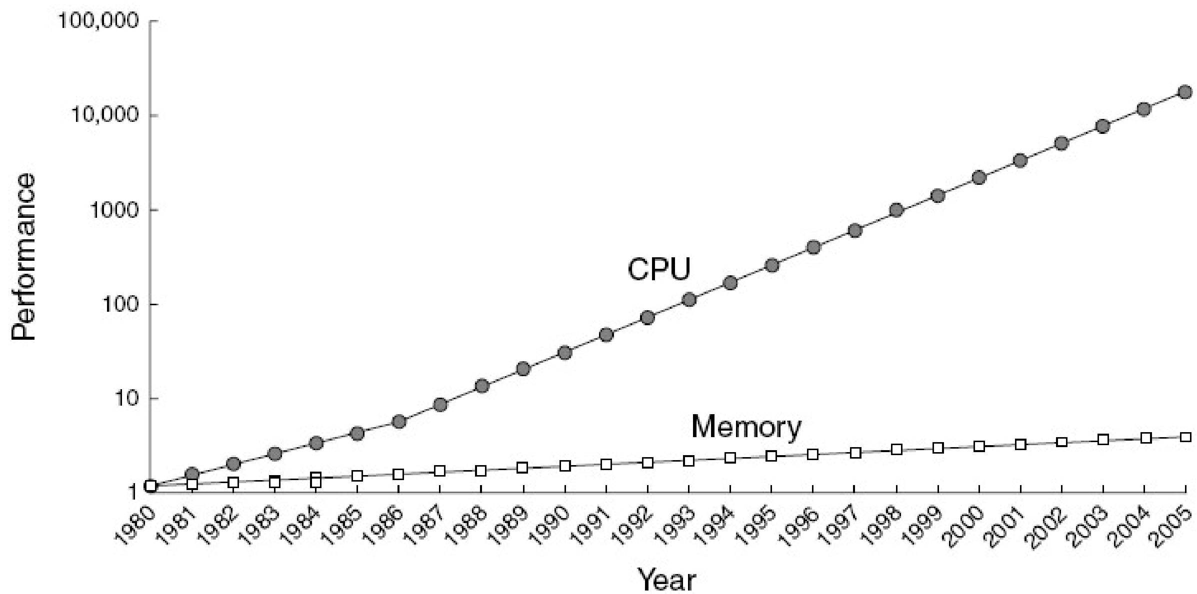
CISC260 Machine Organization and Assembly Language

Memory hierarchy

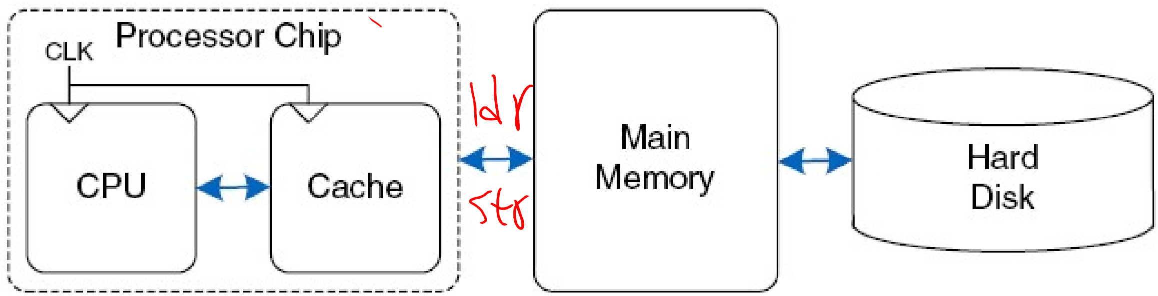
And

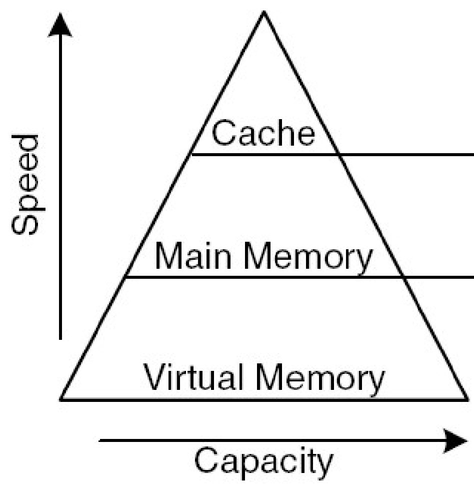
Cache


100,000
10,000
1000
100
10
1
Performance
CPU
Memory
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
Year

- Fact: Large memories are slow and fast memories are small

- How do we create a memory that gives the illusion of being large, cheap and fast (most of the time)?
  - With hierarchy
  - With parallelism





Technology	Cost/GB	Access Time
SRAM	~ $10,000	~ 1 ns
DRAM	~ $100	~ 100 ns
Hard Disk	~ $1	~ 10,000,000 ns

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 Example:

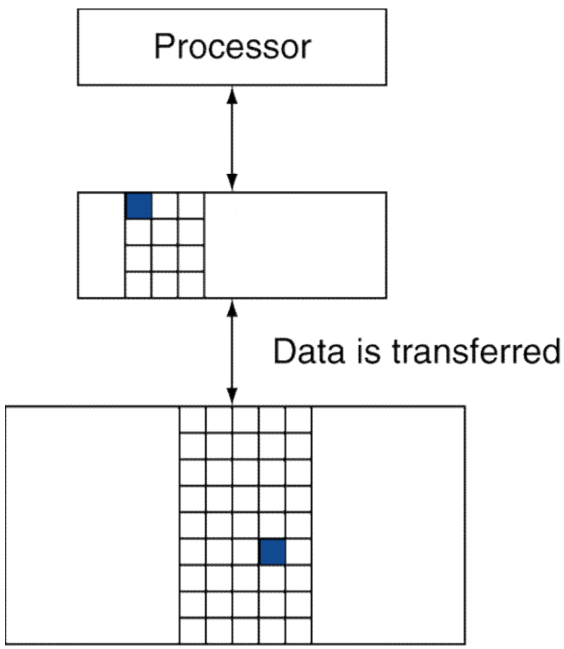
```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- **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**

Taking Advantage of Locality

- Memory hierarchy
- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
  - Main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory attached to CPU

Memory Hierarchy Levels



- Block (aka line): unit of copying
  - May be multiple words
- If accessed data is present in next level
  - Hit: access satisfied by next level
    - Hit ratio: hits/accesses
- If accessed data is absent
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss ratio: misses/accesses
      = 1 – hit ratio
  - Then accessed data supplied from next level

- Two questions to answer (in hardware):

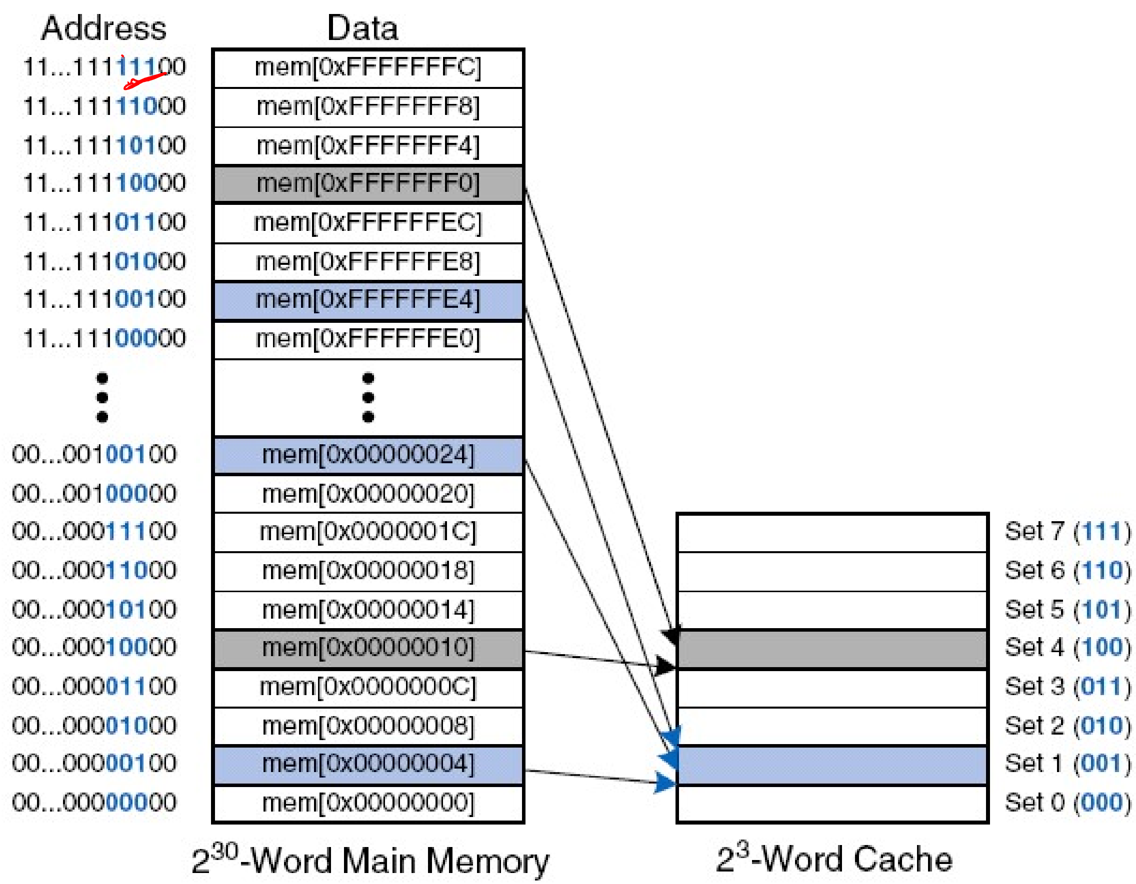
  Q1: How do we know if a data item is in the cache?

  Q2: If it is, how do we find it?

- Direct mapped

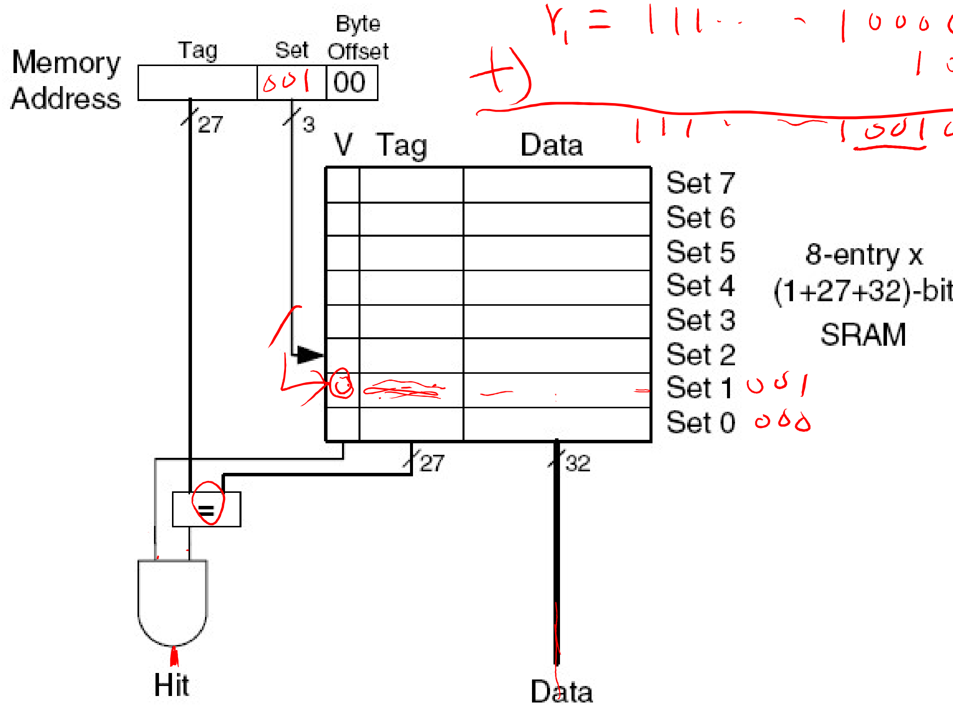
  For each item of data at the lower level, there is exactly one location in the cache where it might be - so lots of items at the lower level must share locations in the upper level

Direct mapped

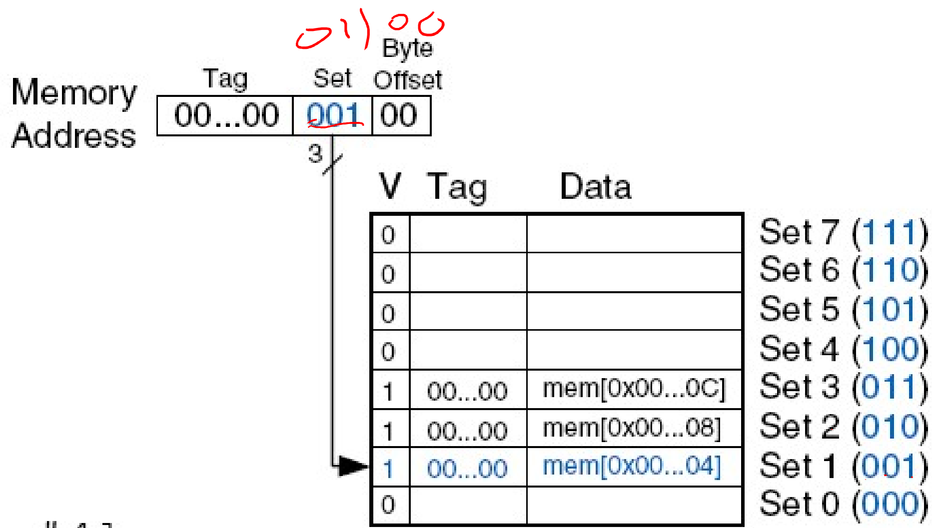


LDR r3, [r2, #4]

r1 = 111... 1000000
+) 100
111... 1000100



Example: run the code on 8-sets one-word block cache which is initially empty

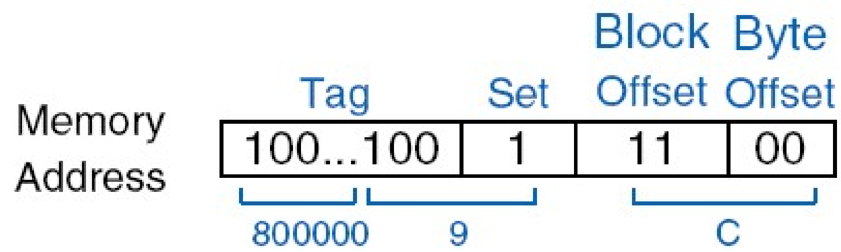


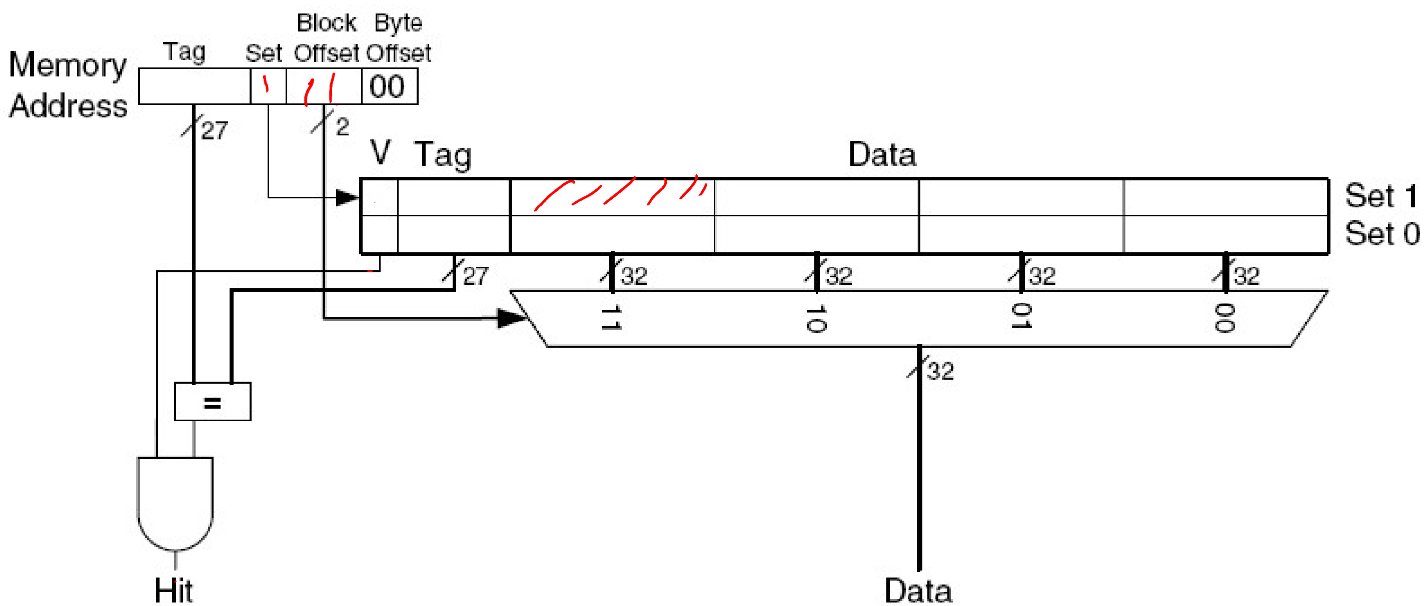

```
        MOV R0, #5
        MOV R1, #0
LOOP    CMP R0, #0
        BEQ DONE
        LDR R2, [R1, #4]
        LDR R3, [R1, #12]
        LDR R4, [R1, #8]
        SUB R0, R0, #1
        B   LOOP
DONE
```

What locality is being used?

A. Spatial; B. Temporal; C. Both; D. None

What is the miss rate? $\frac{3}{15} = \frac{1}{5}$


Memory Address
Tag
Set
Block Offset
Byte Offset
100...100
1
11
00
800000
9
C


Memory Address
Tag
Set
Block Offset
Byte Offset
00
27
2
V
Tag
Data
Set 1
Set 0
27
32
32
32
32
11
10
01
00
32
=
Hit
Data

Example: run the code on 2-sets four-word block cache which is initially empty

```
        MOV R0, #5
        MOV R1, #0
LOOP    CMP R0, 0
        BEQ DONE
        LDR R2, [R1, #4]
        LDR R3, [R1, #12]
        LDR R4, [R1, #8]
        SUB R0, R0, #1
        B   LOOP
DONE
```

miss

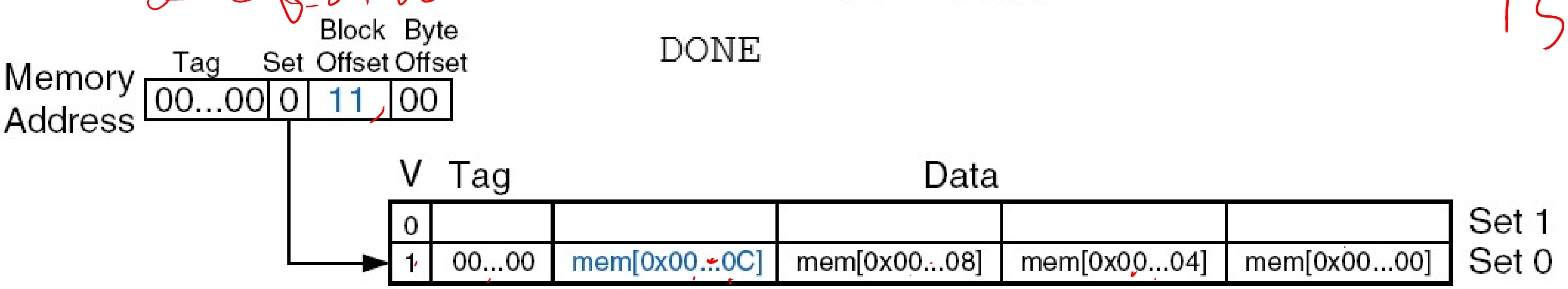
$\frac{1}{15}$

What locality is being used?

A. Spatial; A. Temporal; C. Both; D. None

What is the miss rate?

0 - 0 1 0 0



Memory and Cache performance

$$Miss\ Rate = \frac{Number\ of\ misses}{Number\ of\ total\ memory\ accesses} = 1 - Hit\ Rate$$

$$Hit\ Rate = \frac{Number\ of\ hits}{Number\ of\ total\ memory\ accesses} = 1 - Miss\ Rate$$

Suppose a program has 2000 data access instructions (loads or stores), and 1250 of these requested data values are found in the cache. The other 750 data values are supplied to the processor by main memory or disk memory. What are the miss and hit rates for the cache?

Solution: The miss rate is $750/2000 = 0.375 = 37.5\%$. The hit rate is $1250/2000 = 0.625 = 1 - 0.375 = 62.5\%$.

Suppose a computer system has a memory organization with only two levels of hierarchy, a cache and main memory, with access times and miss rates given as follows:

	Access Time (Cycles)	Miss rate
Cache	1	10%
Main Memory	100	0%

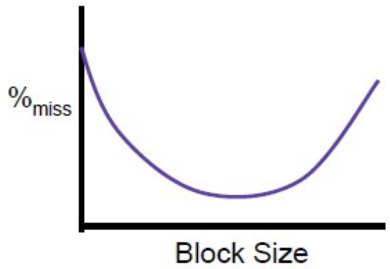
What is the average memory access time ?

Solution: The average memory access time is 1 + 0.1 (100) = 11 cycles.

CPU time = IC × CPI × CC

Block Size Considerations

- Larger blocks should reduce miss rate
  – Due to spatial locality
- But in a fixed-sized cache
  – Larger blocks ⇒ fewer of them
    - More competition ⇒ increased miss rate
  – Larger blocks ⇒ pollution
- Larger miss penalty
  – Can override benefit of reduced miss rate
  – Early restart and critical-word-first can help



Memory Performance Impact on Performance

- Suppose a processor executes at
  - ideal CPI = 1.1
  - 50% arith/logic, 30% ld/st, 20% control

  and that 10% of data
  memory operations miss with a 50 cycle miss penalty

- CPI = ideal CPI + average stalls per instruction =
  1.1(cycle) + (0.30 (datamemops/instr)
  x 0.10 (miss/datamemop) x 50 (cycle/miss))
  = 1.1 cycle + 1.5 cycle = 2.6

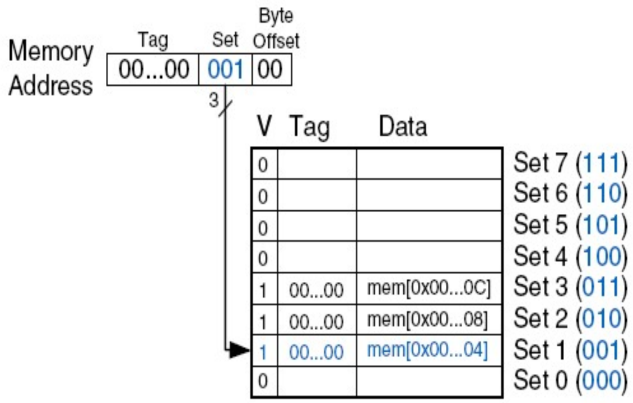
  so 58% of the time the processor is stalled waiting for memory!

- A 1% instruction miss rate would add an *additional* 0.5 to the CPI!

Cache block conflict

e.g., Run this code in a machine with a 8-word cache

```
        MOV R0, #5
        MOV R1, #0
LOOP    CMP R0, #0
        BEQ DONE
        LDR R2, [R1, #0x4]
        LDR R3, [R1, #0x24]
        SUB R0, R0, #1
        B   LOOP
DONE
```



What is the miss rate?

In a Row-major arrangement, for example, the elements for a 3x3 matrix A are stored in memory like

A[0,0], A[0, 1], A[0, 2], A[1, 0], A[1, 1], A[1, 2], A[2, 0], A[2, 1], A[2, 2]

In a Column-major arrangement, for example, the elements for a 3x3 matrix A are stored in memory like

A[0,0], A[1, 0], A[2, 0], A[0, 1], A[1, 1], A[2, 1], A[0, 2], A[1, 2], A[2, 2]

Therefore, the order of looping through the matrix indices is essential for spatial locality.

```
Sum = 0;
for(i=0; i<2; i++)
        for(j=0; j<2; j++)
                sum += A[i,j];
```

Locality

```
Sum = 0;
for(j=0; j<2; j++)
        for(i=0; i<2; i++)
                sum += A[i,j];
```

No locality