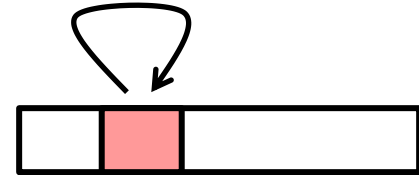# The Memory Hierarchy:
Locality of reference

Locality

- **Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently**

Locality

- **Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently**

- **Temporal locality:**
  - Recently referenced items are likely to be referenced again in the near future
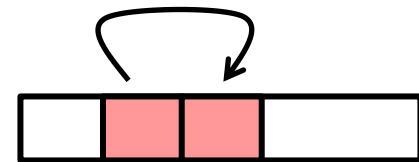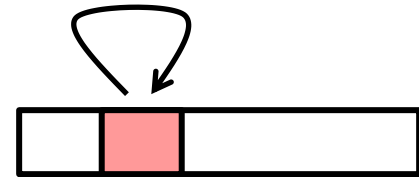
Locality

- **Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently**

- **Temporal locality:**
  - Recently 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

Prof. Michaël Cadilhac (based on slides by Bryant and O'Hallaron, CMU)

4

Locality Example

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

- **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.

Locality Example

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

- **Data references**
  - Reference array elements in succession (stride-1 reference pattern).
  - Reference variable `sum` each iteration.

Spatial locality

- **Instruction references**
  - Reference instructions in sequence.
  - Cycle through loop repeatedly.

Locality Example

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

■ Data references

- ▪ Reference array elements in succession (stride-1 reference pattern).

- ▪ Reference variable `sum` each iteration.

Spatial locality

Temporal locality

■ Instruction references

- ▪ Reference instructions in sequence.

- ▪ Cycle through loop repeatedly.

Locality Example

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

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

Locality Example

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

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

Qualitative Estimates of Locality

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

- **Question:** Does this function have good locality with respect to array a?

```c
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;
}
```

Locality Example

- **Question:** Does this function have good locality with respect to array `a`?

```
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;
}
```

Locality Example

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

Memory Hierarchies

- **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.

- **These fundamental properties complement each other beautifully.**

- **They suggest an approach for organizing memory and storage systems known as a memory hierarchy.**

Prof. Michaël Cadilhac (based on slides by Bryant and O'Hallaron, CMU)

13