CS 201 System Programming and Architecture

Memory Hierarchy

Fang Song
Portland State University

Slides adapted from
CS205@PSU(Prof. Li) / CS15-213 @CMU

Recall: Writing & Reading Memory

- **Write ("Store")**
  - Transfer data from CPU to memory

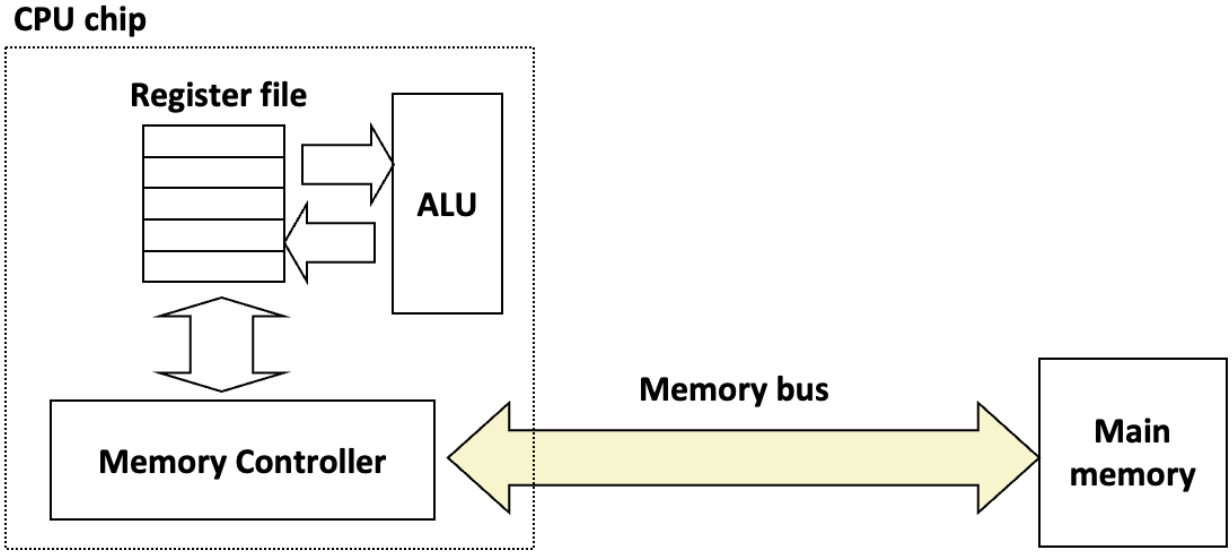
```
movq %rax, 8(%rsp)
```

- **Read ("Load")**
  - Transfer data from memory to CPU

```
movq 8(%rsp), %rax
```

Modern Connection between CPU & Memory

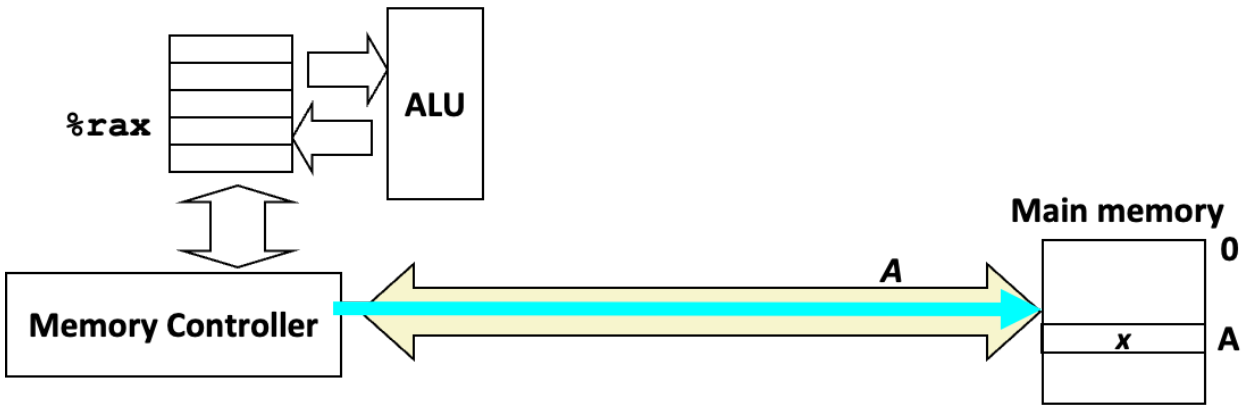
- Bus: a collection of parallel wires that connect hardware components and let them talk to each other.
- Memory bus: carries address, data, and control signals to perform memory operations (read/write).



Memory Read Transaction (1)

- CPU places address A on the memory bus.

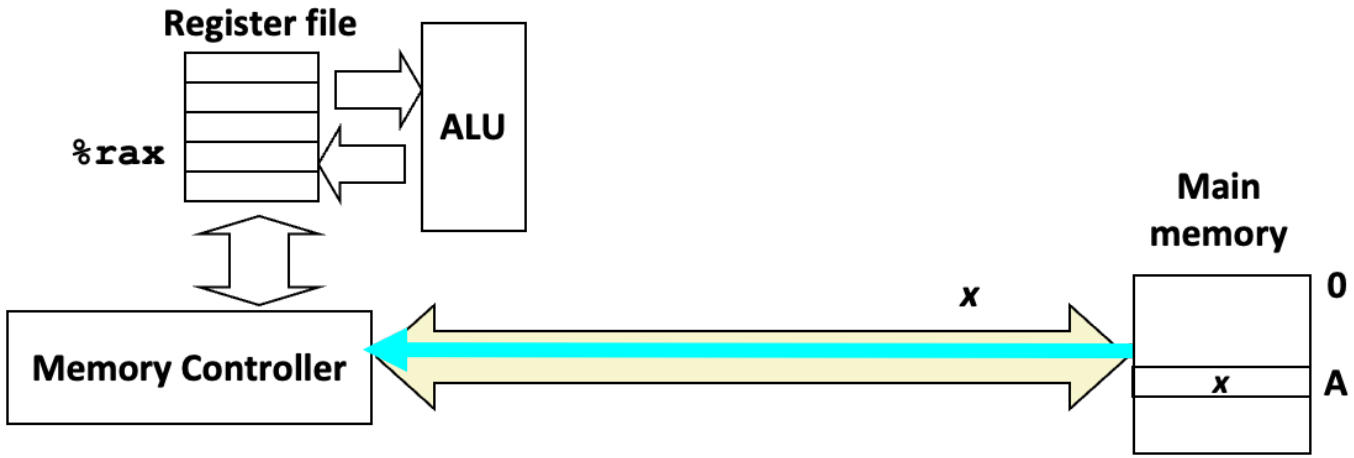
Load: `movq A, %rax`



Memory Read Transaction (2)

- Main memory reads A from the memory bus, retrieves word x, and places it on the bus.

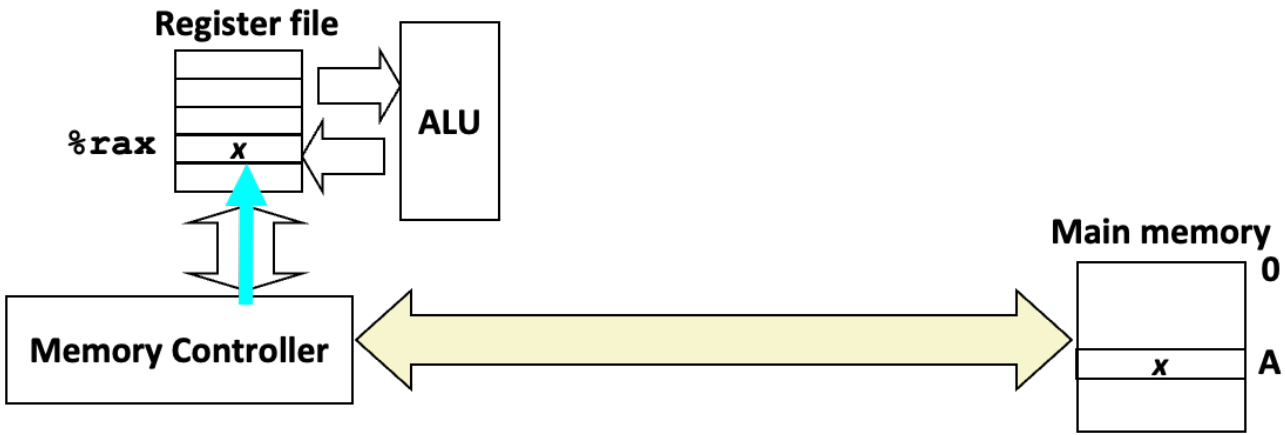
Load: `movq A, %rax`



Memory Read Transaction (3)

- CPU reads word x from the bus and copies it into register %rax.

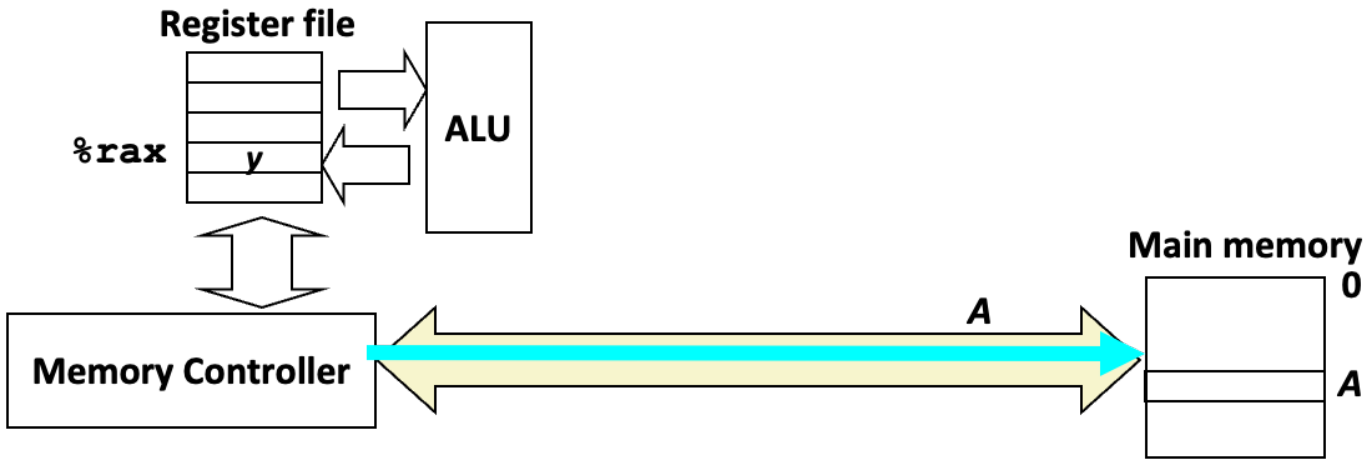
Load: `movq A, %rax`



Memory Write Transaction (1)

- CPU places address A on bus. Main memory reads it and waits for the corresponding data word to arrive.

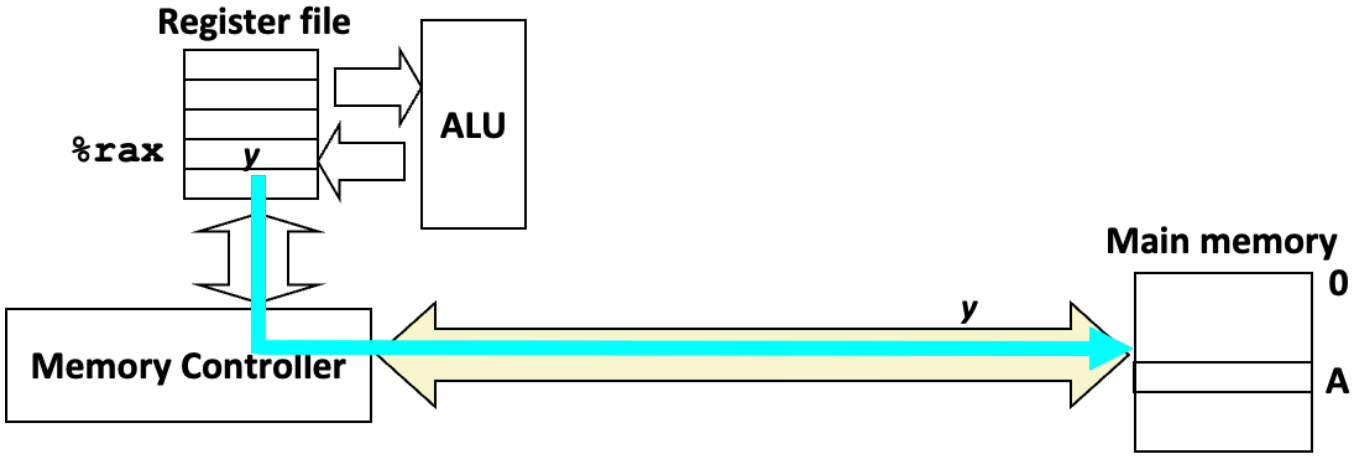
Store: `movq %rax, A`



Memory Write Transaction (2)

- CPU places data word y on the memory bus.

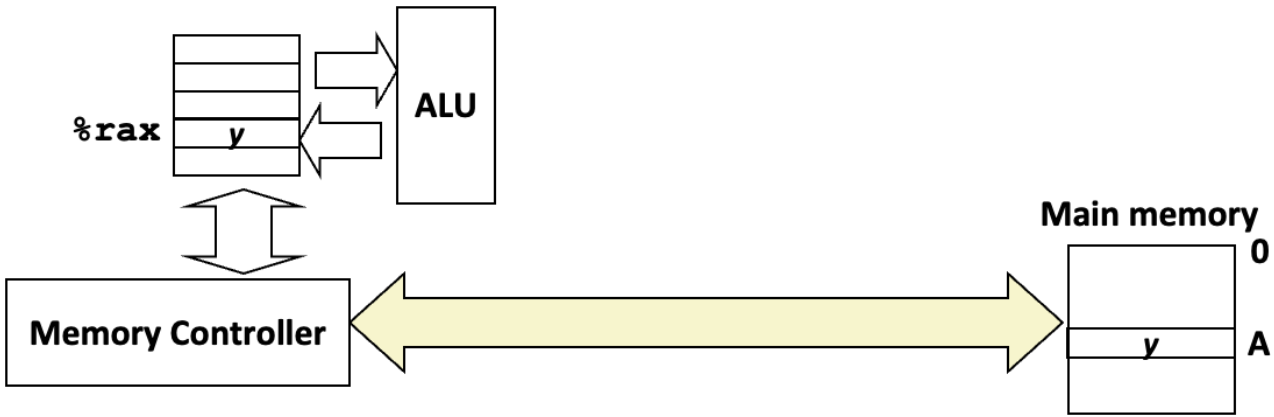
Store: `movq A, %rax`



Memory Write Transaction (3)

- Main memory reads data word y from the bus and stores it at address A.

Store: `movq A, %rax`



Today

- Memory abstraction
- RAM
- Locality
- Memory hierarchy

Radom-Access Memory (RAM)

Main memory building block

- **Key features**
  - Traditionally packaged as a chip (or embedded as part of processor chip).
  - Basic storage unit is normally a cell (one bit per cell).
  - System “main memory” comprises multiple RAM chips.

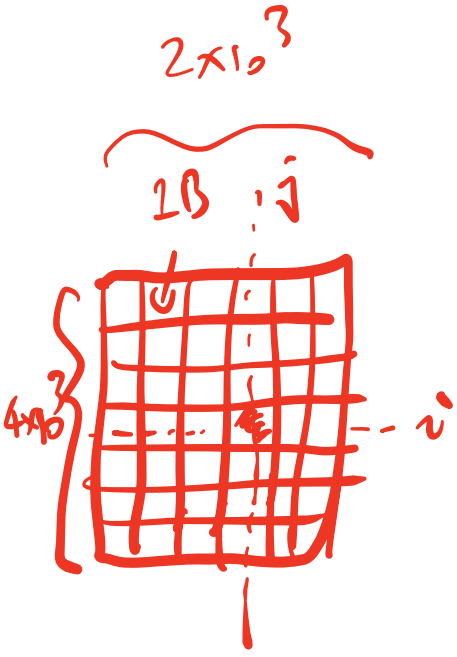
- **Two main varieties:**
  - SRAM (Static RAM): holds state indefinitely (but will still lose data on power loss)
  - DRAM (Dynamic RAM): must refresh state periodically

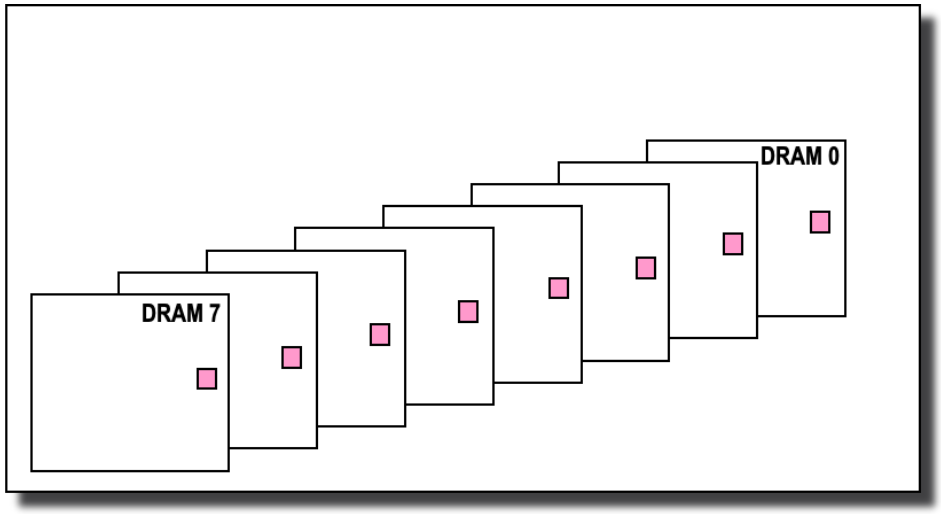
SRAM vs. DRAM

	Access time	Need refresh?	Needs EDC?	Cost	Apps
SRAM	1x	No	Maybe	100x	Cache memories
DRAM	10x	Yes	Yes	1x	Main memories

EDC: Error detection & correction

Memory modules



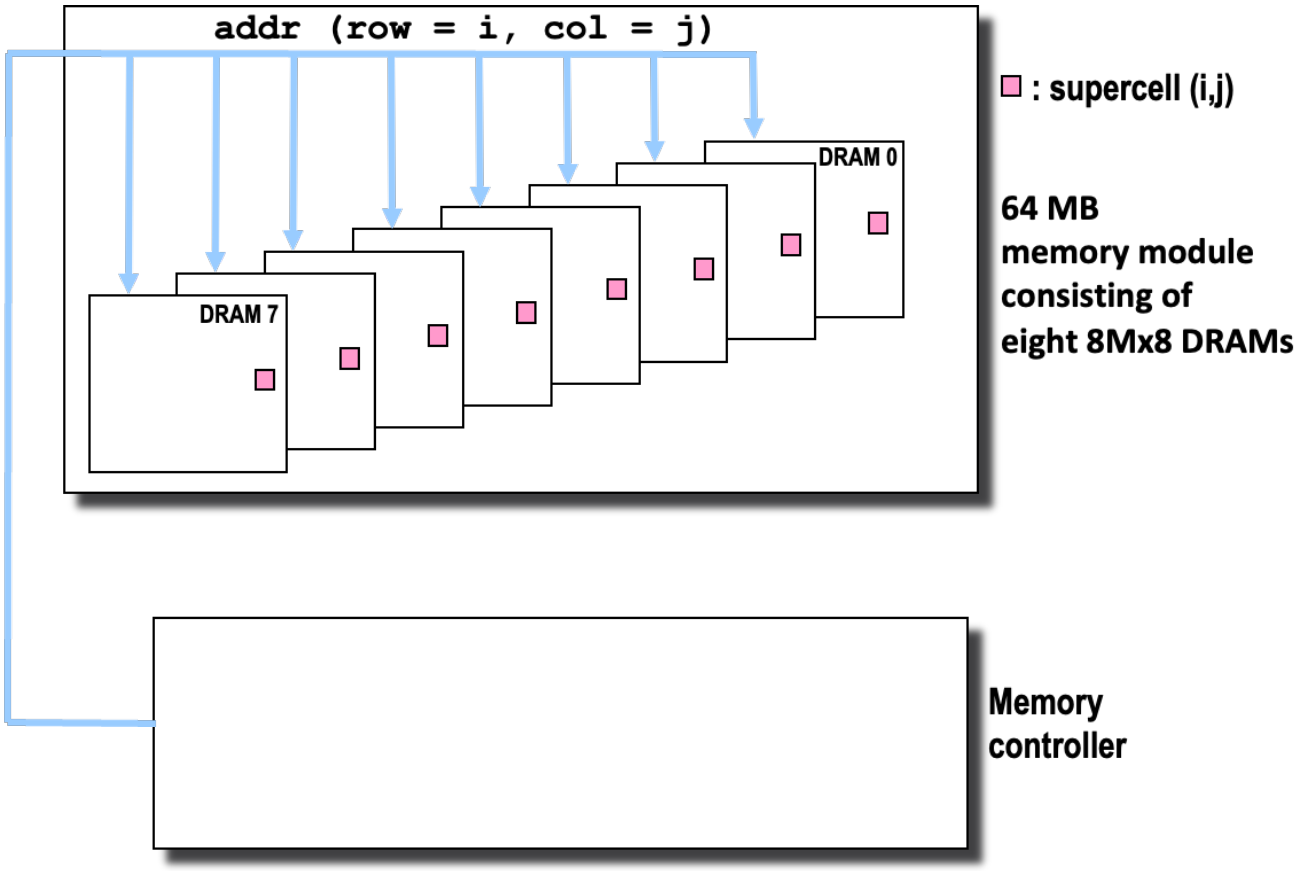


: supercell (i,j)

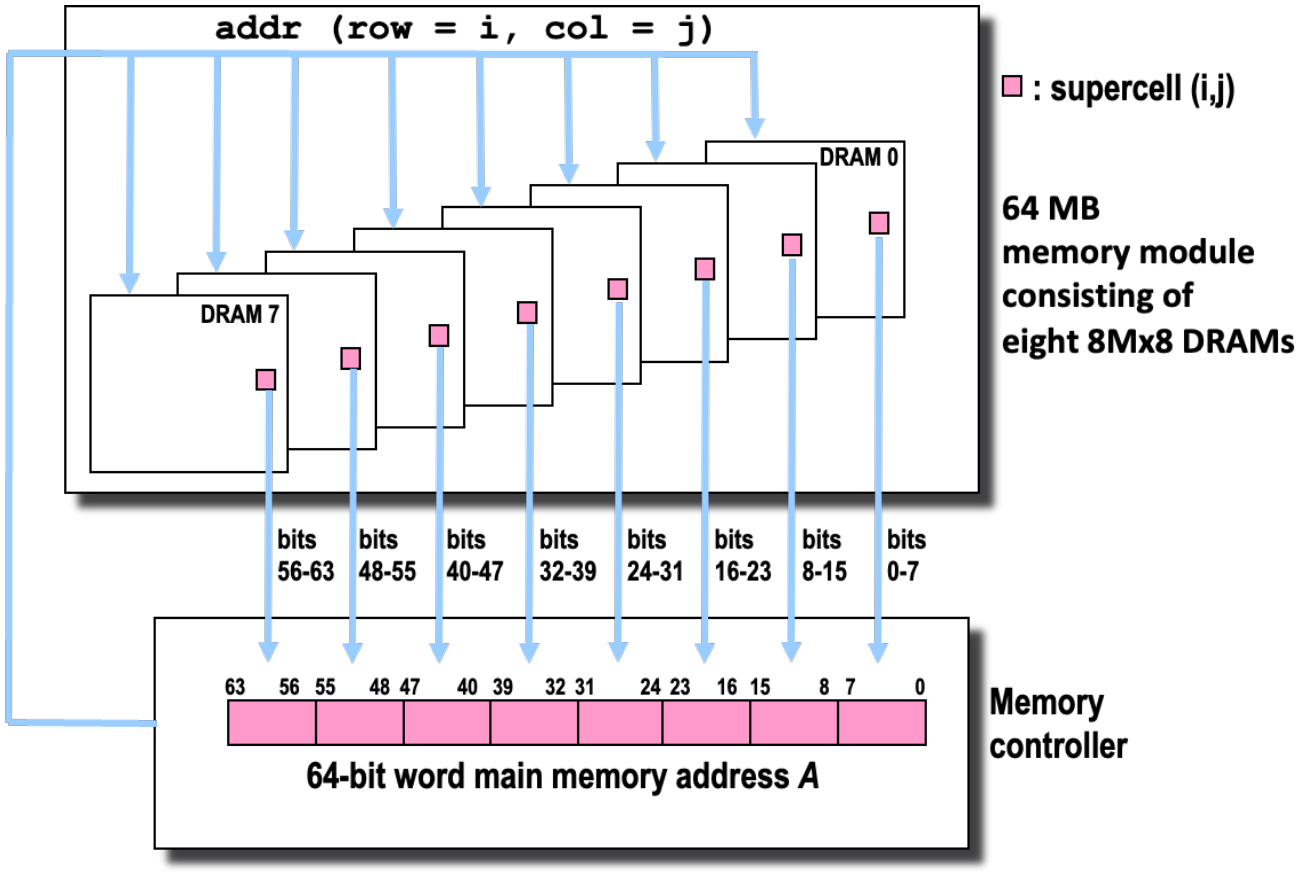
64 MB
memory module
consisting of
eight 8Mx8 DRAMs

Memory
controller

Memory Modules



Memory Modules

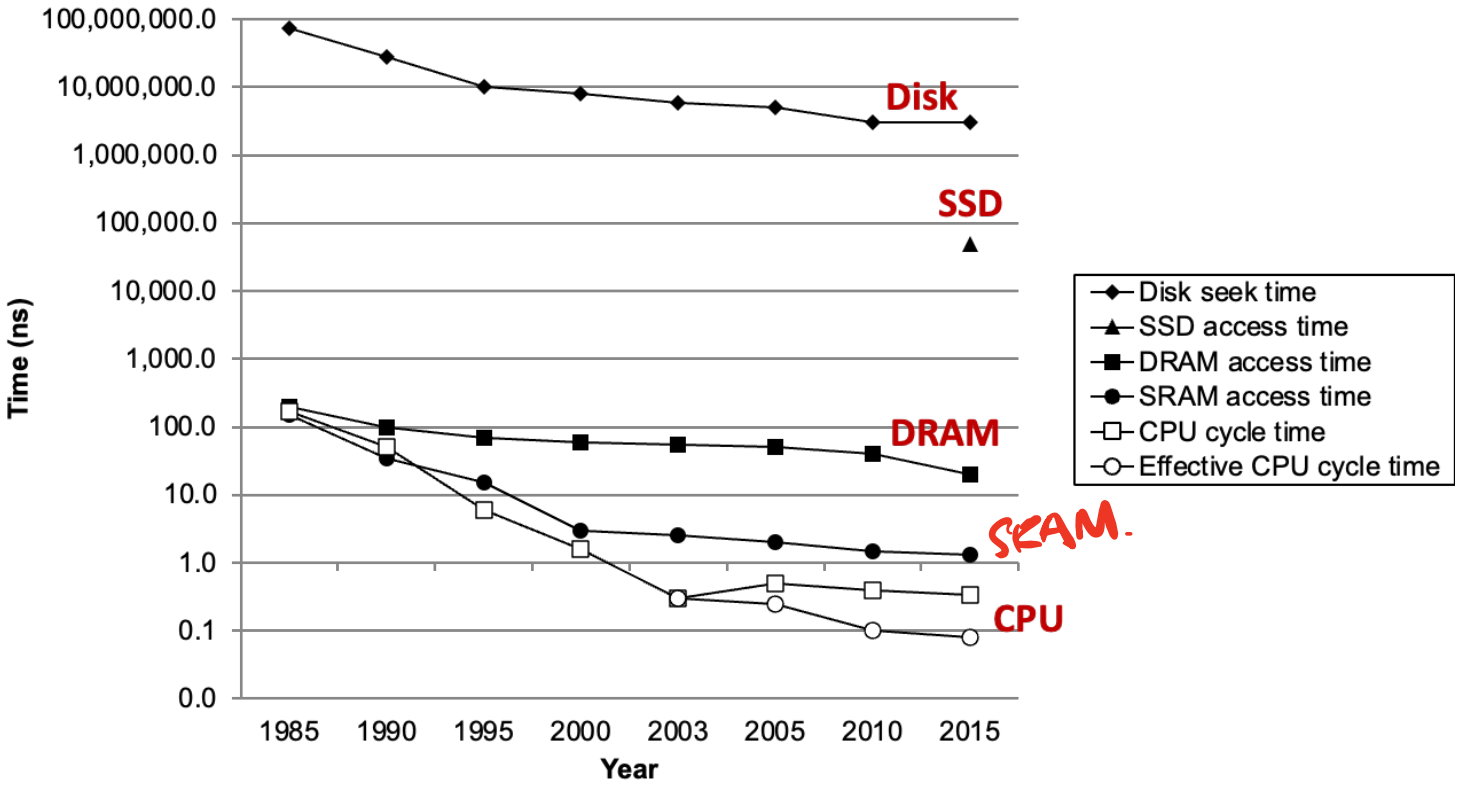


Today

- Memory abstraction
- RAM
- Locality
- Memory hierarchy

The CPU - Memory Gap

:(The gap widens between DRAM, disk, and CPU speeds.

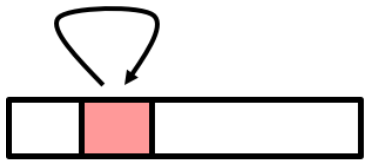


Locality is the Rescue

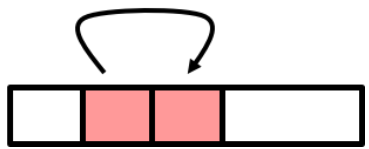
Principle of locality

Programs tend to use data and instructions with addresses near or equal to those they have used recently.

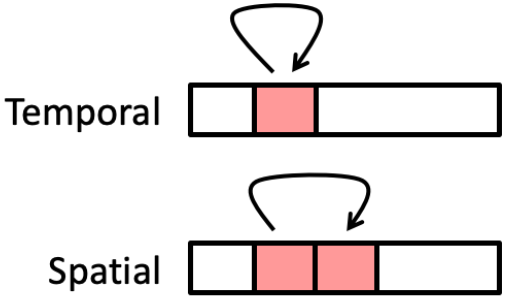
- **Temporal locality**
  - Recent referenced items are likely to be referenced again in the near future



- **Spatial locality**
  - Items with Nearby addresses tend to be referenced close together in time



Know Your Locality




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

? Spatial or Temporal locality

- **Data references**
  - Reference array elements in succession (stride-1 reference pattern).
  - Reference variable **sum** each iteration.
- **Instruction references**
  - Reference instructions in sequence.
  - Cycle through loop repeatedly.

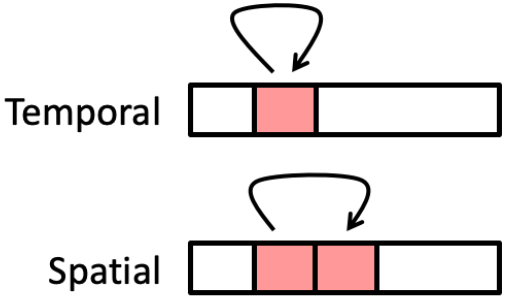
Spatial

Temporal

S

T.

Know Your Locality




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

? Spatial or Temporal locality

- **Data references**
  - Reference array elements in succession (stride-1 reference pattern). — **spatial**
  - Reference variable **sum** each iteration. — **temporal**
- **Instruction references**
  - Reference instructions in sequence. — **spatial**
  - Cycle through loop repeatedly. — **temporal**

Locality Exercise 1

Being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer.

- **Which has better locality with respect to array a?** Hint: array layout is row-major order

```
int sum_array_rows(int a[M][N])
{
  int i, j, sum = 0;

  for (i = 0; i < M; i++)
    for (j = 0; j < N; j++)
      sum += a[i][j];
  return sum;
}
```

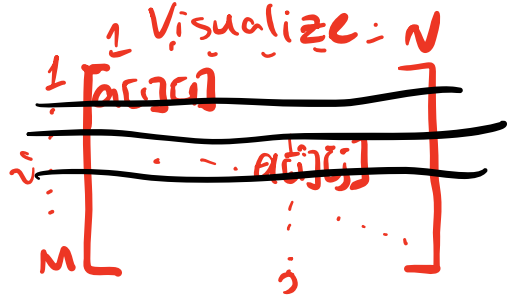
stride-1 ref.

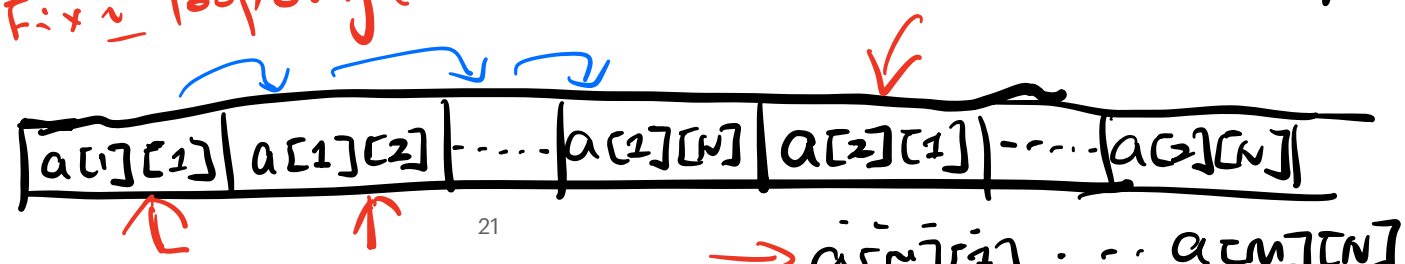
```
int sum_array_cols(int a[M][N])
{
  int i, j, sum = 0;

  for (j = 0; j < N; j++)
    for (i = 0; i < M; i++)
      sum += a[i][j];
  return sum;
}
```

stride-N ref.

(row) Fix i, loop on j (col)

(col) Fix j, loop on (i)

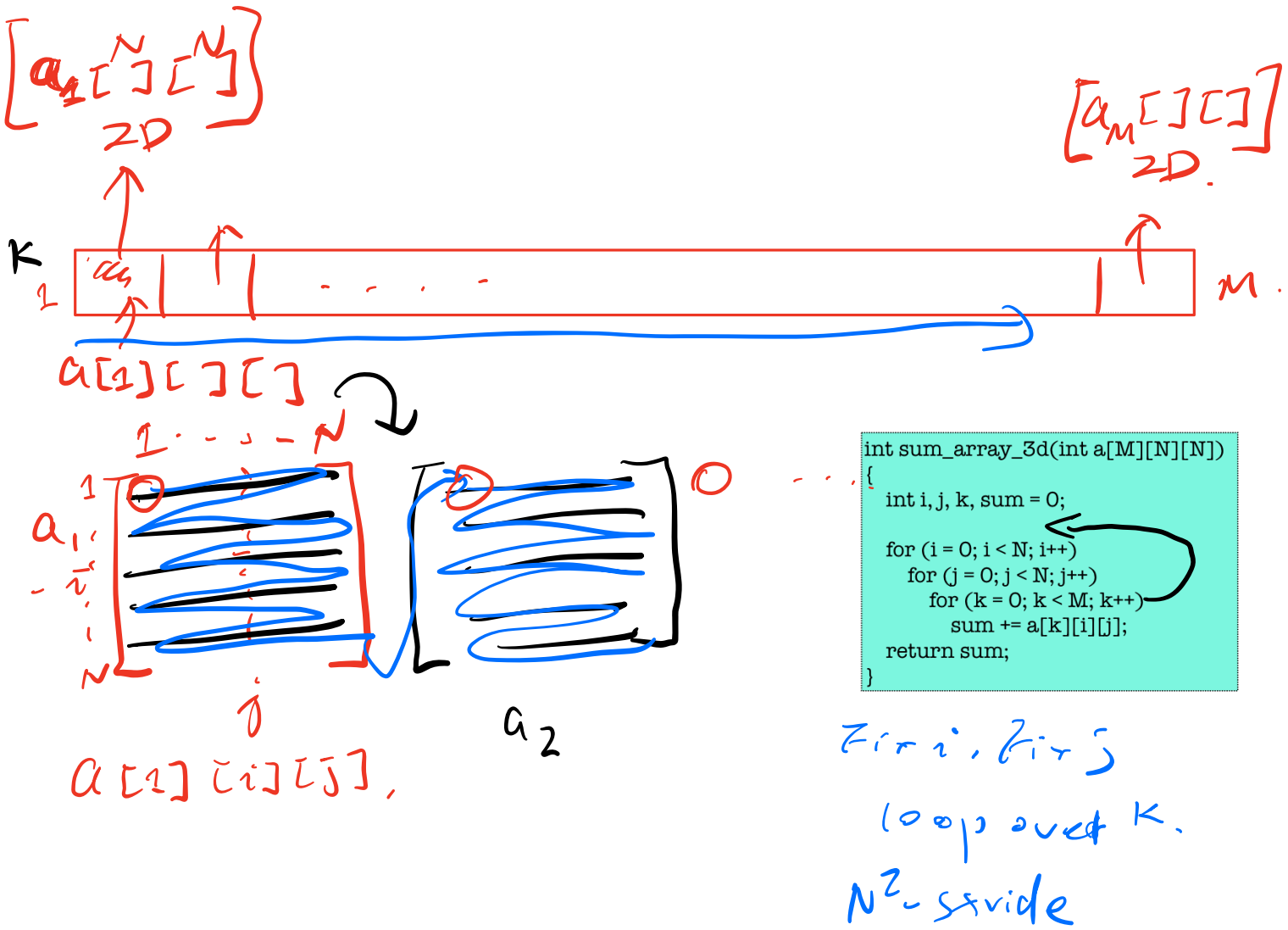




Locality Exercise 2

- Can you permute the loops so that the function scans the 3-d array **a** with a stride-1 reference pattern (and thus has good spatial locality)?

```
int sum_array_3d(int a[M][N][N])
{
   int i, j, k, sum = 0;

   for (i = 0; i < N; i++)
      for (j = 0; j < N; j++)
         for (k = 0; k < M; k++)
            sum += a[k][i][j];
   return sum;
}
```




```
int sum_array_3d(int a[M][N][N])
{
  int i, j, k, sum = 0;

  for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
      for (k = 0; k < M; k++)
        sum += a[k][i][j];
  return sum;
}
```

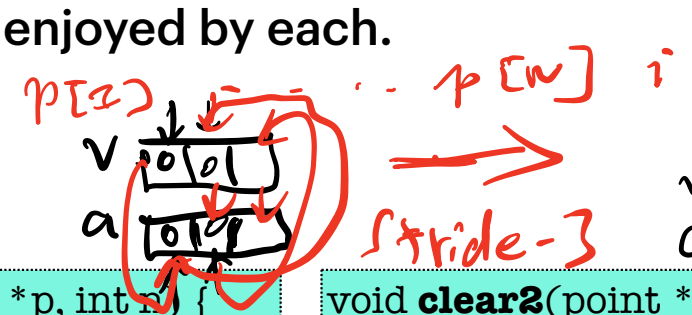
Fix i, fix j

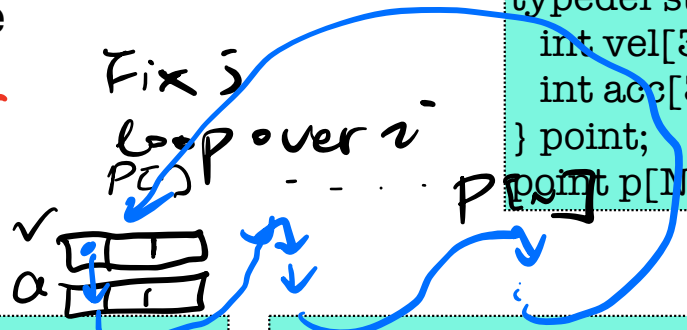
loop over K.

N^2 - side

Locality Exercise 3

- Rank order the functions with respect to the spatial locality enjoyed by each.






```
#define N 1000
typedef struct {
  int vel[3];
  int acc[3];
} point;
point p[N];
```

```
void clear1(point *p, int n) {
  int i,j;
  for (i=0; i<n ; i++) {
    for (j=0;j<3;j++)
 p[i].vel[j]=0;
    for (j=0; j<3 ; j++)
 p[i].acc[j]=0;
  }
}
```

B

```
void clear2(point *p, int n) {
  int i,j;
  for (i=0; i<n ; i++) {
    for (j=0; j<3 ; j++) {
 p[i].vel[j]=0;
 p[i].acc[j]=0;
    }
  }
}
```

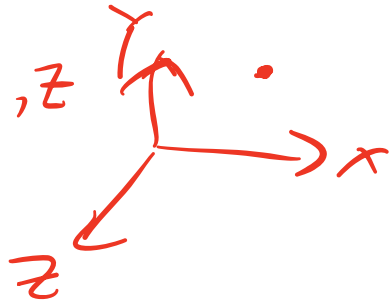
OK

```
void clear3(point *p, int n) {
  int i,j;
  for (j=0; j<3 ; j++) {
    for (i=0; i<n ; i++)
 p[i].vel[j]=0;
    for (i=0; i<n ; i++)
 p[i].acc[j]=0;
  }
}
```

W

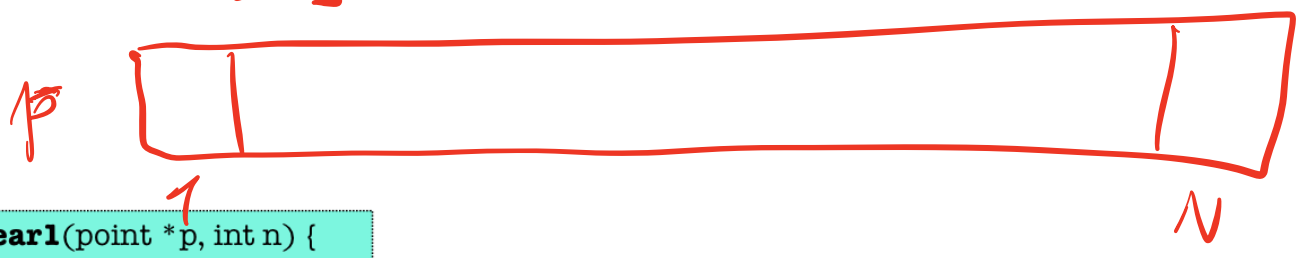
```
#define N 1000
typedef struct {
  int vel[3];
  int acc[3];
} point;
point p[N];
```

velocity along x, y, z



acc ...

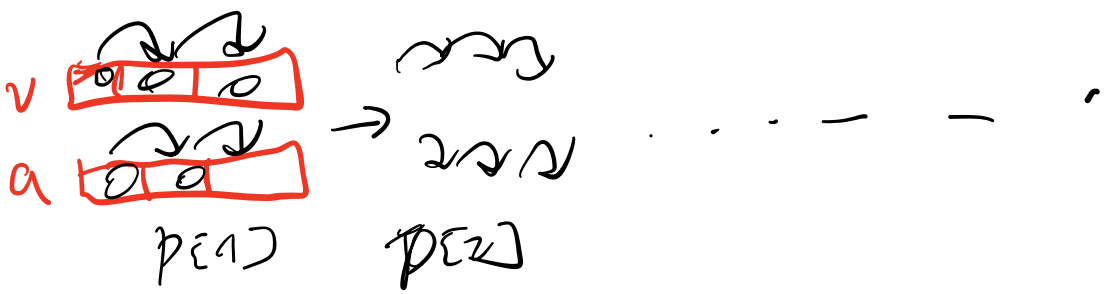
vel[3]
acc[3]




```
void clear1(point *p, int n) {
 int i,j;
 for (i=0; i<n ; i++) {
   for (j=0;j<3;j++)
p[i].vel[j]=0;
   for (j=0; j<3 ; j++)
p[i].acc[j]=0;
 }
}
```

stride-1 ref.

(i i p[1].



Today

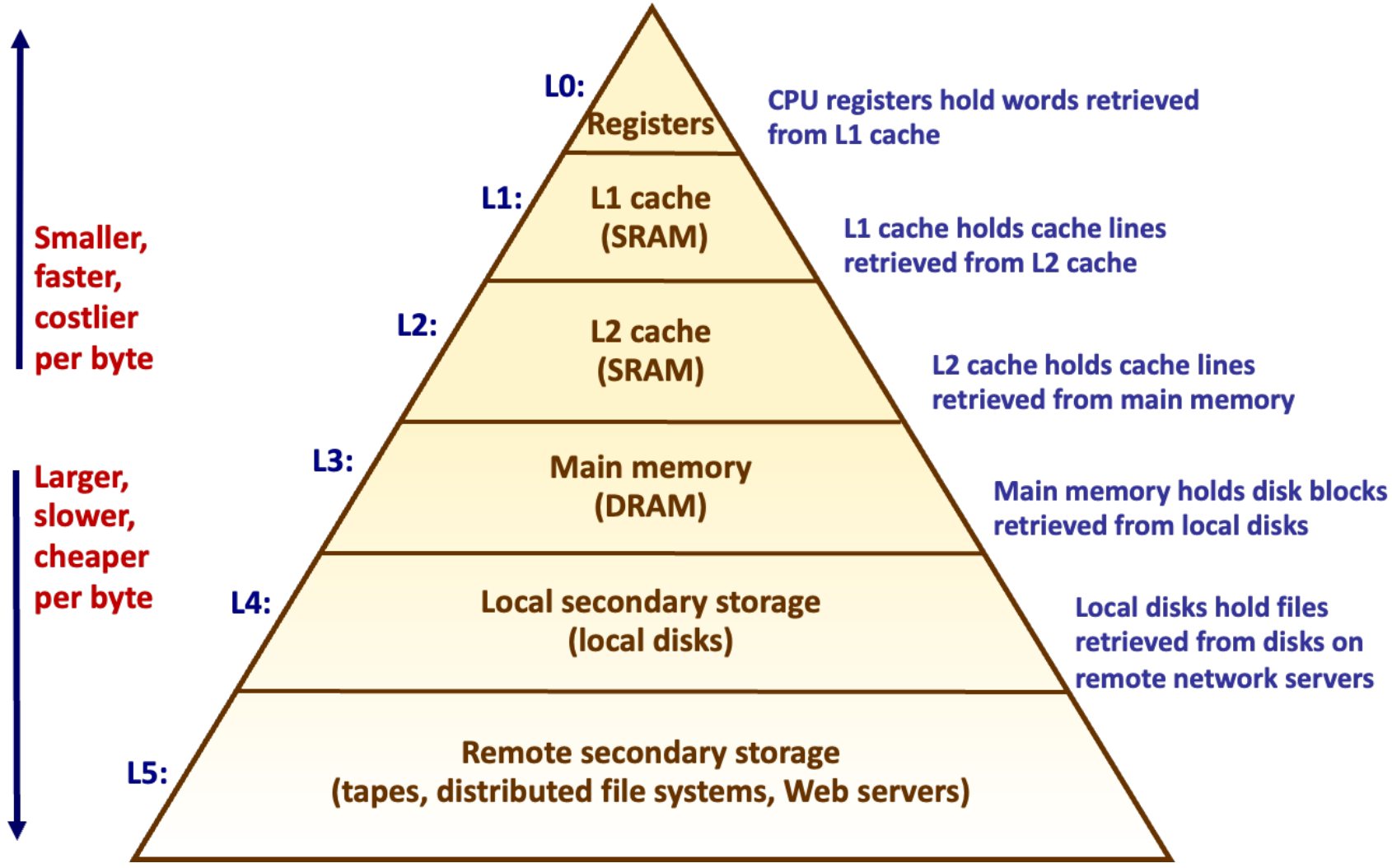
- Memory abstraction
- RAM
- Locality
- Memory hierarchy

The Lesson

◉ **Some fundamental and enduring properties of hardware and software:**

- Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
- The gap between CPU and main memory speed is widening.
- Well-written programs tend to exhibit good locality.

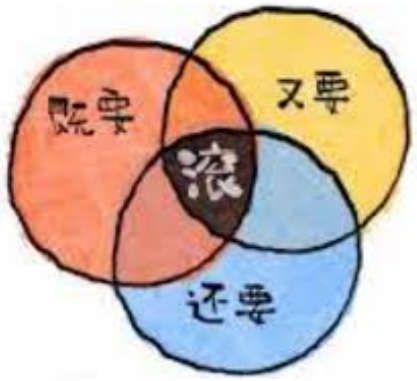
⇒ **Memory hierarchy as an approach to organizing memory and storage systems!**

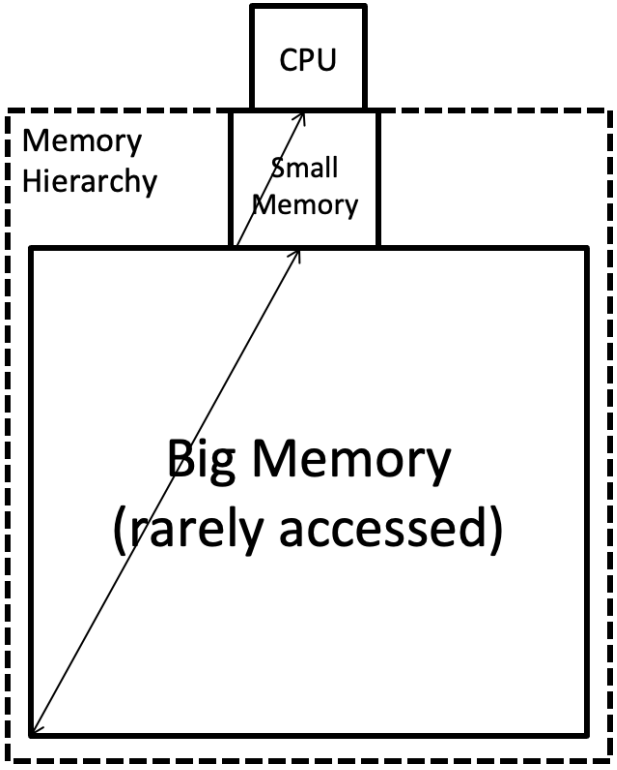

Smaller,
faster,
costlier
per byte
Larger,
slower,
cheaper
per byte
L0:
Registers
CPU registers hold words retrieved
from L1 cache
L1:
L1 cache
(SRAM)
L1 cache holds cache lines
retrieved from L2 cache
L2:
L2 cache
(SRAM)
L2 cache holds cache lines
retrieved from main memory
L3:
Main memory
(DRAM)
Main memory holds disk blocks
retrieved from local disks
L4:
Local secondary storage
(local disks)
Local disks hold files
retrieved from disks on
remote network servers
L5:
Remote secondary storage
(tapes, distributed file systems, Web servers)

Caches

- **A smaller, faster memory device that acts as a staging area for a subset of the data in a larger, slower device.**
- **For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1.**
  - Typically, caches refer to fast SRAM-based memories managed automatically in hardware.
  - Hold frequently accessed blocks of main memory.
  - CPU looks first for data in L1, then in L2, then in main memory.
- **Blessed by locality: programs tend to access the data at level k more often than they access the data at level k+1.**

Hierarchy: offers the illusion of large, cost-effective & fast memory





Scratch

Supplement