In Lecture 4 ..

- There is a significant speed disparity between processor and main memory
- Cache memory: Organization, Address lookup
  - Direct mapped 16KB cache, Block size 32B

Tag	Index	Offset
18 bits	9 bits	5 bits

- Impact on program: Examples
  - Vector sum reduction
  - Vector dot product
  - DAXPY

Example 3: DAXPY

- Double precision Y = aX + Y, where X and Y are vectors and a is a scalar

  double X[2048], Y[2048], a;

  for (i=0; i<2048;i++) Y[i] = a*X[i]+Y[i];

- Reference sequence
  - load X[0] load Y[0] store Y[0] load X[1] load Y[1] store Y[1] ...
- Hits and misses: Assuming that base addresses of X and Y don't conflict in cache, hit ratio of 83.3%

Example 4: 2-d Matrix Sum

```
double A[1024][1024], B[1024][1024];
for (j=0;j<1024;j++)
for (i=0;i<1024;i++)
  B[i][j] = A[i][j] + B[i][j];
```

- Reference Sequence:

  load A[0,0] load B[0,0] store B[0,0]

  load A[1,0] load B[1,0] store B[1,0] **…**

- Question: In what order are the elements of a multidimensional array stored in memory?

Storage of Multi-dimensional Arrays

- Row major order
  - Example: for a 2-dimensional array, the elements of the first row of the array are followed by those of the 2nd row of the array, then the 3rd row, and so on
  - This is what is used in C
- Column major order
  - A 2-dimensional array is stored column by column in memory
  - Used in FORTRAN

Example 4: 2-d Matrix Sum

```
double A[1024][1024], B[1024][1024];
for (j=0;j<1024;j++)
for (i=0;i<1024;i++)
  B[i][j] = A[i][j] + B[i][j];
```

- Reference Sequence:

  load A[0,0] load B[0,0] store B[0,0]

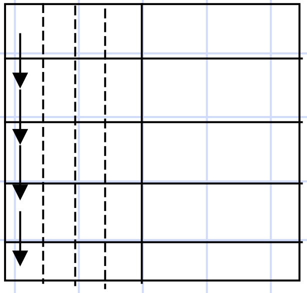
  load A[1,0] load B[1,0] store B[1,0] **...**

- Question: In what order are the elements of a multidimensional array stored in memory?

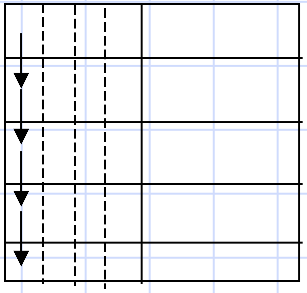
Example 4: Hits and Misses

- Reference order and storage order for our arrays are not the same
- Our loop will show no spatial locality
  - Assume that packing has been done to eliminate conflict misses due to base addresses
  - Miss(cold), Miss(cold), Hit for each array element
  - Hit ratio: 33.3%
  - Question: Will A[0,1] be in the cache when required later in the loop?

A



B



Example 4 with Loop Interchange

```
double A[1024][1024], B[1024][1024];
for (i=0;i<1024;i++)
for (j=0;j<1024;j++)
   B[i][j] = A[i][j] + B[i][j];
```

- Reference Sequence:

load A[0,0] load B[0,0] store B[0,0]

load A[0,1] load B[0,1] store B[0,1]

Hit ratio: 83.3%

Is Loop Interchange Always Safe?

```
for (j=1; j<2048; j++)
  for (i=1; i<2048; i++)
    A[i][j] = A[i+1][j-1] + A[i][j-1];
```

A[1,1] = A[2,0]+A[1,0]

A[2,1] = A[3,0]+A[2,0]

…

A[1,2] = A[2,1]+A[1,1]

Is Loop Interchange Always Safe?

```
for (i=1; i<2048; i++)   / interchanged
    for (j=1; j<2048; j++)
    A[i][j] = A[i+1][j-1] + A[i][j-1];
```

NO!

```
A[1,1] = A[2,0]+A[1,0]
A[2,1] = A[3,0]+A[2,0]
…
A[1,2] = A[2,1]+A[1,1]
```

```
A[1,1] = A[2,0]+A[1,0]
A[1,2] = A[2,1]+A[1,1]
…
A[2,1] = A[3,0]+A[2,0]
```

Is Loop Interchange Always Safe?

```
for (i=2047; i>1; i--)
~~for (i=1; i<2048; i++)~~
    for (j=1; j<2048; j++)
        A[i][j] = A[i+1][j-1] + A[i][j-1];
```

A[1,1] = A[2,0]+A[1,0]

A[2,1] = A[3,0]+A[2,0]

…

A[1,2] = A[2,1]+A[1,1]

A[1,1] = A[2,0]+A[1,0]

A[1,2] = A[2,1]+A[1,1]

…

A[2,1] = A[3,0]+A[2,0]