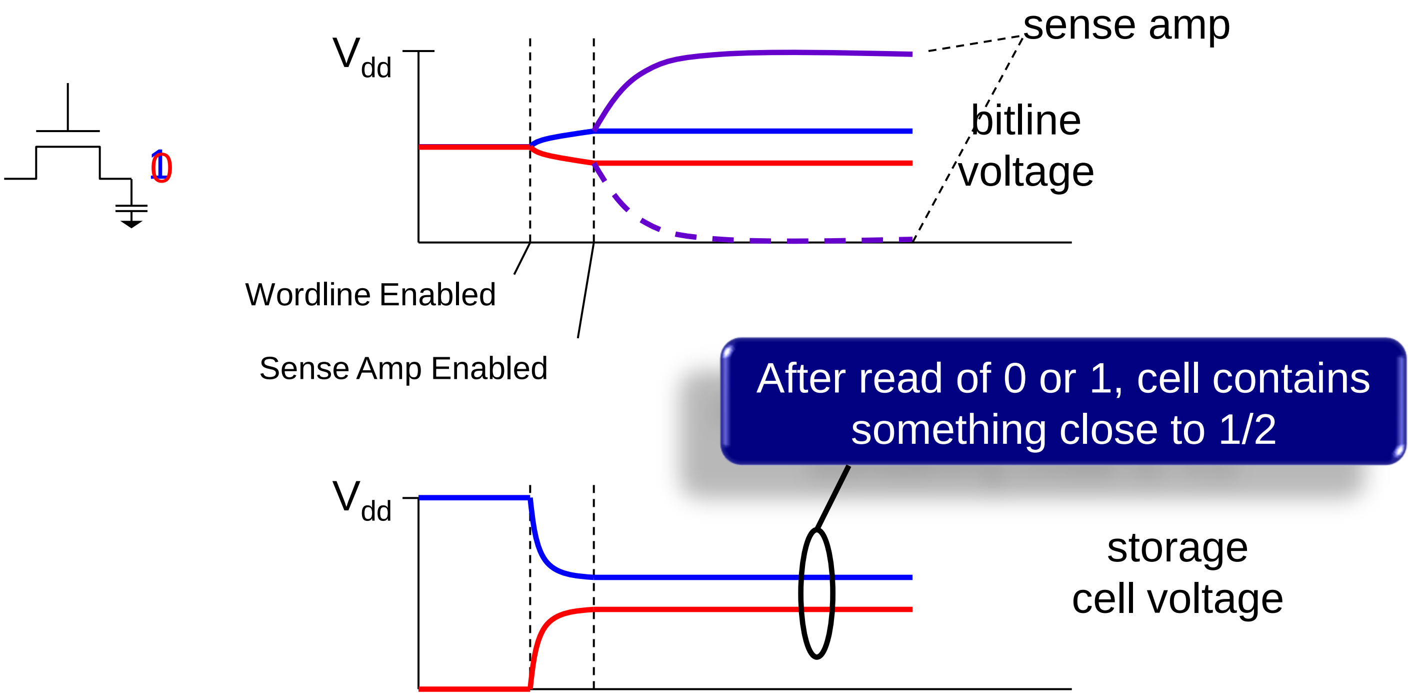
- Differences with SRAM
    - reads are *destructive*: contents are erased after reading
  - row buffer
    - read lots of bits all at once, and then parcel them out based on different column addresses
      - similar to reading a full cache line, but only accessing one word at a time
    - “Fast-Page Mode” FPM DRAM organizes the DRAM row to contain bits for a complete page
      - row address held constant, and then fast read from different locations from the same page

DRAM Read Operation



Accesses need not be sequential

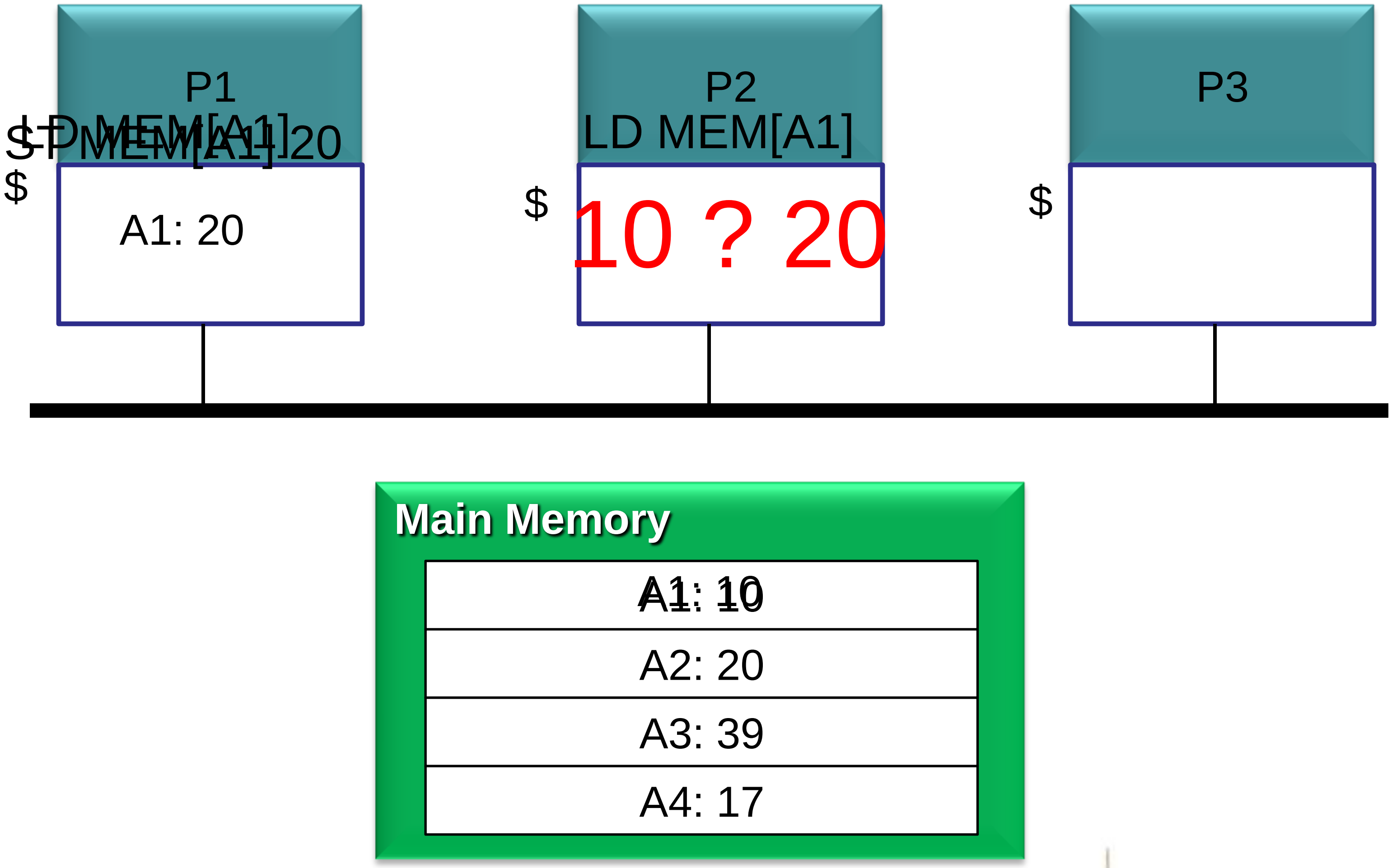
Destructive Read



After read of 0 or 1, cell contains something close to 1/2

CACHE COHERENCE

Problem



P1

LD MEM[A1]

ST MEM[A1]20

$

A1: 20

P2

LD MEM[A1]

$

10 ? 20

P3

$

Main Memory

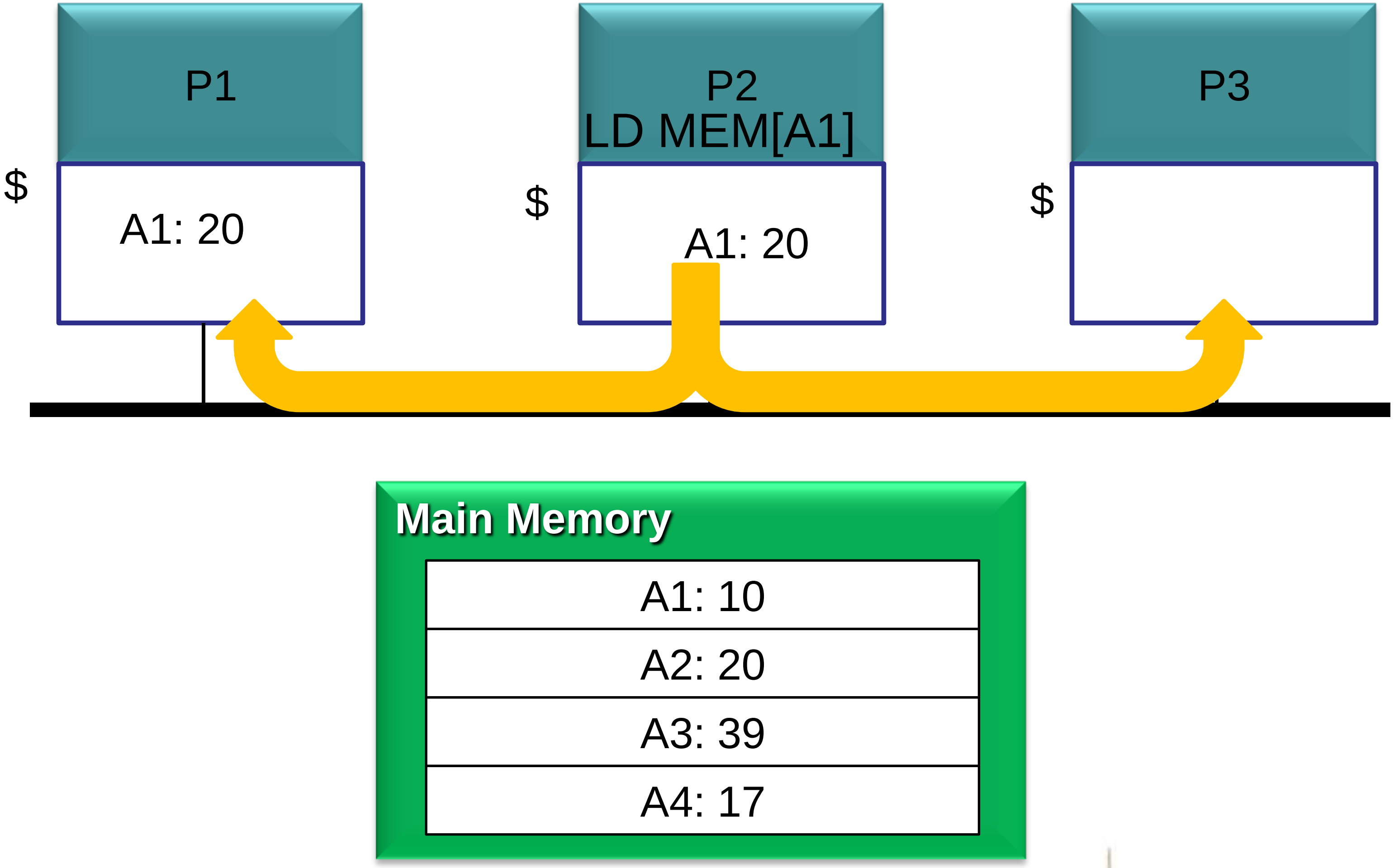
A1: 10

A2: 20

A3: 39

A4: 17

SNOOPING

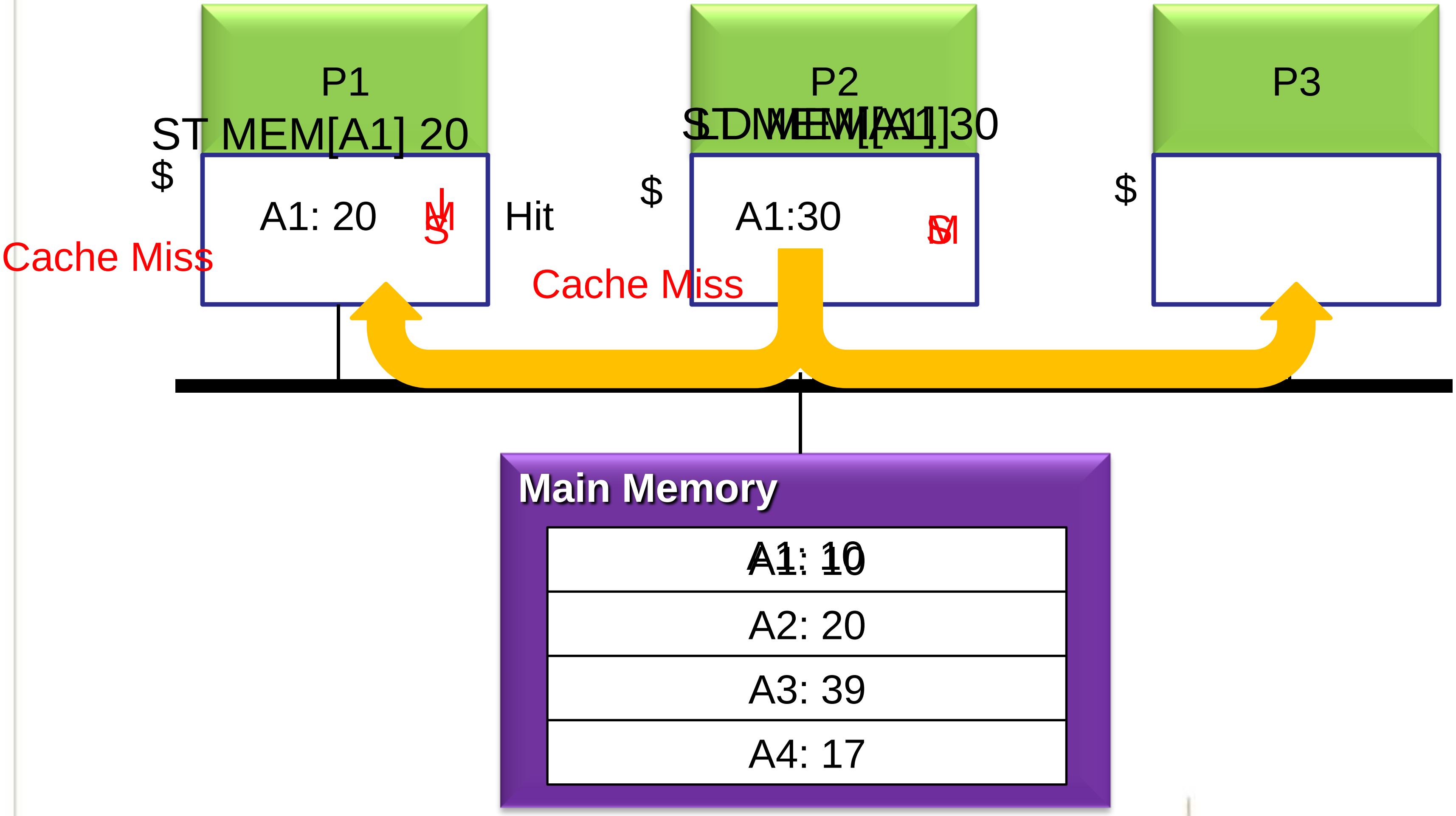


A1: 10
A2: 20
A3: 39
A4: 17

MSI Snoopy Protocol

- State of block B in cache C can be
  - Invalid: B is not cached in C
    - To read or write, must make a request on the bus
  - Modified: B is dirty in C
    - has the block, no other cache has the block,
      and C must update memory when it displaces B
    - Can read or write B without going to the bus
  - Shared: B is clean in C
    - C has the block, other caches have the block,
      and C need not update memory when it displaces B
    - Can read B without going to bus
    - ***To write, must send an upgrade request to the bus***

MSI Example


P1
ST MEM[A1] 20
$
A1: 20
I
M
S
Cache Miss
Hit
P2
ST MEM[A1] 30
LD MEM[A1]
$
A1:30
S
M
Cache Miss
P3
$
Main Memory
A1: 10
A1: 10
A2: 20
A3: 39
A4: 17