Writing Cache-friendly Code

Adalbert **Gerald** Soosai Raj
CS 354 - Section 2
UW-Madison

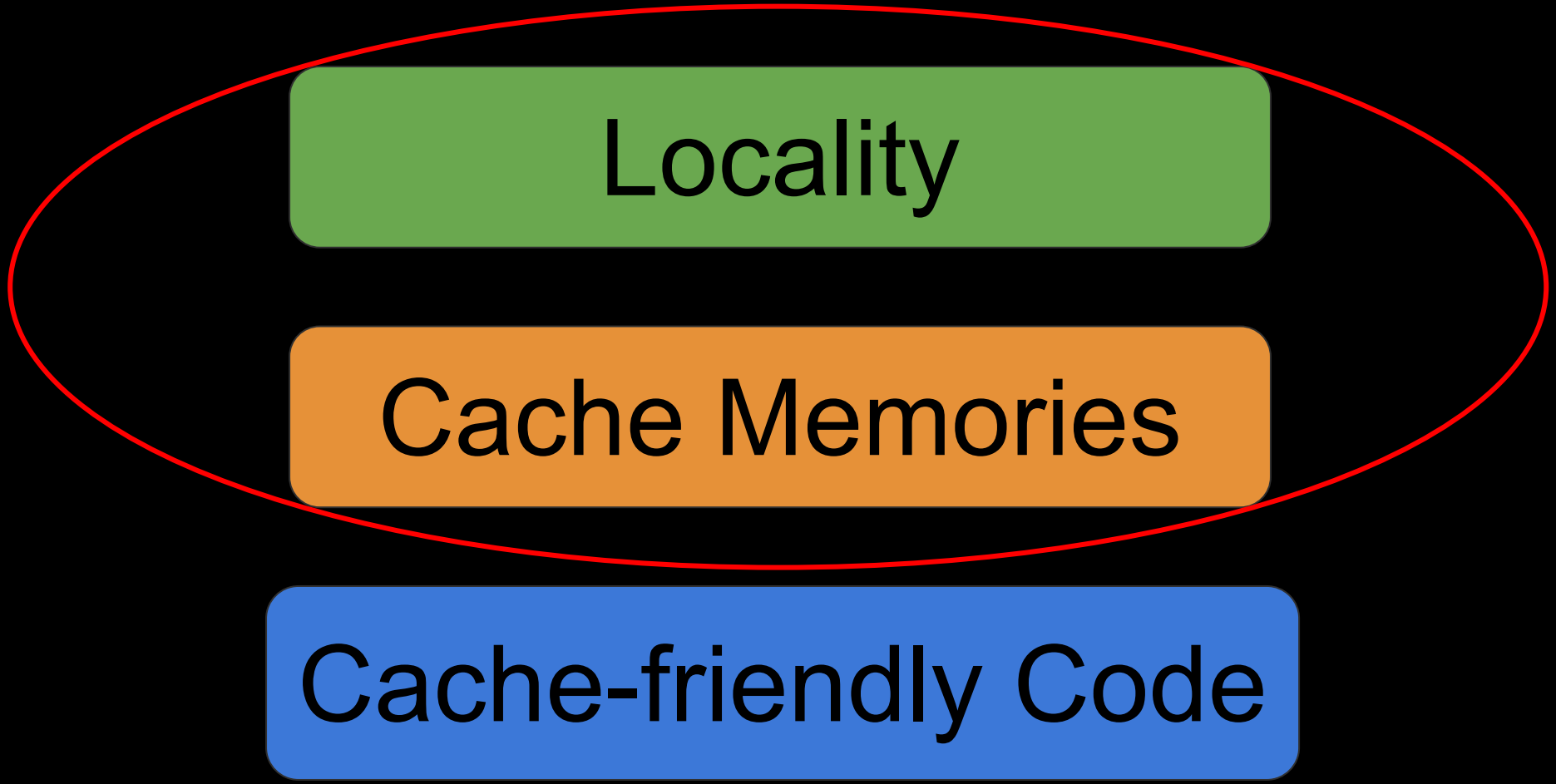
Locality

Locality

Cache Memories

Locality

Cache Memories


Locality
Cache Memories
Cache-friendly Code

Better locality => Lower cache miss rates

Better locality => Lower cache miss rates

Lower cache miss rates => Faster programs

Approach to write cache-friendly code

Approach to write cache-friendly code

Make the common case go fast.

Approach to write cache-friendly code

Make the common case go fast.

Minimize the number of cache misses in each inner loop.

Sum the elements of an array

```
int sum_array(int a[], int n)
{
    int i, sum = 0;

    for(i = 0; i < n; i++) {
        sum += a[i];
    }

    return sum;
}
```

Cache Properties

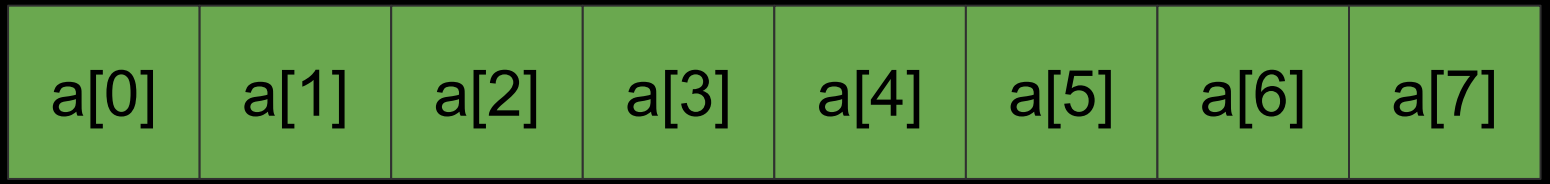
1 block = 4 words

1 word = 4 bytes

1 block = 16 bytes

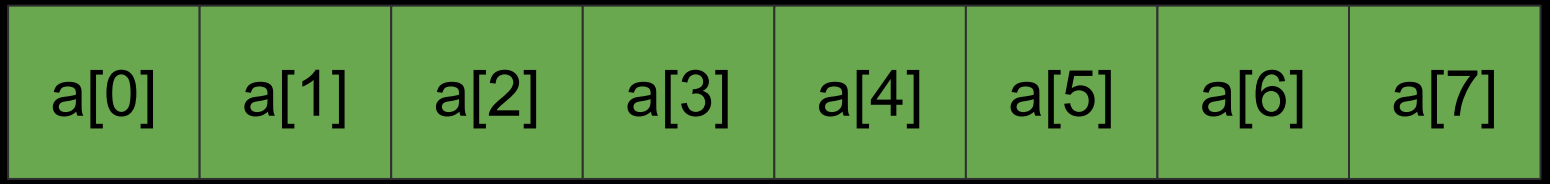
Size of the array in bytes

sizeof(int) = 4 bytes



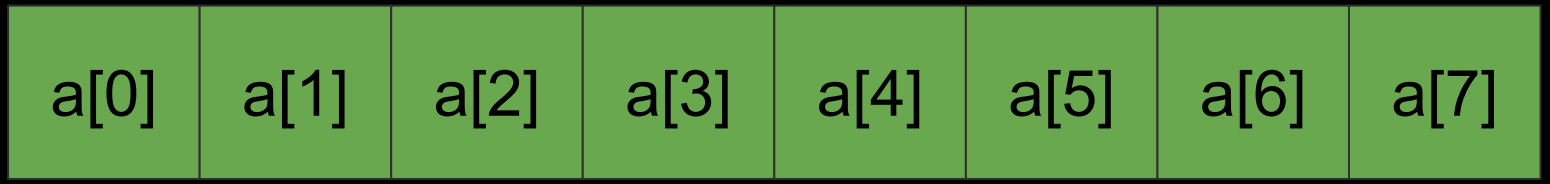
Array size in bytes
= 8 elements x 4 bytes per element = 32 bytes

Size of the array in words



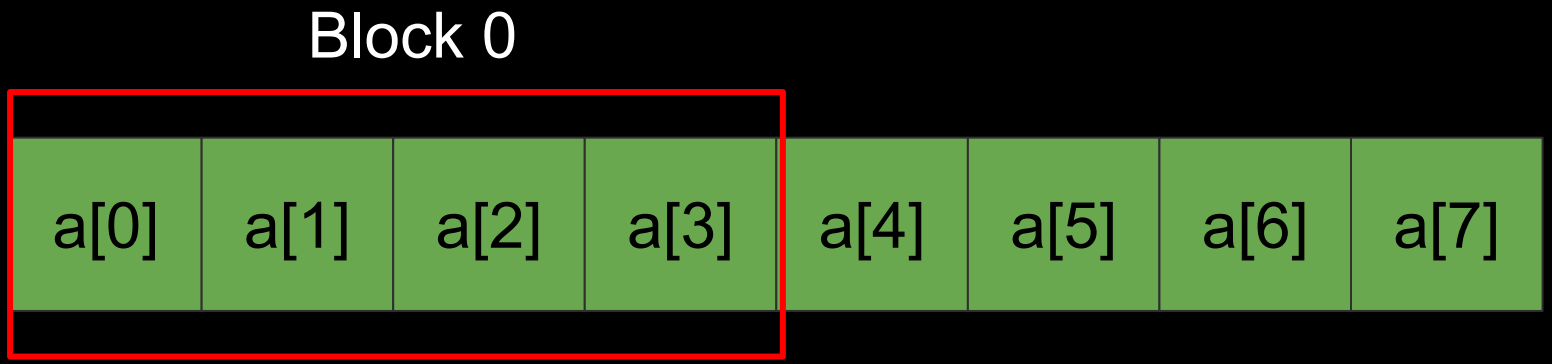
No. of words in the array
= 32 bytes / 4 bytes per word = 8 words

Size of the array in blocks



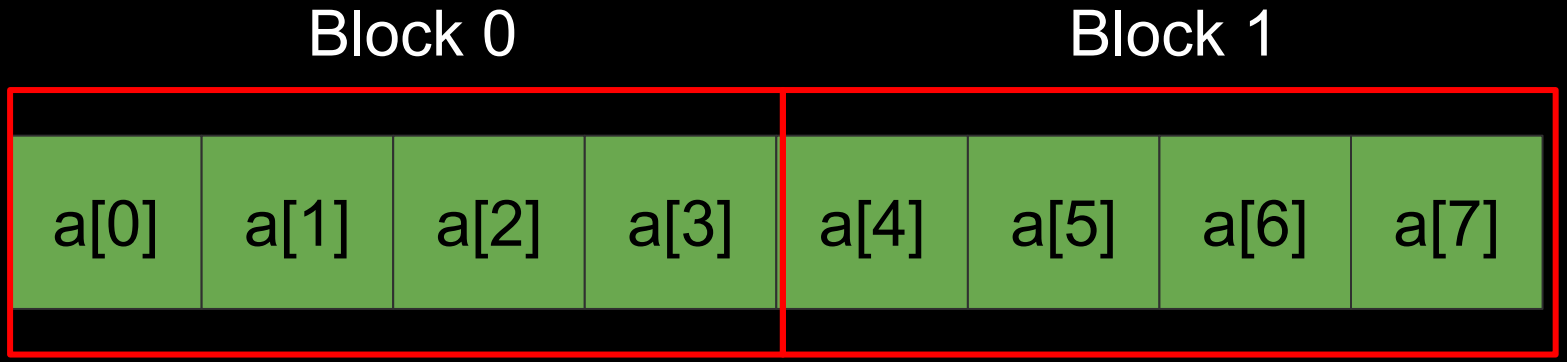
No. of blocks in the array
= 8 words / 4 words per block = 2 blocks

Size of the array in blocks



No. of blocks in the array
= 8 words / 4 words per block = 2 blocks

Size of the array in blocks



No. of blocks in the array
= 8 words / 4 words per block = 2 blocks

Sum the elements of an array

```
int sum_array(int a[], int n)
{
    int i, sum = 0;

    for(i = 0; i < n; i++) {
        sum += a[i];
    }

    return sum;
}
```

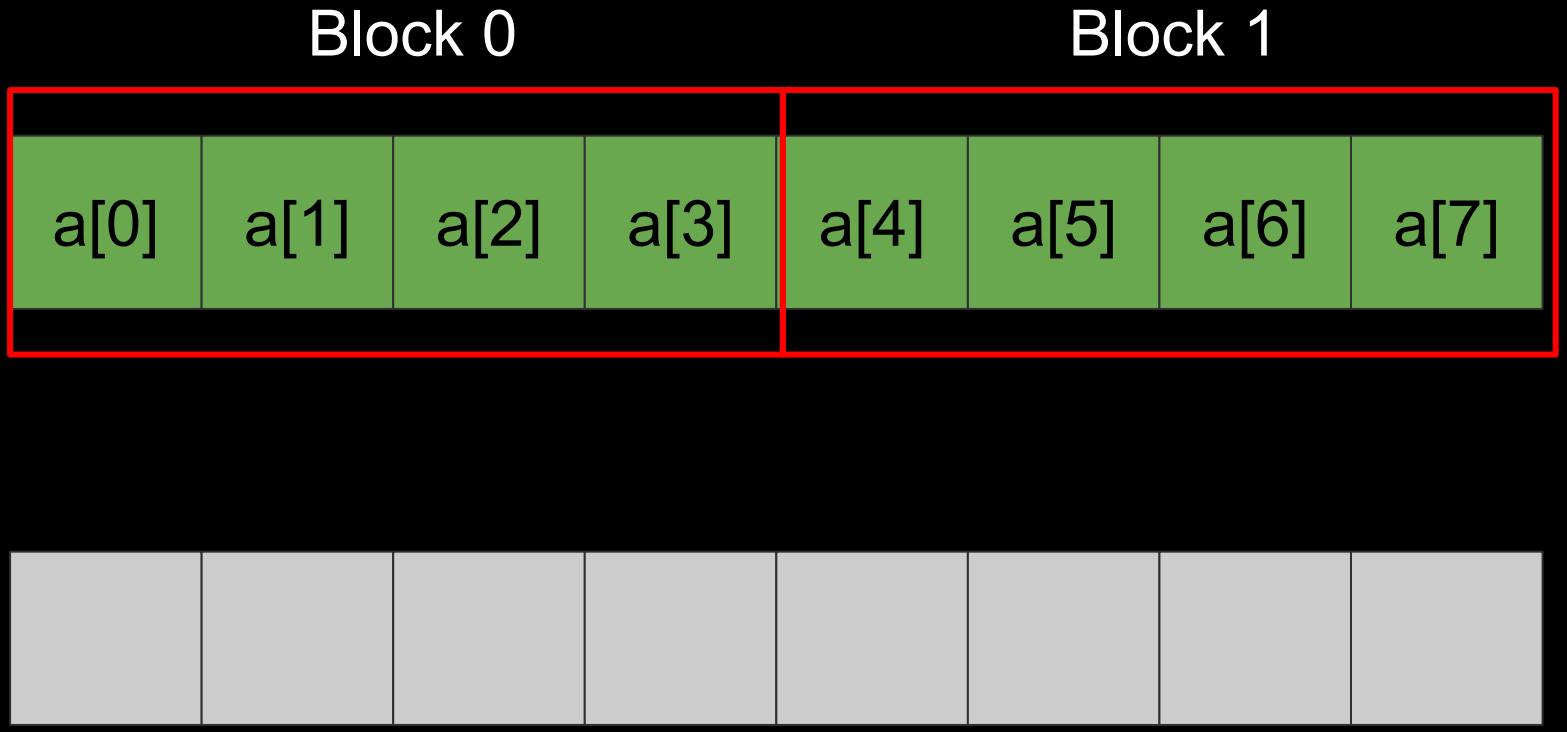
Sum the elements of an array

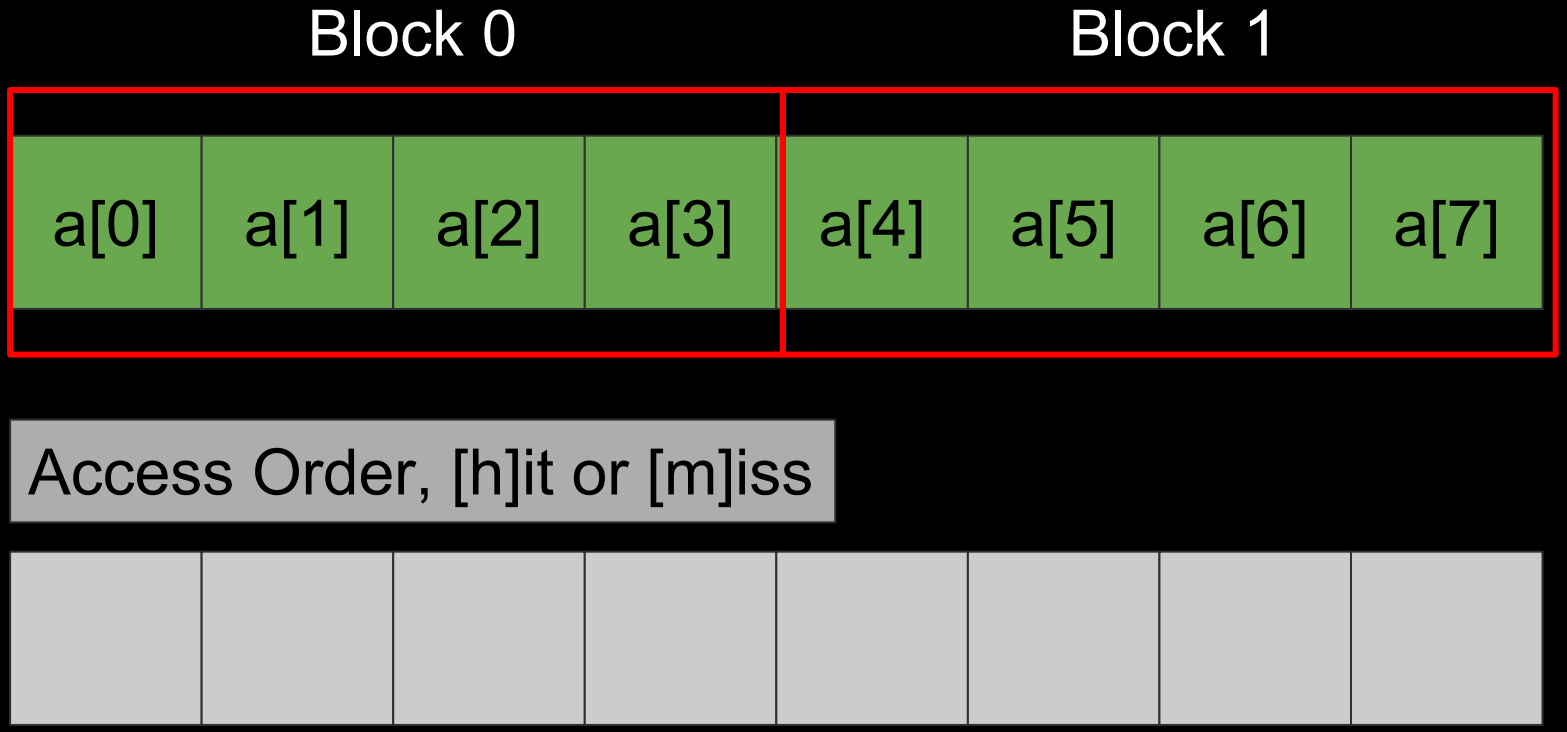
```
int sum_array(int a[], int n)
{
    int i, sum = 0;

    for(i = 0; i < n; i++) {
        sum += a[i];
    }

    return sum;
}
```

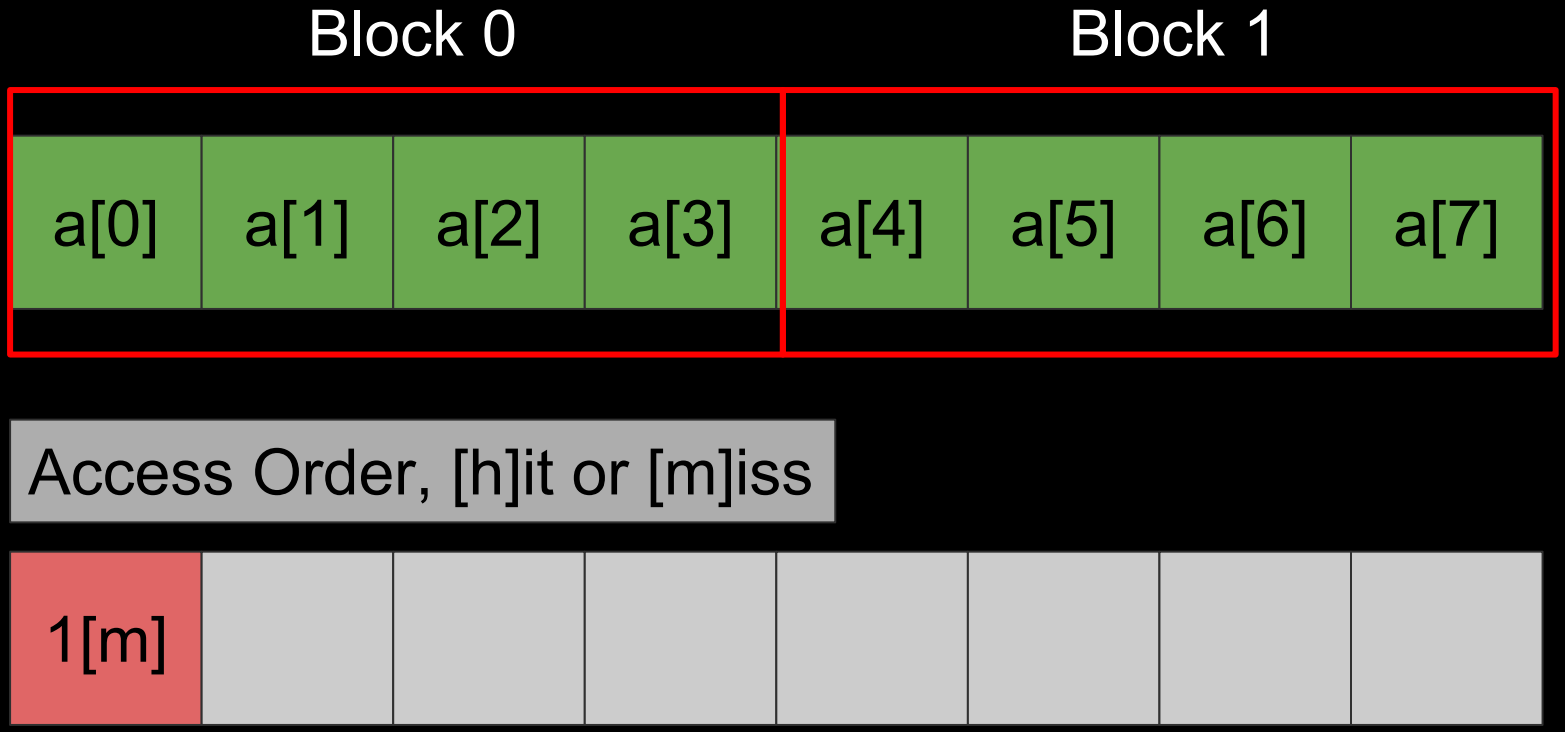
Sum the elements of an array

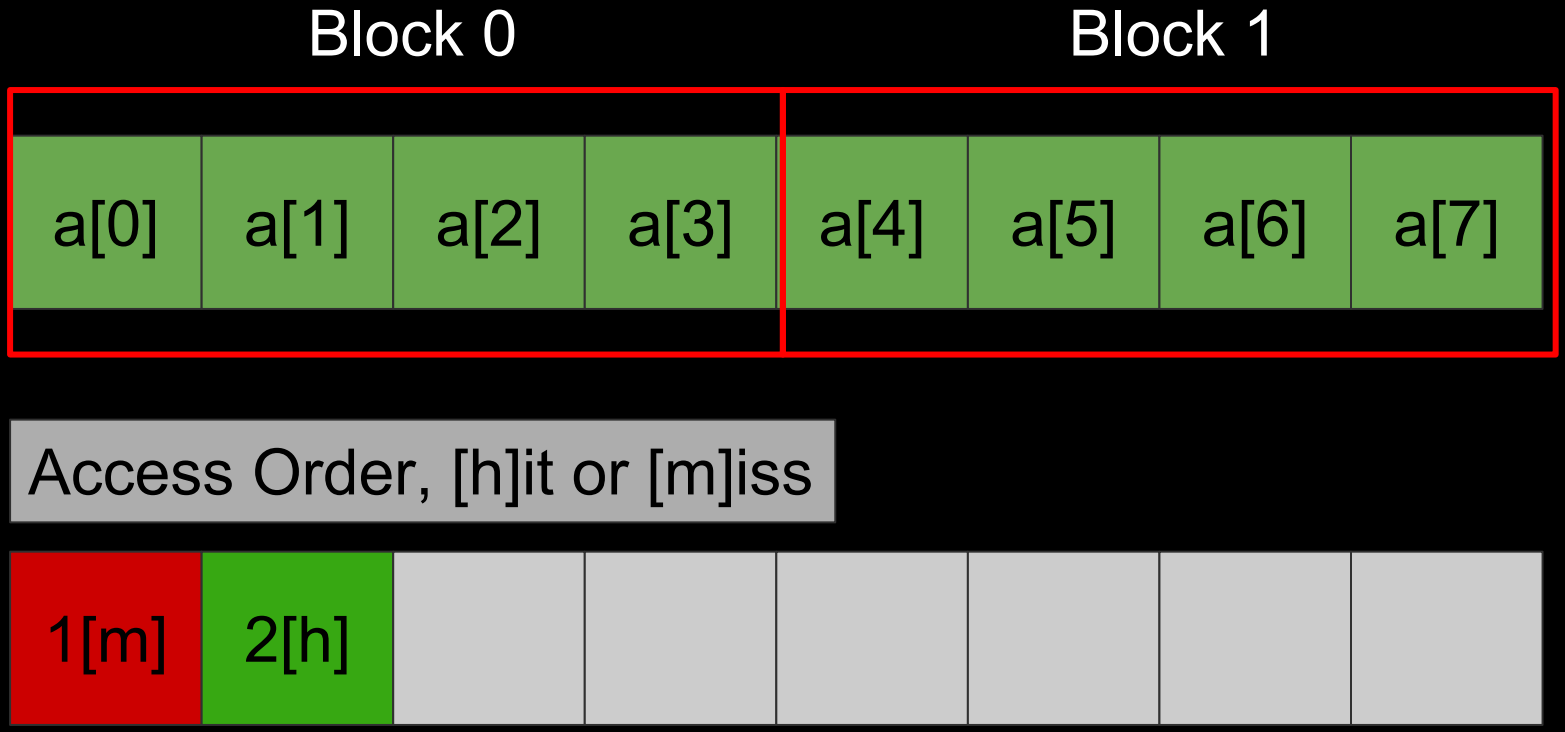
Stride-1 reference pattern

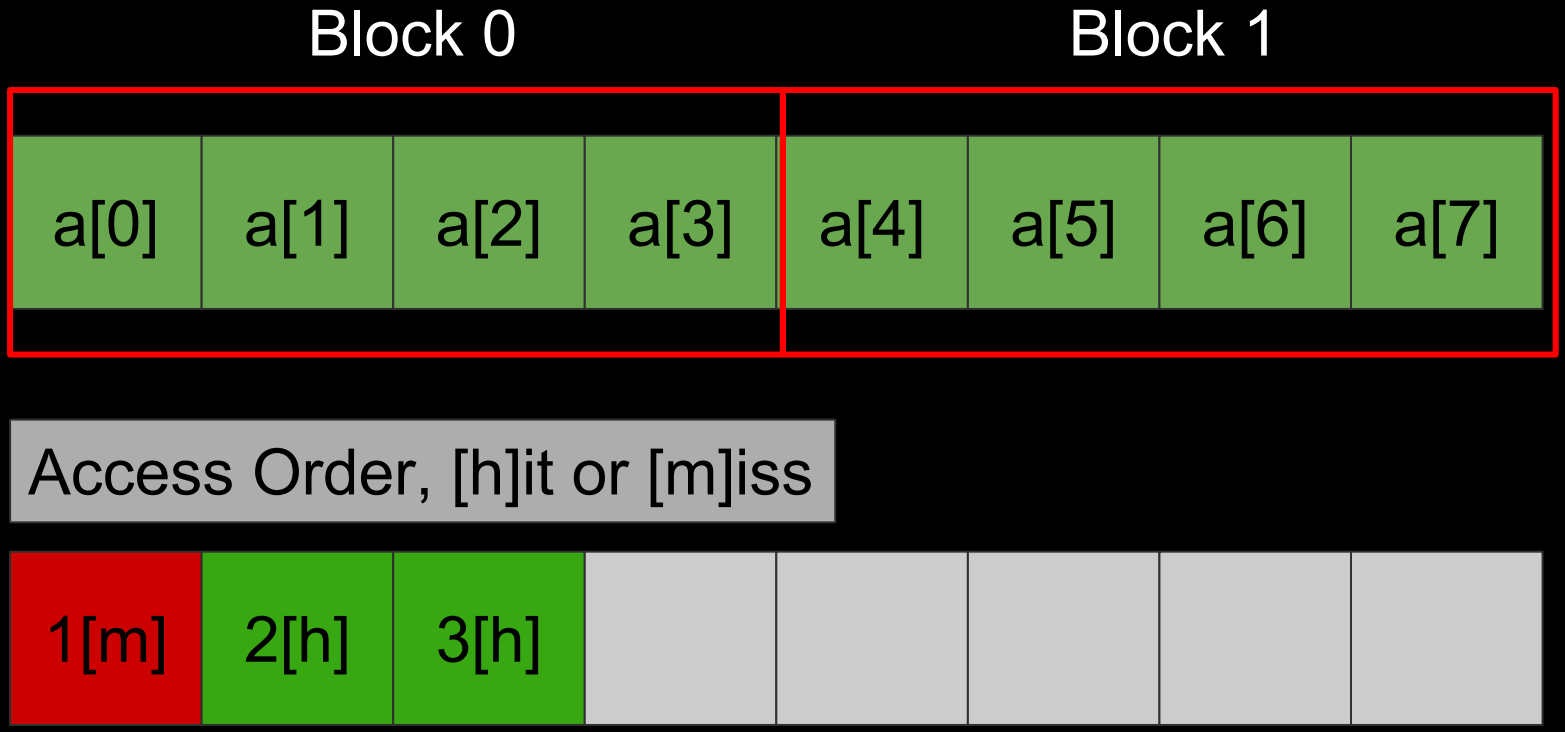
```
    for(i = 0; i < n; i++) {
        sum += a[i];
    }

    return sum;
}
```

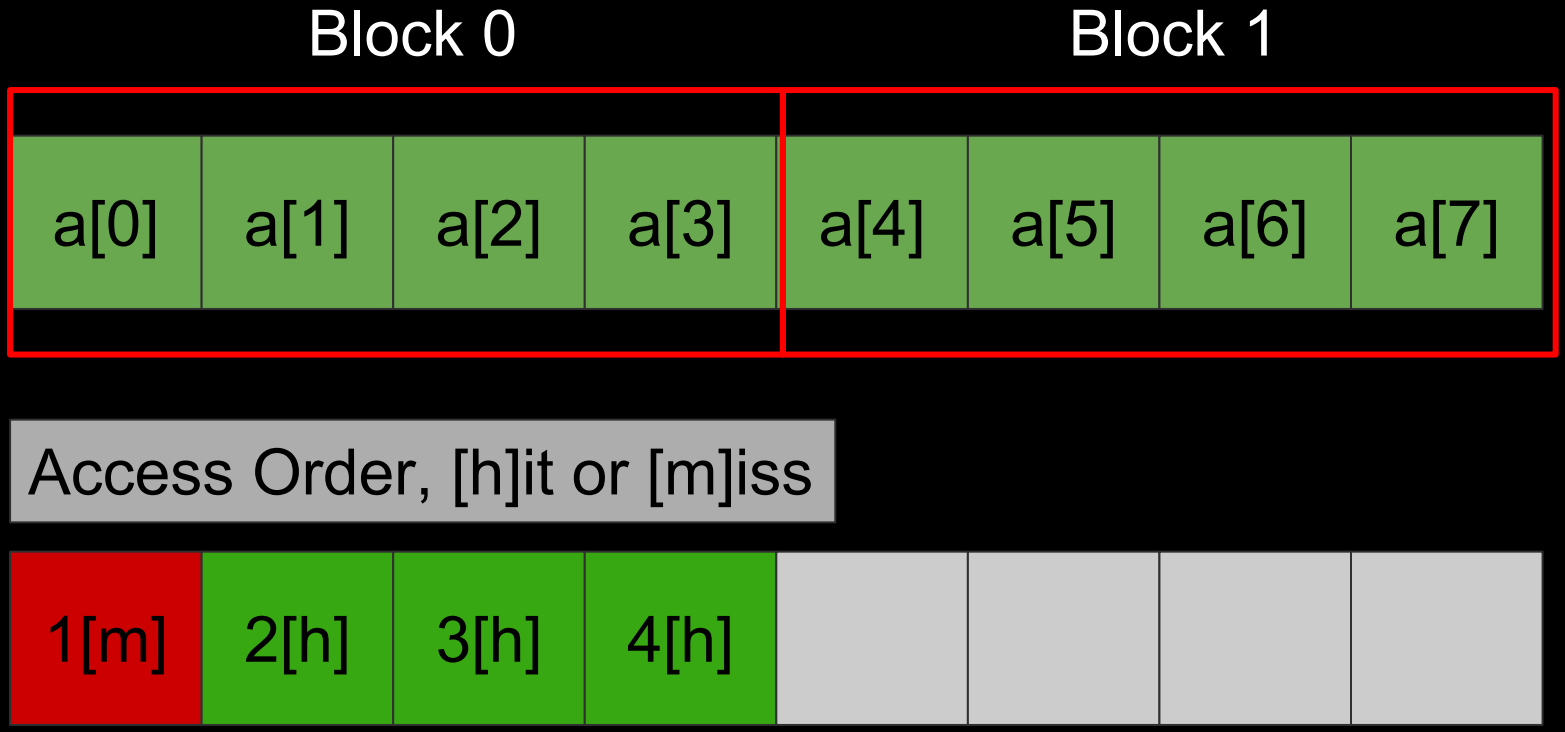

Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]

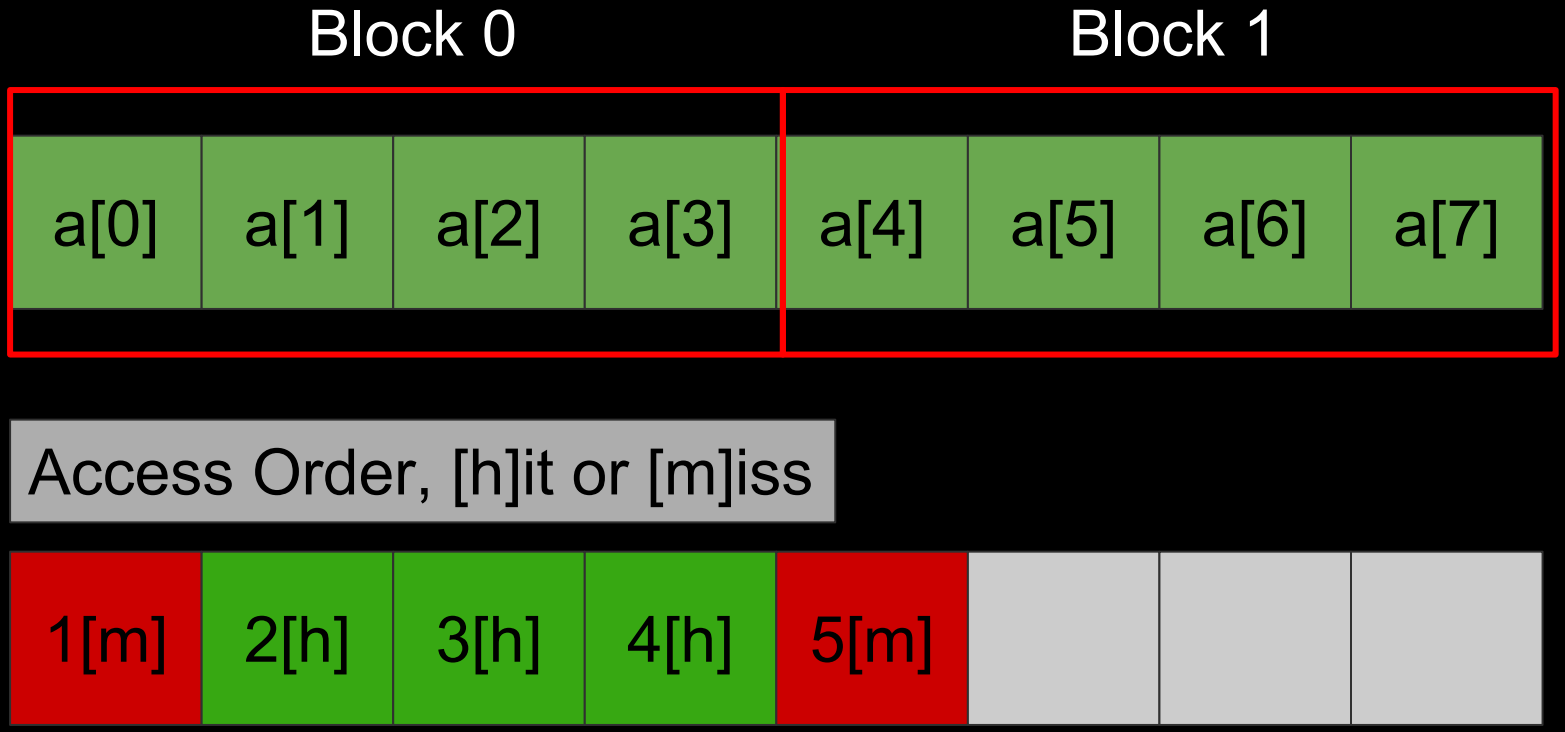

Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss

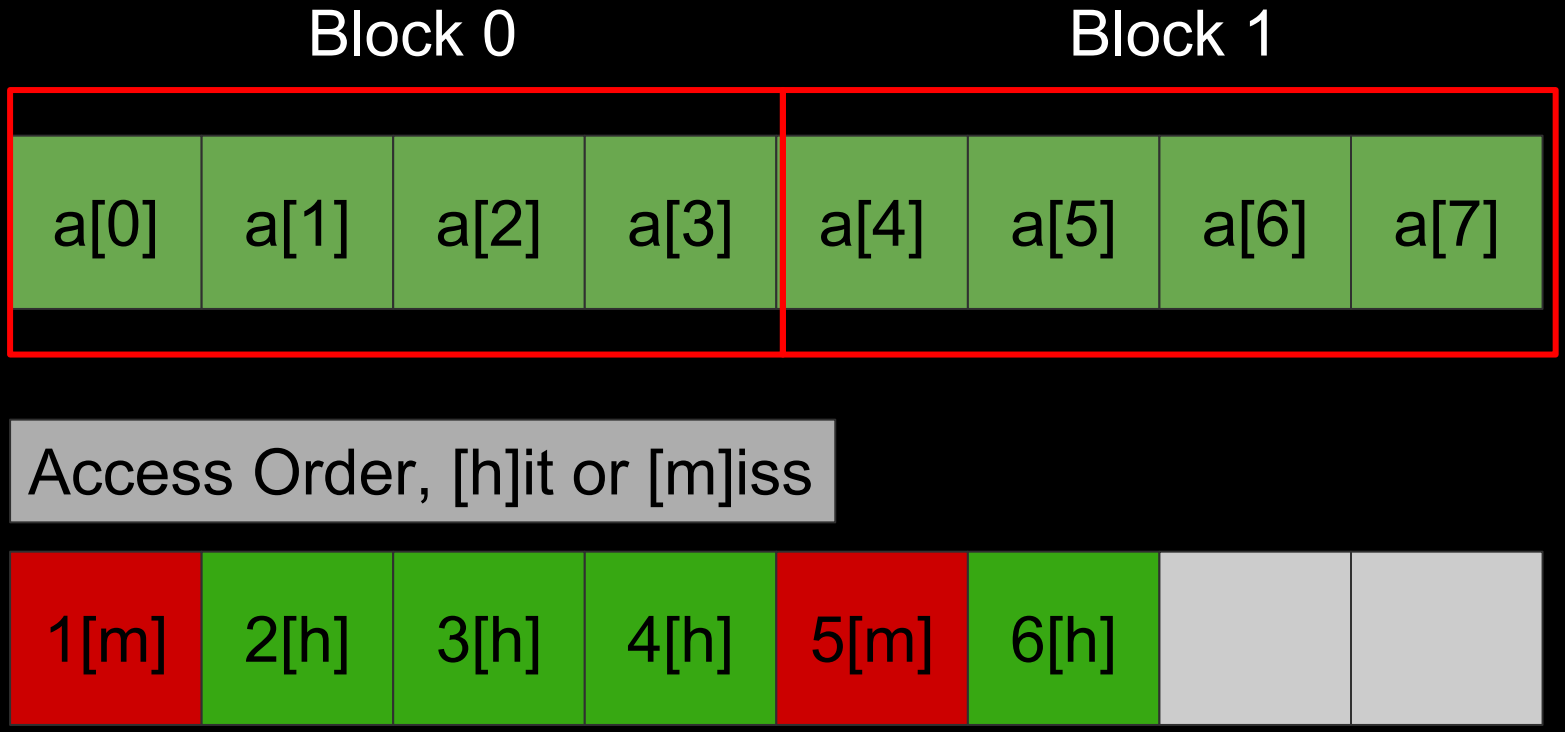

Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss
1[m]

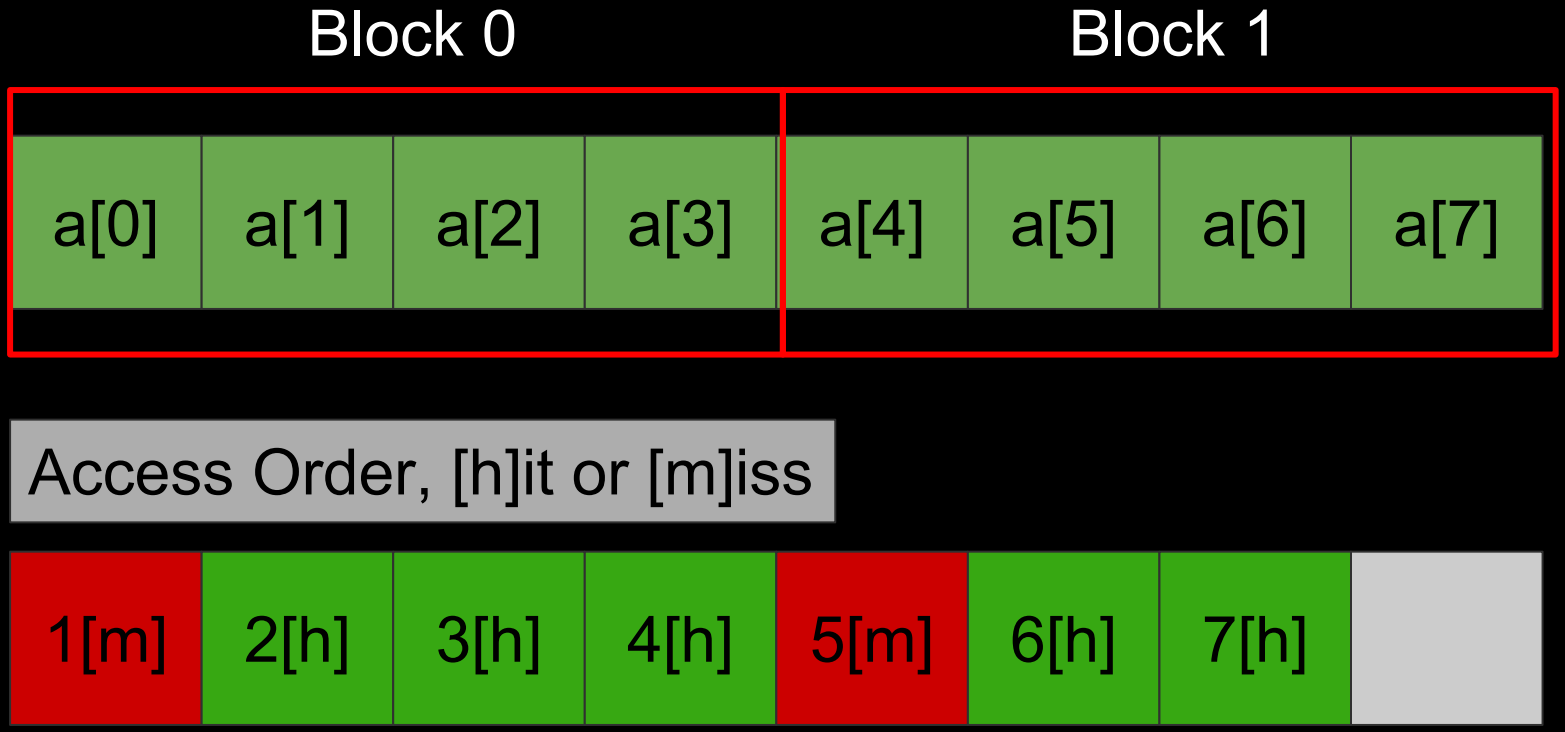

Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss
1[m]
2[h]

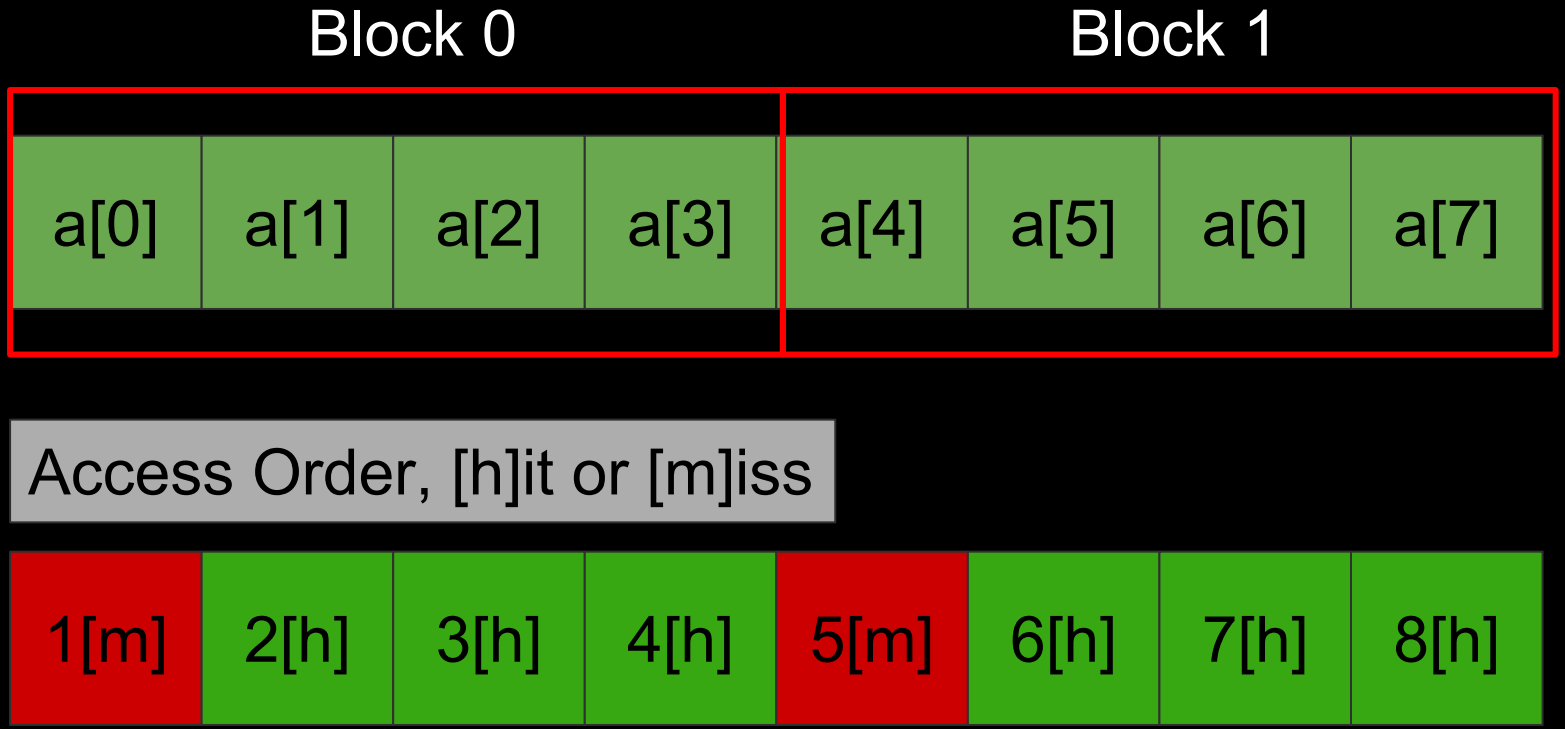

Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss
1[m]
2[h]
3[h]


Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss
1[m]
2[h]
3[h]
4[h]

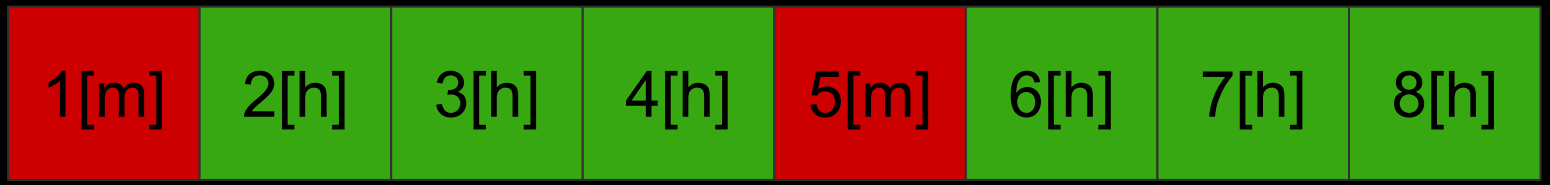

Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss
1[m]
2[h]
3[h]
4[h]
5[m]


Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss
1[m]
2[h]
3[h]
4[h]
5[m]
6[h]

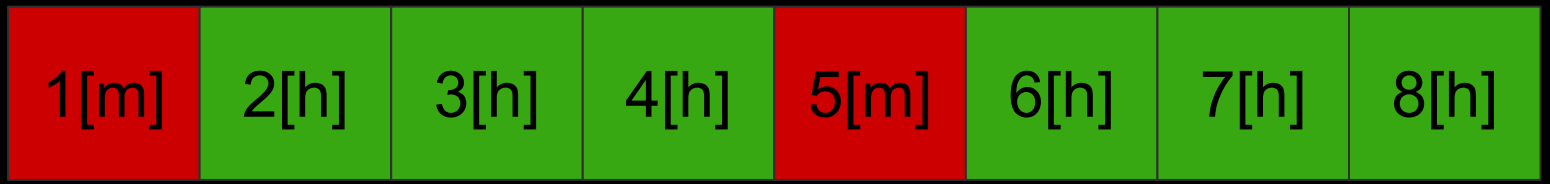

Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss
1[m]
2[h]
3[h]
4[h]
5[m]
6[h]
7[h]


Block 0
Block 1
a[0]
a[1]
a[2]
a[3]
a[4]
a[5]
a[6]
a[7]
Access Order, [h]it or [m]iss
1[m]
2[h]
3[h]
4[h]
5[m]
6[h]
7[h]
8[h]

Hit ratio = 6/8 = ¾ = 75 %

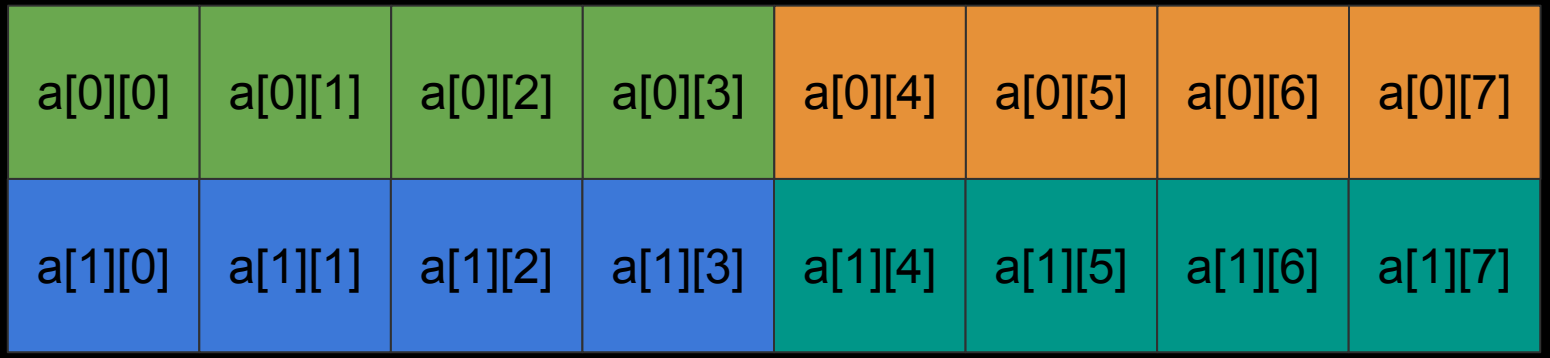


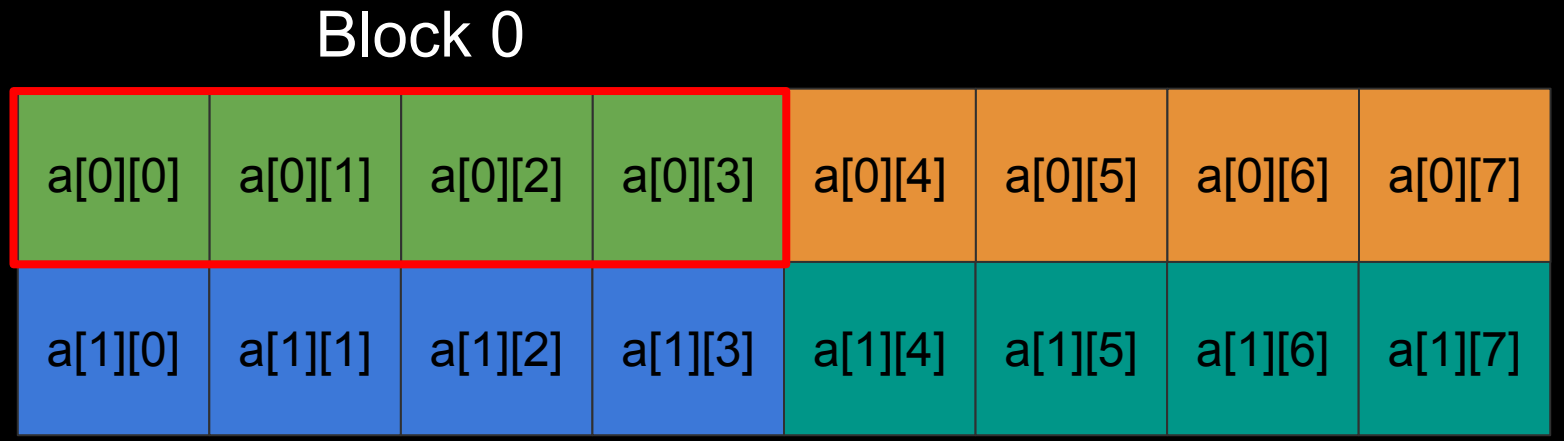
Miss ratio = 2/8 = ¼ = 25 %

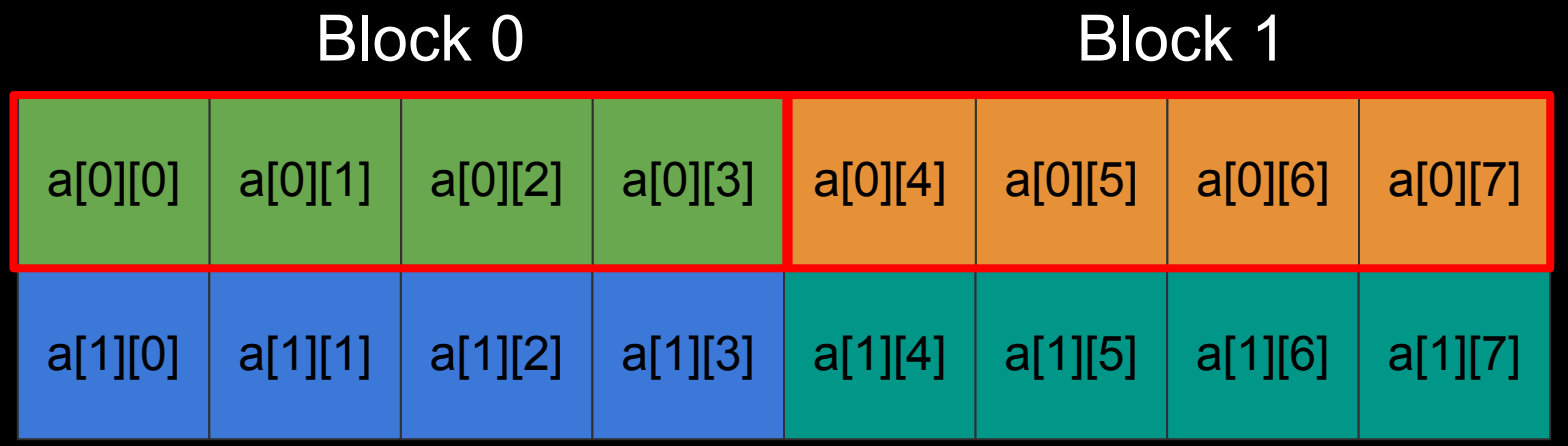


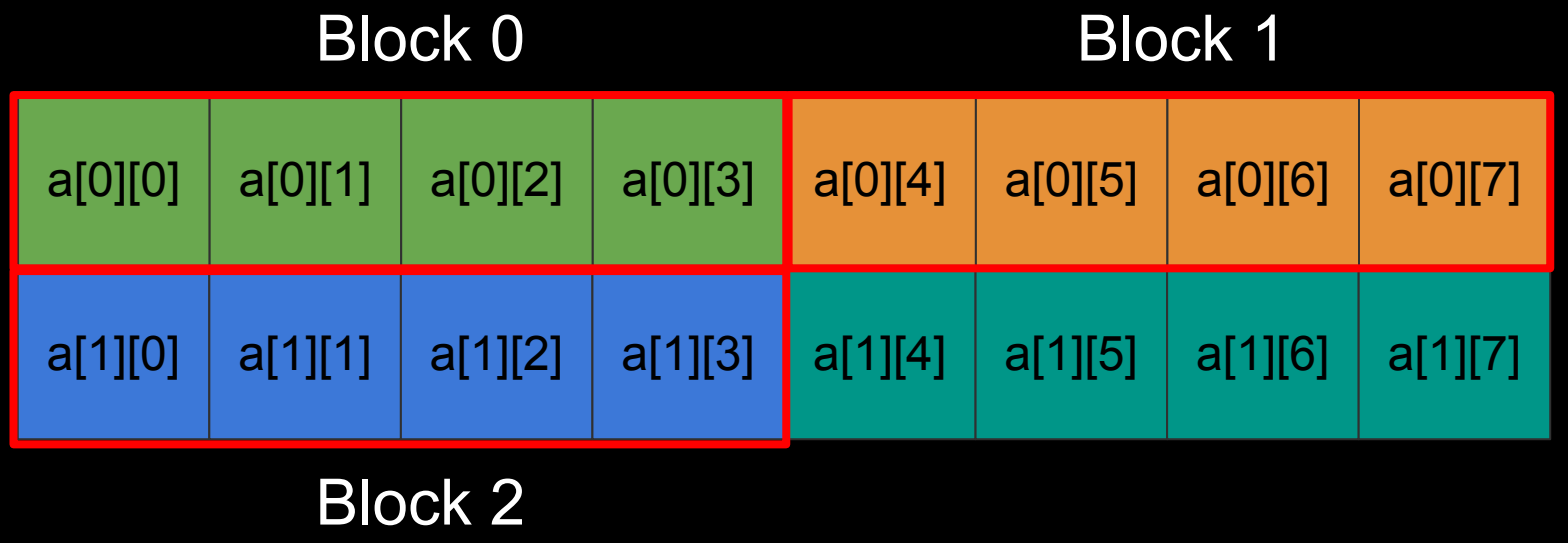
Sum the elements of a 2-d array row-wise

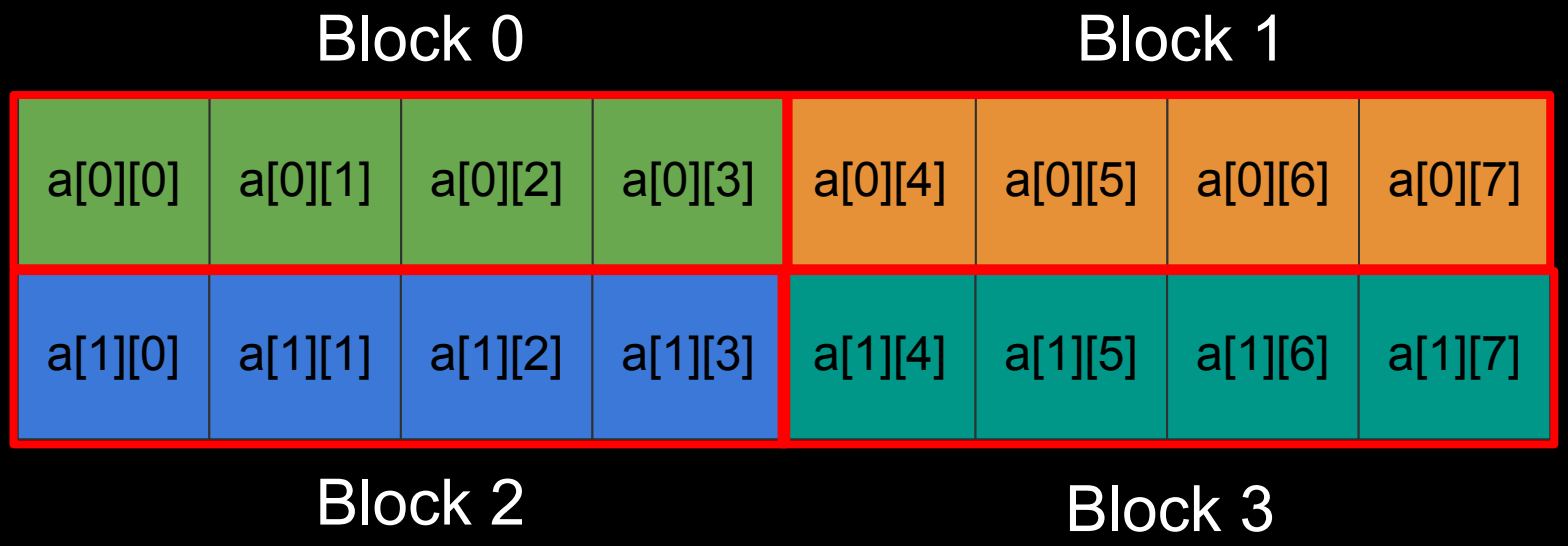
```
int sum_array_rows(int a[][], int m, int n)
{
    int i, j, sum = 0;

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


a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]
a[0][5]
a[0][6]
a[0][7]
a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]
a[1][5]
a[1][6]
a[1][7]


Block 0
a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]
a[0][5]
a[0][6]
a[0][7]
a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]
a[1][5]
a[1][6]
a[1][7]


Block 0
Block 1
a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]
a[0][5]
a[0][6]
a[0][7]
a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]
a[1][5]
a[1][6]
a[1][7]


Block 0
Block 1
a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]
a[0][5]
a[0][6]
a[0][7]
a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]
a[1][5]
a[1][6]
a[1][7]
Block 2


Block 0
Block 1
a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]
a[0][5]
a[0][6]
a[0][7]
a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]
a[1][5]
a[1][6]
a[1][7]
Block 2
Block 3

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]							

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]						

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]					

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]				

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]			

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]		

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]							

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]	10[h]						

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]	10[h]	11[h]					

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]	10[h]	11[h]	12[h]				

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]	10[h]	11[h]	12[h]	13[m]			

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]	10[h]	11[h]	12[h]	13[m]	14[h]		

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]	10[h]	11[h]	12[h]	13[m]	14[h]	15[h]	

Hit ratio = 12/16 = ¾ = 75 %

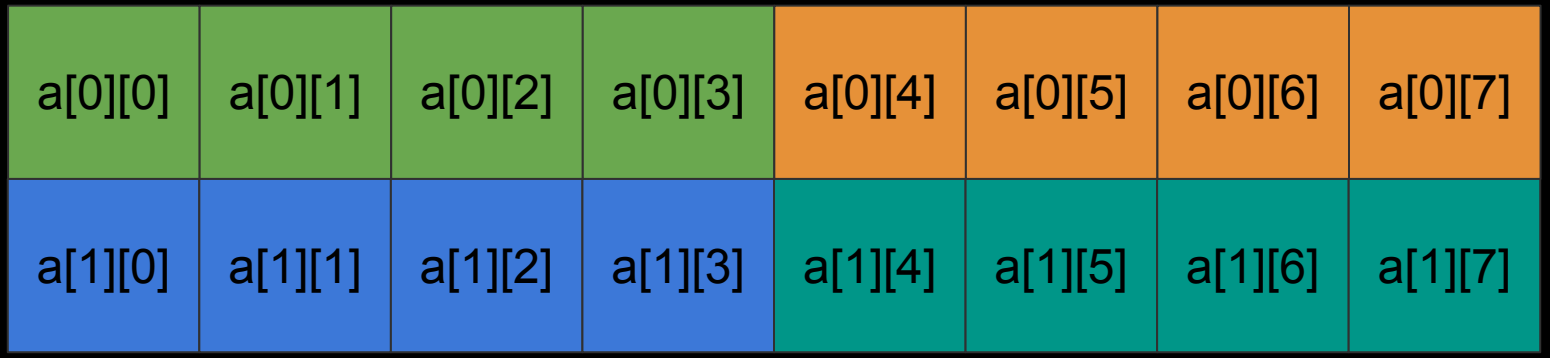
1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]	10[h]	11[h]	12[h]	13[m]	14[h]	15[h]	16[h]

Miss ratio = 4/16 = ¼ = 25 %

1[m]	2[h]	3[h]	4[h]	5[m]	6[h]	7[h]	8[h]
9[m]	10[h]	11[h]	12[h]	13[m]	14[h]	15[h]	16[h]

Sum the elements of a 2-d array column-wise

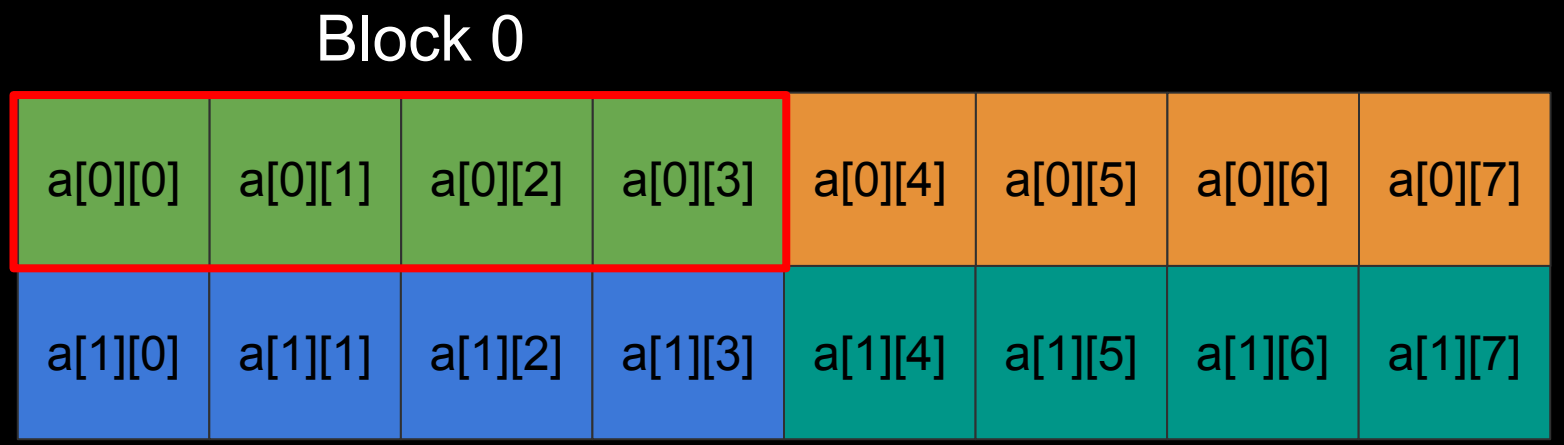
```
int sum_array_cols(int a[][], int m, int n)
{
    int i, j, sum = 0;

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


a[0][0]
a[0][1]
a[0][2]
a[0][3]
a[0][4]
a[0][5]
a[0][6]
a[0][7]
a[1][0]
a[1][1]
a[1][2]
a[1][3]
a[1][4]
a[1][5]
a[1][6]
a[1][7]

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

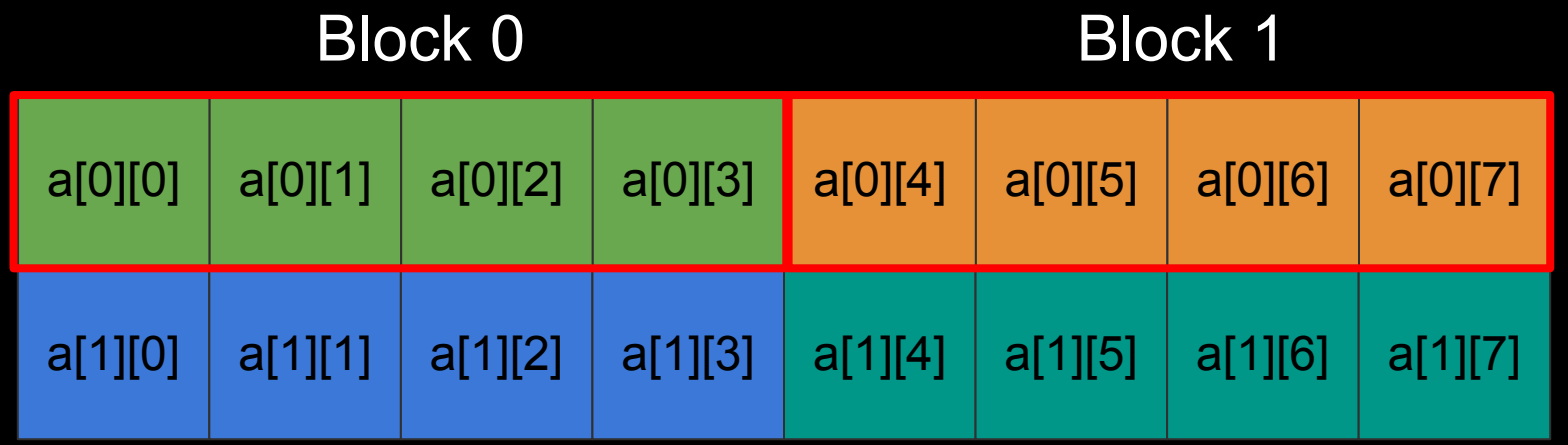
Assumption: Cache size is less that the array size.

For example: (S, E, B, m) = (2, 1, 16, 32)



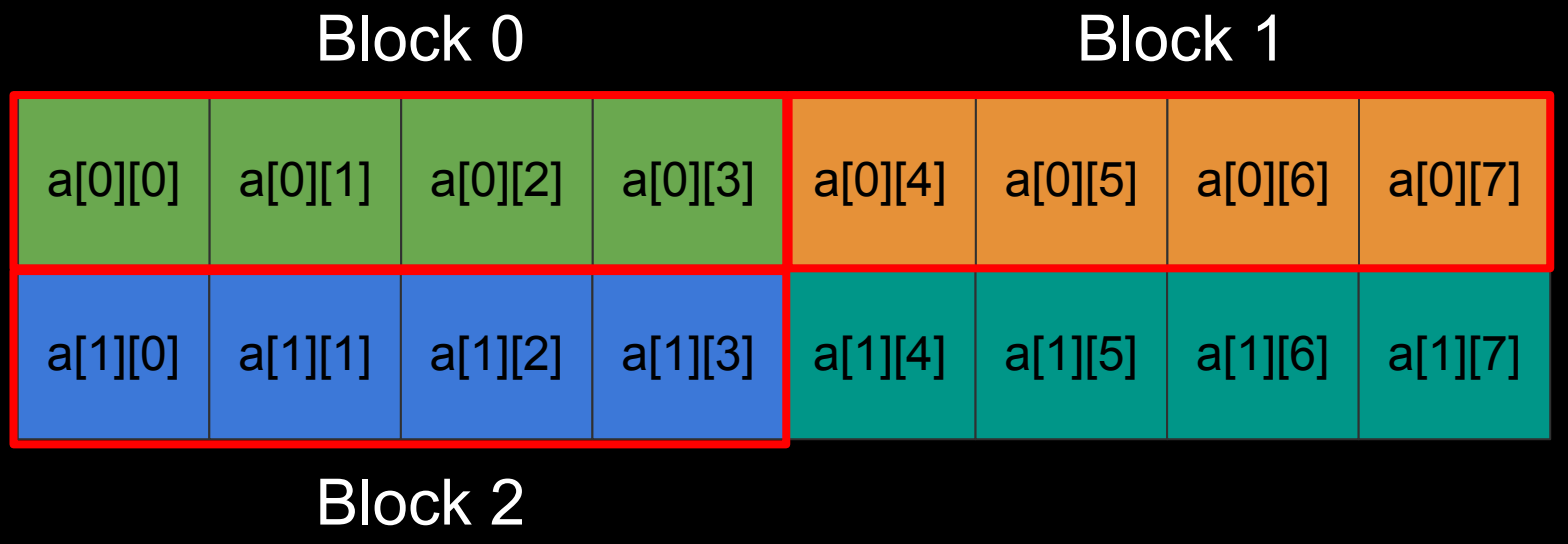
Assumption: Cache size is less that the array size.

For example: (S, E, B, m) = (2, 1, 16, 32)



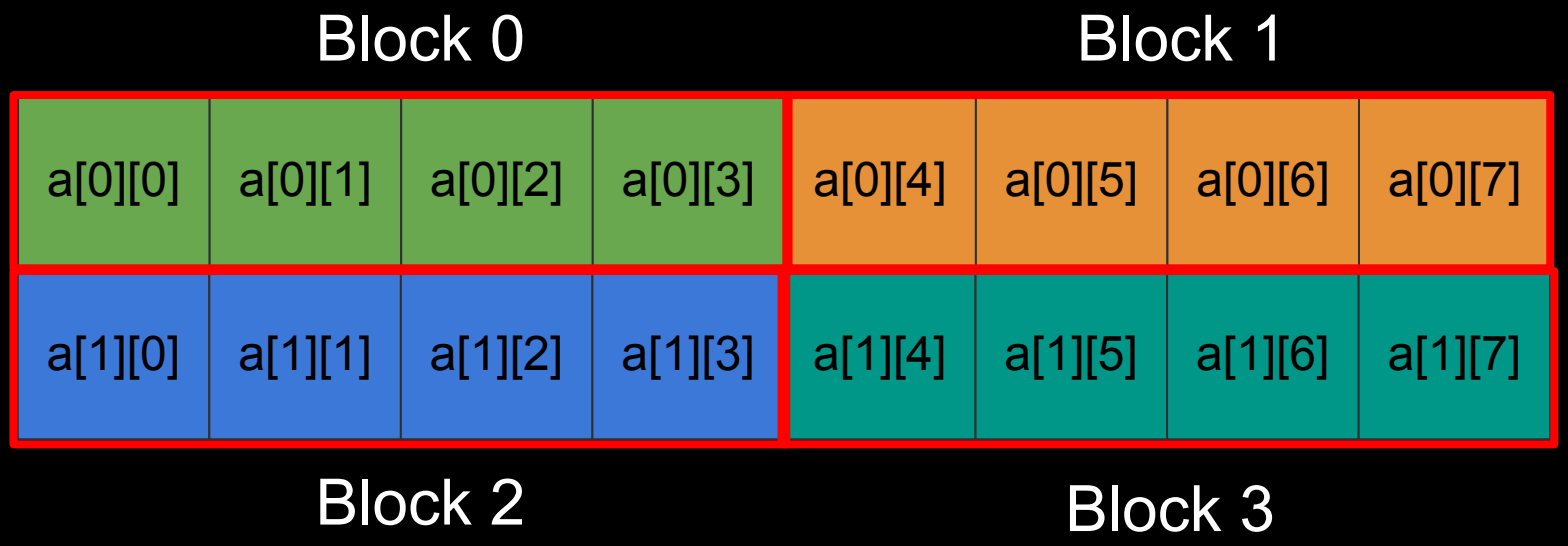
Assumption: Cache size is less that the array size.

For example: (S, E, B, m) = (2, 1, 16, 32)



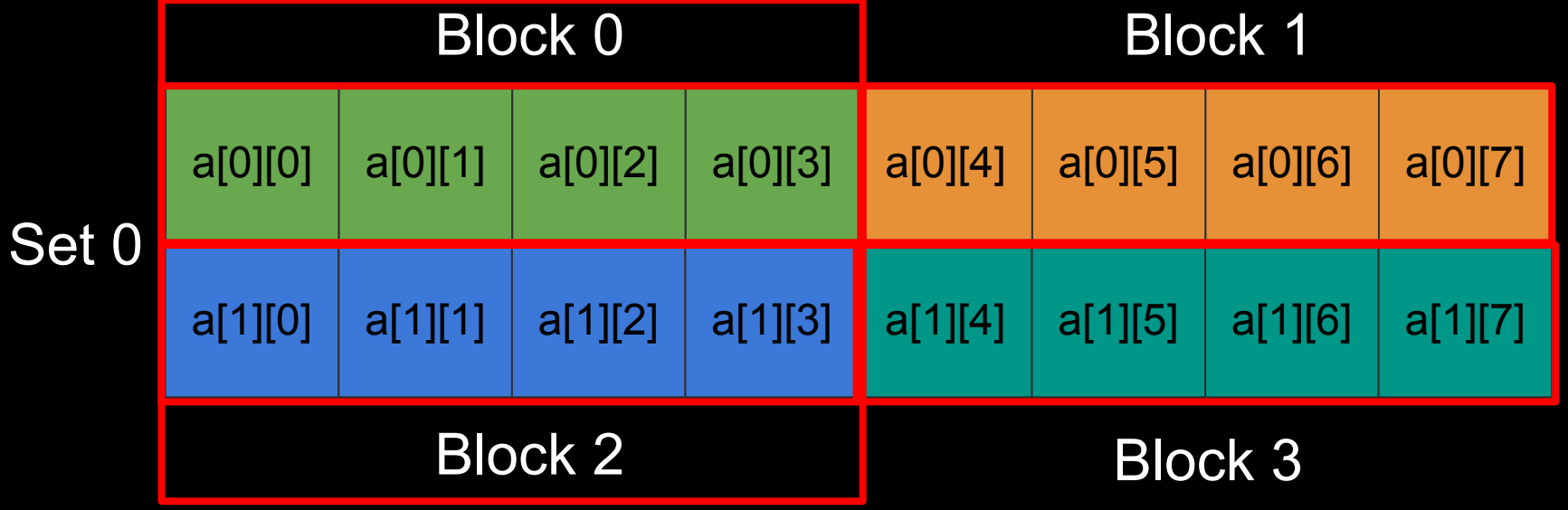
Assumption: Cache size is less that the array size.

For example: (S, E, B, m) = (2, 1, 16, 32)



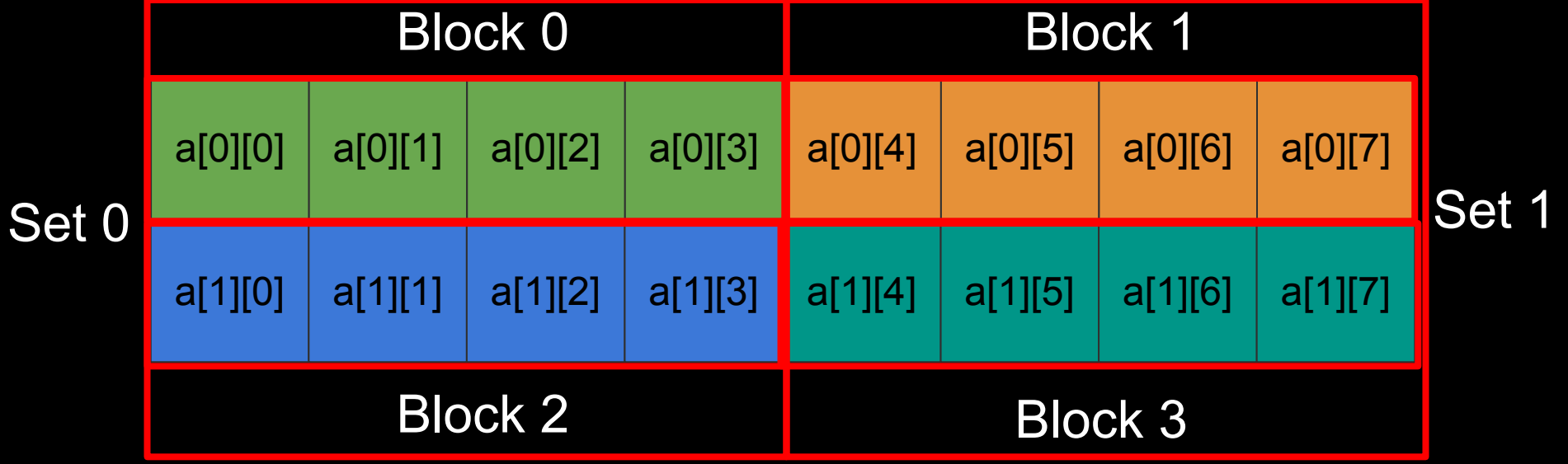
Assumption: Cache size is less that the array size.

For example: (S, E, B, m) = (2, 1, 16, 32)



Assumption: Cache size is less that the array size.

For example: (S, E, B, m) = (2, 1, 16, 32)



Assumption: Cache size is less that the array size.

For example: (S, E, B, m) = (2, 1, 16, 32)

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]							

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]							
2[m]							

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]						
2[m]							

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]						
2[m]	4[m]						

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]					
2[m]	4[m]						

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]					
2[m]	4[m]	6[m]					

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]				
2[m]	4[m]	6[m]					

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]				
2[m]	4[m]	6[m]	8[m]				

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]	9[m]			
2[m]	4[m]	6[m]	8[m]				

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]	9[m]			
2[m]	4[m]	6[m]	8[m]	10[m]			

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]	9[m]	11[m]		
2[m]	4[m]	6[m]	8[m]	10[m]			

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]	9[m]	11[m]		
2[m]	4[m]	6[m]	8[m]	10[m]	12[m]		

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]	9[m]	11[m]	13[m]	
2[m]	4[m]	6[m]	8[m]	10[m]	12[m]		

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]	9[m]	11[m]	13[m]	
2[m]	4[m]	6[m]	8[m]	10[m]	12[m]	14[m]	

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]	9[m]	11[m]	13[m]	15[m]
2[m]	4[m]	6[m]	8[m]	10[m]	12[m]	14[m]	

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]	a[0][7]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]	a[1][6]	a[1][7]

Access Order, [h]it or [m]iss

1[m]	3[m]	5[m]	7[m]	9[m]	11[m]	13[m]	15[m]
2[m]	4[m]	6[m]	8[m]	10[m]	12[m]	14[m]	16[m]

Hit ratio = 0/16 = 0 %

1[m]	3[m]	5[m]	7[m]	9[m]	11[m]	13[m]	15[m]
2[m]	4[m]	6[m]	8[m]	10[m]	12[m]	14[m]	16[m]

Miss ratio = 16/16 = 100 %

1[m]	3[m]	5[m]	7[m]	9[m]	11[m]	13[m]	15[m]
2[m]	4[m]	6[m]	8[m]	10[m]	12[m]	14[m]	16[m]