

Introduction to 8086 Assembly

Lecture 17

2D and N-D Arrays

2D Arrays



K. N. Toosi
University of Technology

0	0	0	0	0	0
0	1	2	3	4	5
0	2	4	6	8	10
0	3	6	9	12	15
0	4	8	12	16	20

- tabular data
- rows and columns

2D Arrays



K. N. Toosi
University of Technology

0	0	0	0	0	0
0	1	2	3	4	5
0	2	4	6	8	10
0	3	6	9	12	15
0	4	8	12	16	20
0	5	10	15	20	25

$$A = \begin{pmatrix} 3 & -5 & 4 \\ 9 & 8 & -7 \\ -6 & 4 & 2 \end{pmatrix}, B = \begin{pmatrix} -2 & -1 & 1 \\ 5 & -7 & 6 \\ 9 & 3 & 2 \end{pmatrix}$$

<https://advancedmathclubsk.weebly.com/matrices.html>

2D Arrays



K. N. Toosi
University of Technology

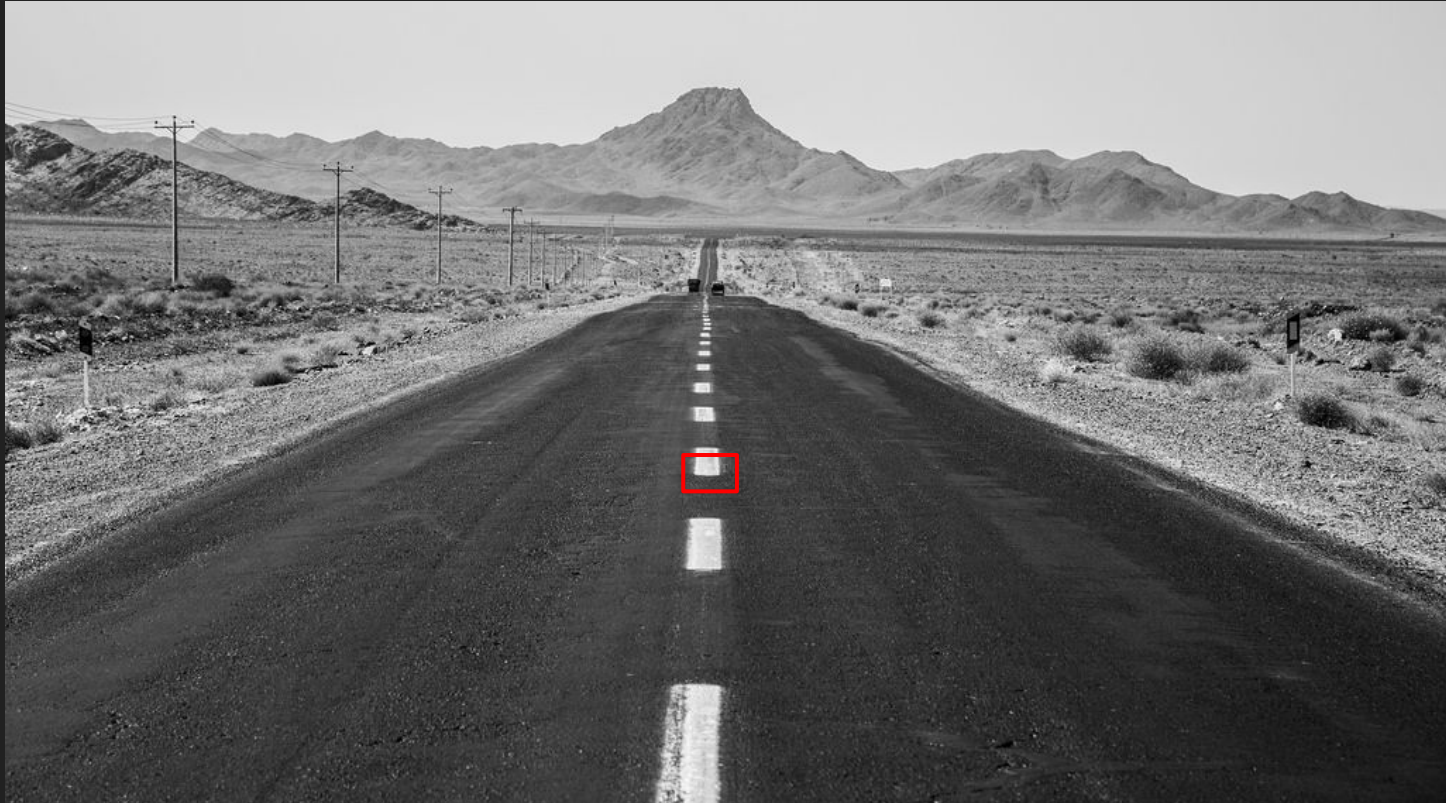


<https://hiveminer.com/Tags/desert.isfahan/Recent>

2D Arrays



K. N. Toosi
University of Technology

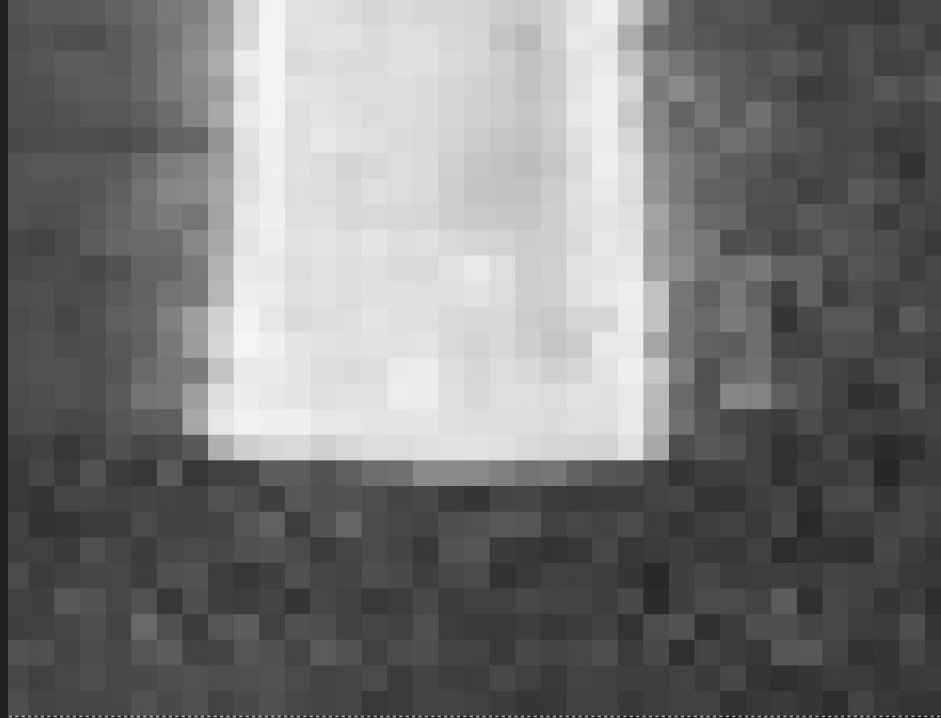


<https://hiveminer.com/Tags/desert.isfahan/Recent>

2D Arrays



K. N. Toosi
University of Technology



2D Arrays



K. N. Toosi
University of Technology

```
int a[3][4] = {{1, 3, 5, 7},  
              {2, 4, 6, 8},  
              {4, 11, -1, 7}};
```

```
printf("%d\n", a[1][2]);
```

	0	1	2	3
0	1	3	5	7
1	2	4	6	8
2	4	11	-1	7

2D Arrays



K. N. Toosi
University of Technology

```
int a[3][4] = {{1, 3, 5, 7},  
              {2, 4, 6, 8},  
              {4, 11, -1, 7}};
```

```
printf("%d\n", a[i][j]);
```

	0	1	2	3
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

How to implement 2D arrays?

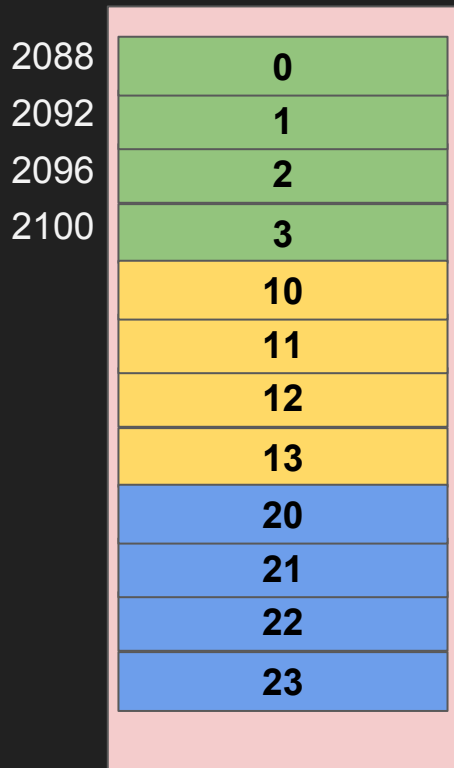


K. N. Toosi
University of Technology

	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23



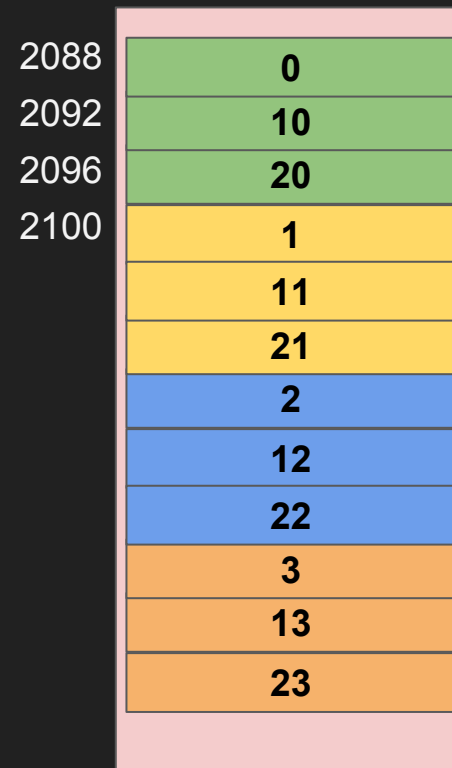
Memory



row by row

	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

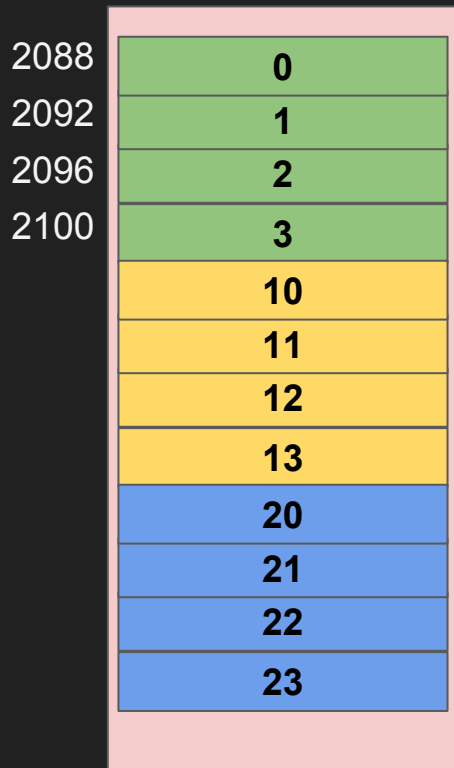
Memory



column by column



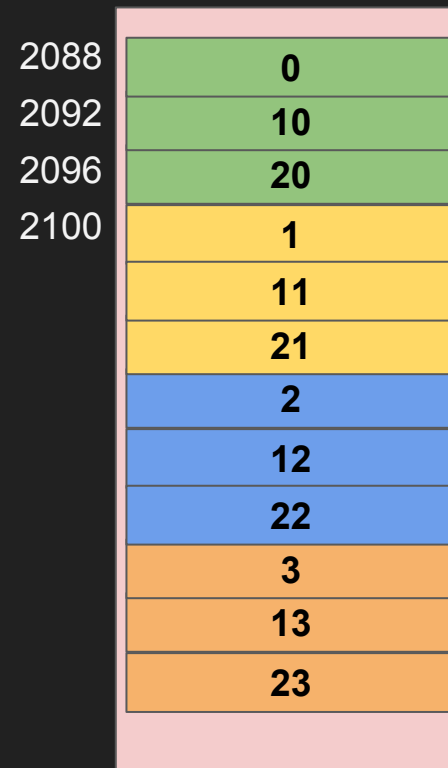
Memory



Row Major

	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

Memory

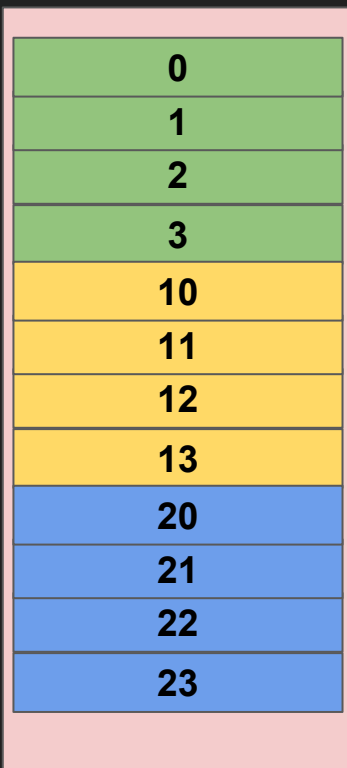


Column Major



Memory

2088
2092
2096
2100



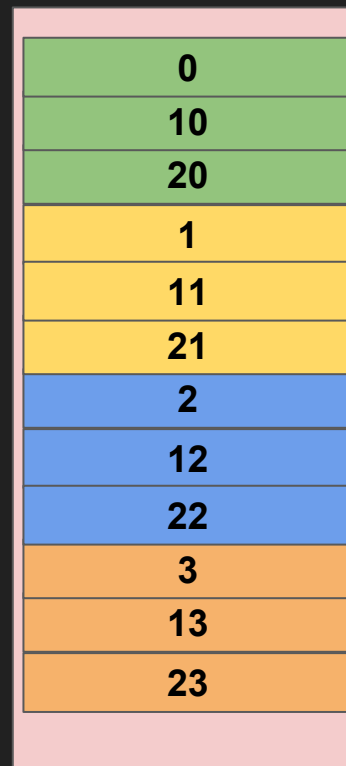
Row Major

(C,C++,Pascal, Python-numpy)

	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

Memory

2088
2092
2096
2100



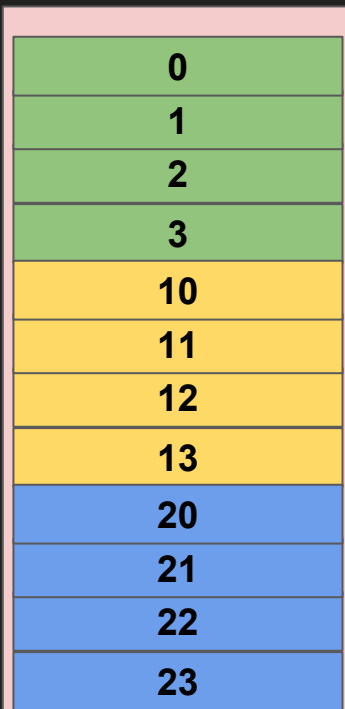
Column Major

(Fortran, Matlab, R, ...)



Memory

2088
2092
2096
2100

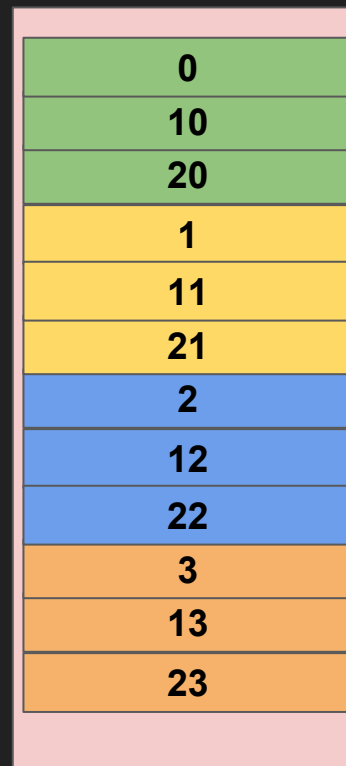


	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

Assembly?

Memory

2088
2092
2096
2100

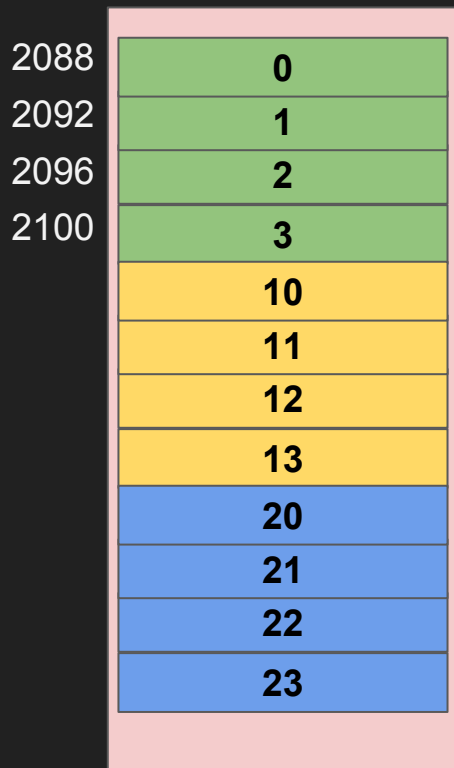


Row Major
(C,C++,Pascal, Python-numpy)

Column Major
(Fortran, Matlab, R, ...)



Memory

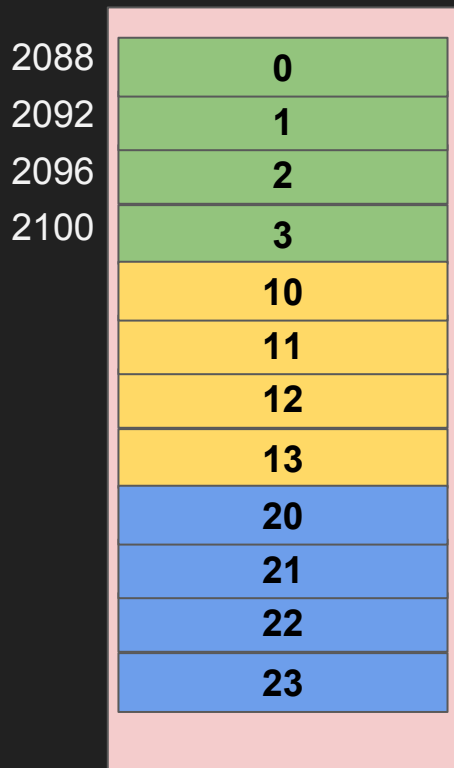


	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

Row Major
(C,C++,Pascal, numpy)



Memory



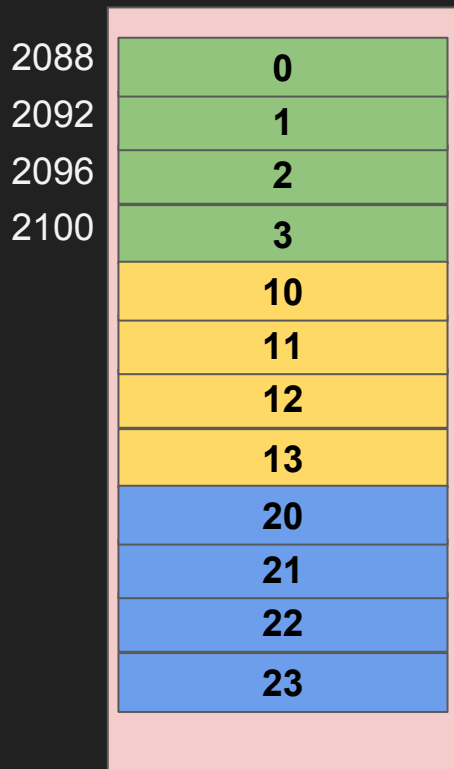
`a[i][0] : ?`

	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

Row Major
(C,C++,Pascal, numpy)



Memory



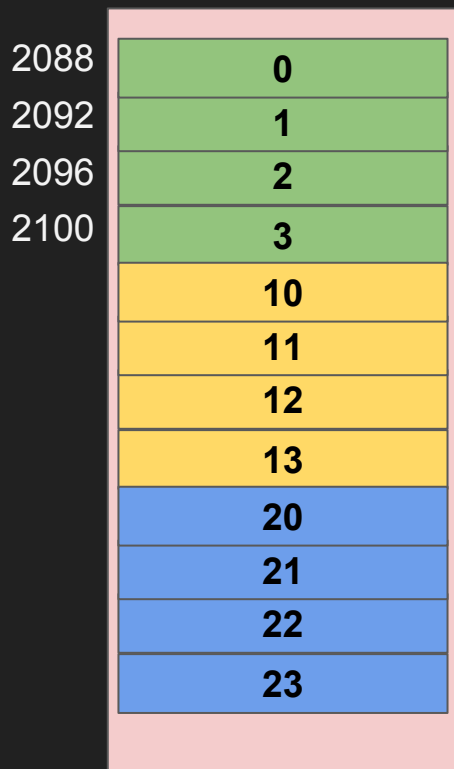
$$a[i][0] : 4*i$$

	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

Row Major
(C,C++,Pascal, numpy)



Memory



Row Major
(C,C++,Pascal, numpy)

	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

`a[i][0] : 4*i`

`a[i][1] : ?`



Memory

2088	0
2092	1
2096	2
2100	3
	10
	11
	12
	13
	20
	21
	22
	23

Row Major
(C,C++,Pascal, numpy)

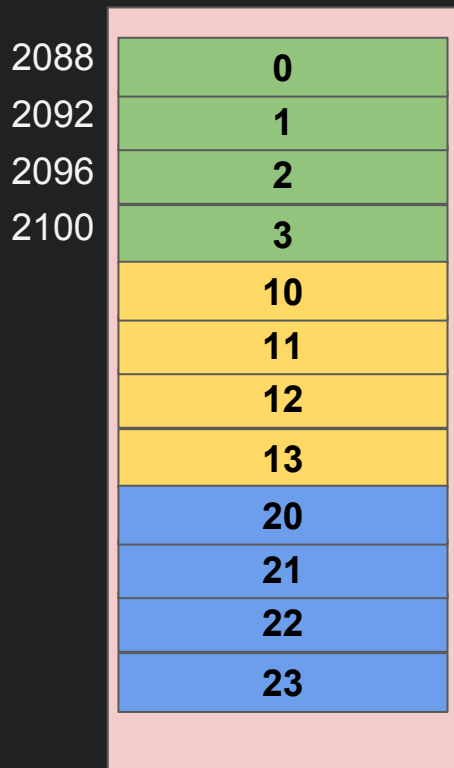
	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

$$a[i][0] : 4*i$$

$$a[i][1] : 4*i+1$$



Memory



Row Major
(C,C++,Pascal, numpy)

	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

$a[i][0] : 4*i$

$a[i][j] : ?$



Memory

2088	0
2092	1
2096	2
2100	3
	10
	11
	12
	13
	20
	21
	22
	23

Row Major
(C,C++,Pascal, numpy)

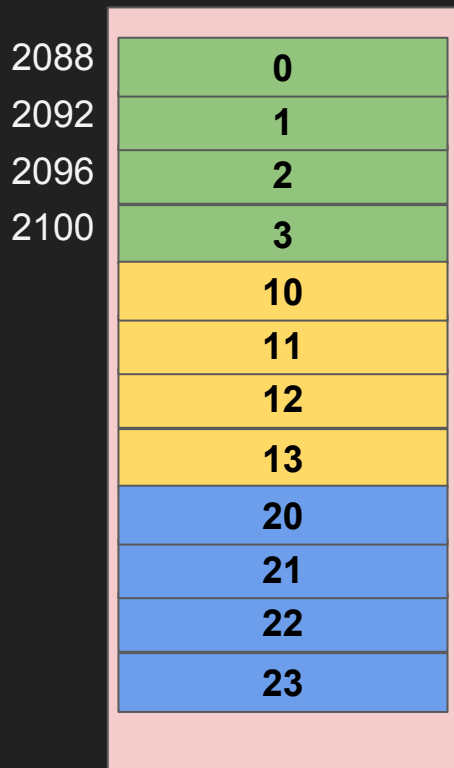
	0	1	2	3
0	0	1	2	3
1	10	11	12	13
2	20	21	22	23

$$a[i][0] : 4*i$$

$$a[i][j] : 4*i+j$$



Memory



Row Major
(C,C++,Pascal, numpy)

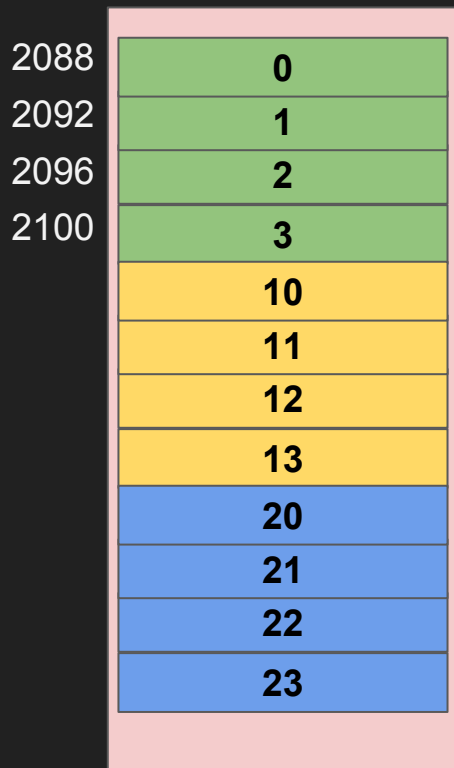
	0	1	...	n-1
0	0	1	...	9
1	10	11	...	19
	⋮	⋮		⋮
	⋮	⋮		⋮
m-1	80	81	...	83

```
int a[m][n];
```

```
a[i][j] : n*i + j
```



Memory



Row Major
(C,C++,Pascal, numpy)

	0	1	...	n-1
0	0	1	...	9
1	10	11	...	19
	⋮	⋮		⋮
	⋮	⋮		⋮
m-1	80	81	...	83

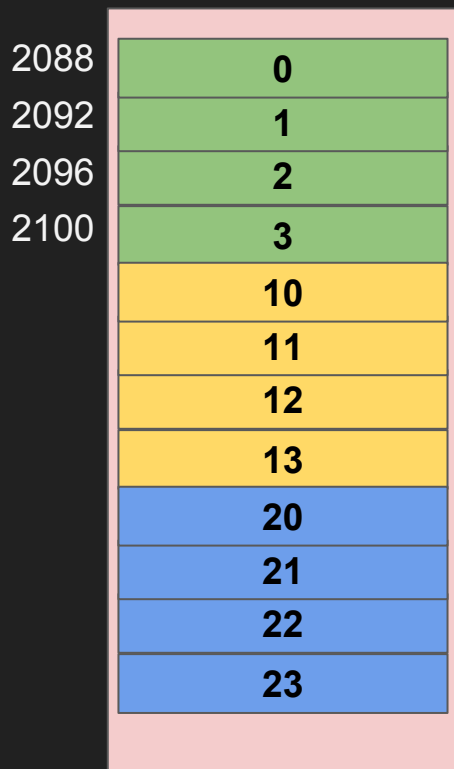
```
int a[m][n];
```

```
a[i][j]
```

```
offset = (n*i+j)*sizeof(int)
```



Memory



Row Major
(C,C++,Pascal, numpy)

	0	1	...	n-1
0	0	1	...	9
1	10	11	...	19
	⋮	⋮		⋮
	⋮	⋮		⋮
m-1	80	81	...	83

```
int a[m][n];
```

$a[i][j] : n*i + j$

to find $a[i][j]$ we need to know

- element size?
- m (no. of rows of a)?
- n (no. of columns of a)?

Other standards?



K. N. Toosi
University of Technology

- Column major?
 - `a[i][j] : ?`

	1	2	3	4
1	0	1	2	3
2	10	11	12	13
3	20	21	22	23

Other standards?



K. N. Toosi
University of Technology

- Column major?
 - $a[i][j] : 4*i+j$

	1	2	3	4
1	0	1	2	3
2	10	11	12	13
3	20	21	22	23

Other standards?



K. N. Toosi
University of Technology

- Column major?
 - $a[i][j] : 4*i+j$
- Index starting at 1
 - $a[i][j] : ?$

	1	2	3	4
1	0	1	2	3
2	10	11	12	13
3	20	21	22	23

Other standards?



K. N. Toosi
University of Technology

- Column major?
 - $a[i][j] : 4*i+j$
- Index starting at 1
 - $a[i][j] : 4*i+j+1$

	1	2	3	4
1	0	1	2	3
2	10	11	12	13
3	20	21	22	23

Example: Print a 2D Array



```
#include <stdio.h>

int print2Darray(int a[][6], int, int);

int a[4][6] = {{10, 20, 30, 40, 50, 60},
               {11, 21, 31, 41, 51, 61},
               {12, 22, 32, 42, 52, 62},
               {14, 24, 34, 44, 54, 64}};

int main() {
    print2Darray(a, 4, 6);
}

int print2Darray(int a[][6], int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%d,", a[i][j]);

        putchar('\n');
    }
}
```

Example: Print a 2D Array



K. N. Toosi
University of Technology

```
#include <stdio.h>

int print2Darray(int a[][6], int, int);

int a[4][6] = {{10, 20, 30, 40, 50, 60},
               {11, 21, 31, 41, 51, 61},
               {12, 22, 32, 42, 52, 62},
               {14, 24, 34, 44, 54, 64}};

int main() {
    print2Darray(a, 4, 6);
}

int print2Darray(int a[][6], int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%d,", a[i][j]);

        putchar('\n');
    }
}
```

```
nasihatkon@kntu:code$ gcc print2DArray.c && ./a.out
10,20,30,40,50,60,
11,21,31,41,51,61,
12,22,32,42,52,62,
14,24,34,44,54,64,
```

Example: Print a 2D Array



```
#include <stdio.h>

int print2Darray(int a[][6], int, int);

int a[4][6] = {{10, 20, 30, 40, 50, 60},
               {11, 21, 31, 41, 51, 61},
               {12, 22, 32, 42, 52, 62},
               {14, 24, 34, 44, 54, 64}};

int main() {
    print2Darray(a, 4, 6);
}

int print2Darray(int a[][6], int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%d,", a[i][j]);

        putchar('\n');
    }
}
```

```
%include "asm_io.inc"
segment .data

array:  dd    10, 20, 30, 40, 50, 60
        dd    11, 21, 31, 41, 51, 61
        dd    12, 22, 32, 42, 52, 62
        dd    14, 24, 34, 44, 54, 64

segment .text
global asm_main

asm_main:
    pusha
```

Example: Print a 2D Array



```
#include <stdio.h>

int print2Darray(int a[][6], int, int);

int a[4][6] = {{10, 20, 30, 40, 50, 60},
               {11, 21, 31, 41, 51, 61},
               {12, 22, 32, 42, 52, 62},
               {14, 24, 34, 44, 54, 64}};

int main() {
    print2Darray(a, 4, 6);
}

int print2Darray(int a[][6], int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%d,", a[i][j]);

        putchar('\n');
    }
}
```

segment .data

print2DArray1.asm

```
array: dd 10, 20, 30, 40, 50, 60
        dd 11, 21, 31, 41, 51, 61
        dd 12, 22, 32, 42, 52, 62
        dd 14, 24, 34, 44, 54, 64
```

segment .text

```
:
; print2DArray (array, m, n)
push 6 ; no of columns
push 4 ; no of rows
push array ; address of array
call print2DArray
add esp, 12
```

Example: Print a 2D Array



```
#include <stdio.h>

int print2Darray(int a[][6], int, int);

int a[4][6] = {{10, 20, 30, 40, 50, 60},
               {11, 21, 31, 41, 51, 61},
               {12, 22, 32, 42, 52, 62},
               {14, 24, 34, 44, 54, 64}};

int main() {
    print2Darray(a, 4, 6);
}

int print2Darray(int a[][6], int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%d,", a[i][j]);

        putchar('\n');
    }
}
```

segment .data

print2DArray3.asm

```
array: dd 10, 20, 30, 40, 50, 60
        dd 11, 21, 31, 41, 51, 61
        dd 12, 22, 32, 42, 52, 62
        dd 14, 24, 34, 44, 54, 64
```

segment .text

```
:
; print2DArray (array, m, n)
push 6 ; no of columns
push 4 ; no of rows
push array ; address of array
call print2DArray
add esp, 12
```


Example: Print a 2D Array



```
int print2Darray(int a[][6], int m, int n) {  
    for (int i = 0; i < m; i++) {  
        for (int j = 0; j < n; j++)  
            printf("%d,", a[i][j]);  
  
        putchar('\n');  
    }  
}
```

```
delim: db " , ", 0
```

```
%define ARRAY [ebp+8]
```

```
%define M [ebp+12]
```

```
%define N [ebp+16]
```

```
; print2Darray(ARRAY, M, N)
```

```
print2Darray:
```

```
    push ebp
```

```
    mov ebp, esp
```

```
    mov ebx, ARRAY
```

```
    mov esi, 0
```

```
loop1:
```

```
    cmp esi, M
```

```
    jge endloop1
```

```
    mov edi, 0
```

```
loop2:
```

```
    cmp edi, N
```

```
    jge endloop2
```

```
    :
```

```
; index = esi*N+edi
```

```
    mov eax, N
```

```
    mul esi
```

```
    add eax, edi
```

```
    mov eax, [ebx+4*eax]
```

```
    call print_int
```

```
    mov eax, delim
```

```
    call print_string
```

```
    inc edi
```

```
    jmp loop2
```

```
endloop2:
```

```
    mov al, 10
```

```
    call print_char
```

```
    inc esi
```

```
    jmp loop1
```

```
endloop1:
```

```
    mov esp, ebp
```

```
    pop ebp
```

```
    ret
```

```
print2Darray3.asm
```

Example: Print a 2D Array



```
int print2Darray(int a[][6], int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%d,", a[i][j]);

        putchar('\n');
    }
}
```

```
delim: db " , " , 0 ←
```

```
%define ARRAY [ebp+8]
%define M     [ebp+12]
%define N     [ebp+16]
```

```
; print2Darray(ARRAY, M, N)
```

```
print2Darray:
```

```
    push ebp
    mov  ebp, esp
```

```
    mov ebx, ARRAY
    mov esi, 0
```

```
loop1:
```

```
    cmp esi, M
    jge endloop1
```

```
    mov edi, 0
```

```
loop2:
```

```
    cmp edi, N
    jge endloop2
```

```
    :
```

```
; index = esi*N+edi
```

```
    mov eax, N
    mul esi
    add eax, edi
```

```
    mov eax, [ebx+4*eax]
    call print_int
    mov eax, delim
    call print_string
```

```
    inc edi
    jmp loop2
```

```
endloop2:
```

```
    mov al, 10
    call print_char
```

```
    inc esi
    jmp loop1
```

```
endloop1:
```

```
    mov esp, ebp
    pop ebp
    ret
```

print2Darray3.asm

Example: Print a 2D Array



```
int print2Darray(int a[][6], int m, int n) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++)
            printf("%d,", a[i][j]);

        putchar('\n');
    }
}
```

```
nasihatkon@kntu:code$ ./run print2DArray3
10, 20, 30, 40, 50, 60,
11, 21, 31, 41, 51, 61,
12, 22, 32, 42, 52, 62,
14, 24, 34, 44, 54, 64,
```

```
delim: db ", ", 0
```

```
%define ARRAY [ebp+8]
```

```
%define M [ebp+12]
```

```
%define N [ebp+16]
```

```
; print2DArray(ARRAY, M, N)
```

```
print2DArray:
```

```
    push ebp
```

```
    mov ebp, esp
```

```
    mov ebx, ARRAY
```

```
    mov esi, 0
```

```
loop1:
```

```
    cmp esi, M
```

```
    jge endloop1
```

```
; index = esi*N+edi
```

```
    mov eax, N
```

```
    mul esi
```

```
    add eax, edi
```

```
    mov eax, [ebx+4*eax]
```

```
    call print_int
```

```
    mov eax, delim
```

```
    call print_string
```

```
    inc edi
```

```
    jmp loop2
```

```
endloop2:
```

```
    mov al, 10
```

```
    call print_char
```

```
    inc esi
```

```
    jmp loop1
```

```
endloop1:
```

```
    mov esp, ebp
```

```
    pop ebp
```

```
    ret
```

```
print2DArray3.asm
```



Example: Print a 2D Array

```
int print2Darray(int a[][6], int m, int n) {  
    for (int i = 0; i < m; i++) {  
        for (int j = 0; j < n; j++)  
            printf("%d,", a[i][j]);  
  
        putchar('\n');  
    }  
}
```

```
delim: db " , ", 0
```

```
%define ARRAY [ebp+8]
```

```
%define M [ebp+12]
```

```
%define N [ebp+16]
```

```
; print2DArray(ARRAY, M, N)
```

Make it faster?

```
mov esi, 0
```

```
loop1:
```

```
cmp esi, M
```

```
jge endloop1
```

```
; index = esi*N+edi
```

```
mov eax, N
```

```
mul esi
```

```
add eax, edi
```

```
mov eax, [ebx+4*eax]
```

```
call print_int
```

```
mov eax, delim
```

```
call print_string
```

```
inc edi
```

```
jmp loop2
```

```
endloop2:
```

```
mov al, 10
```

```
call print_char
```

```
inc esi
```

```
jmp loop1
```

```
endloop1:
```

```
mov esp, ebp
```

```
pop ebp
```

```
ret
```

```
nasihatkon@kntu:code$ ./run print2DArray3
```

```
10, 20, 30, 40, 50, 60,  
11, 21, 31, 41, 51, 61,  
12, 22, 32, 42, 52, 62,  
14, 24, 34, 44, 54, 64,
```

```
print2DArray3.asm
```

Example: Print a 2D Array



print2DArray3.asm

```
mov ebx, ARRAY
mov esi, 0
loop1:
  cmp esi, M
  jge endloop1

  mov edi, 0
loop2:
  cmp edi, N
  jge endloop2

  ; index = esi*N+edi
  mov eax, N
  mul esi
  add eax, edi

  mov eax, [ebx+4*eax]
  call print_int
  mov eax, delim
  call print_string

  :
```



print2DArray4.asm

```
mov ebx, ARRAY
mov esi, 0
loop1:
  cmp esi, M
  jge endloop1

  mov edi, 0
loop2:
  cmp edi, N
  jge endloop2

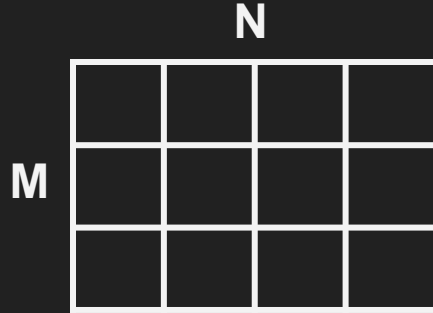
  mov eax, [ebx]
  call print_int
  mov eax, delim
  call print_string

  add ebx, 4

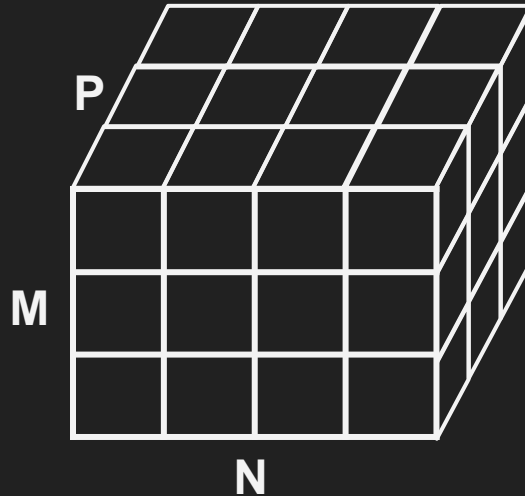
  :
```

3D Arrays

- 2D array
 - $M \times N$



- 3D array
 - $M \times N \times P$

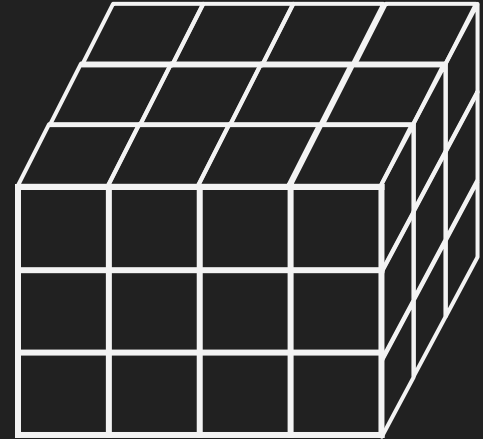


ND Arrays: Row-major vs. Column-major



K. N. Toosi
University of Technology

- What does row-major and column-major mean?
 - Matlab vs Numpy ND-arrays

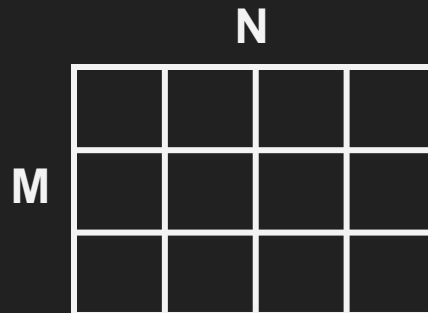




3D Arrays

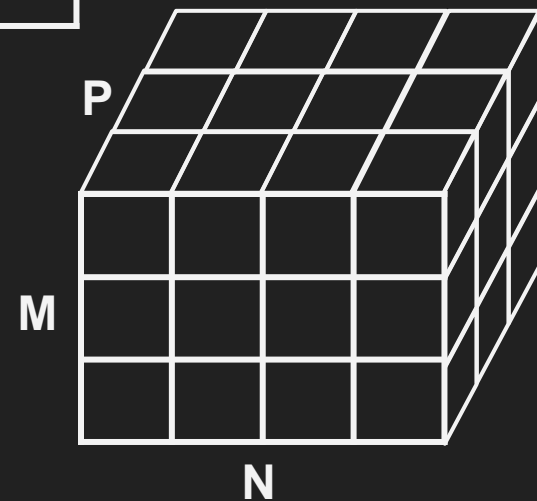
- **2D array (row-major)**

- $M \times N$
- M 1D arrays (rows) of size N
- $\text{Array}[i, j] = \text{Array}[N*i + j]$



- **3D array (row-major)**

- $M \times N \times P$
- M 2D arrays of dimension $N \times P$
 - $\text{Array}[i, j, k] = \text{Array}[i, j*P + k] = \text{Array}[i * N*P + j * P + k]$

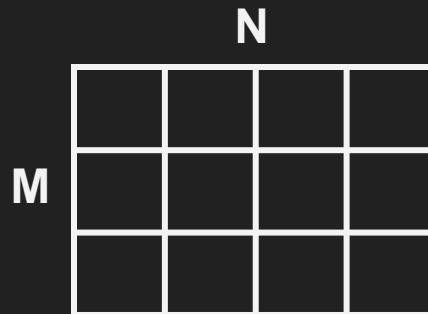




3D Arrays

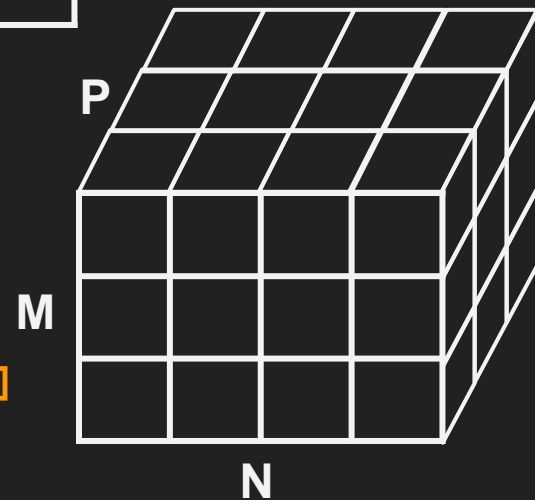
- **2D array (row-major)**

- $M \times N$
- M 1D arrays (rows) of size N
- $\text{Array}[i, j] = \text{Array}[N*i + j]$



- **3D array (row-major)**

- $M \times N \times P$
- M 2D arrays of dimension $N \times P$
 - $\text{Array}[i, j, k] = \text{Array}[i, j*P + k] = \text{Array}[i * N*P + j * P + k]$
- An $M \times N$ 2D array of 1D arrays of size P
 - $\text{Array}[i, j, k] = \text{Array}[i, j][k]$
 $= \text{Array}[i*N + j][k] = \text{Array}[(i * N + j) * P + k]$

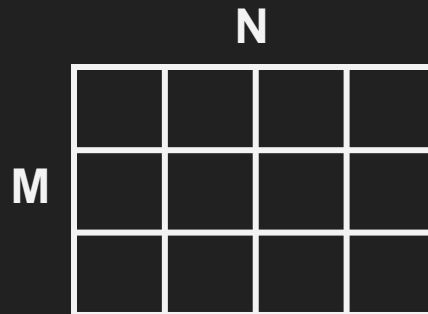




3D Arrays

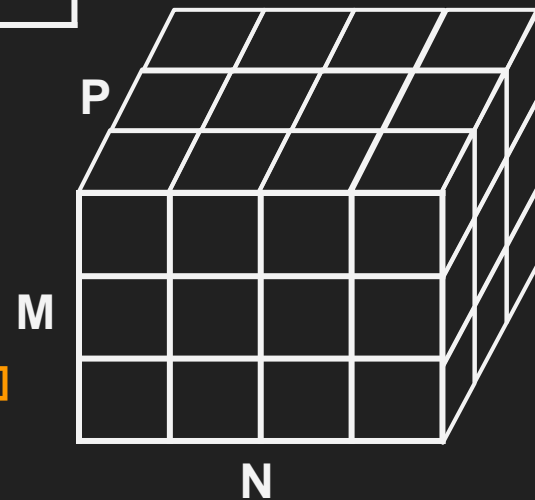
- **2D array (row-major)**

- $M \times N$
- M 1D arrays (rows) of size N
- $\text{Array}[i, j] = \text{Array}[N*i + j]$



- **3D array (row-major)**

- $M \times N \times P$
- M 2D arrays of dimension $N \times P$
 - $\text{Array}[i, j, k] = \text{Array}[i, j*P + k] = \text{Array}[i * N*P + j * P + k]$
- An $M \times N$ 2D array of 1D arrays of size P
 - $\text{Array}[i, j, k] = \text{Array}[i, j][k]$
 $= \text{Array}[i*N + j][k] = \text{Array}[(i * N + j) * P + k]$



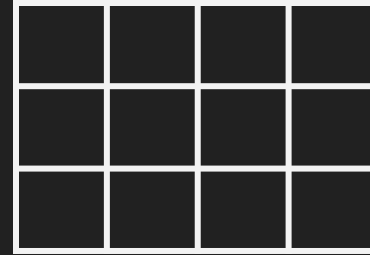
number of + and * operations?

ND Arrays



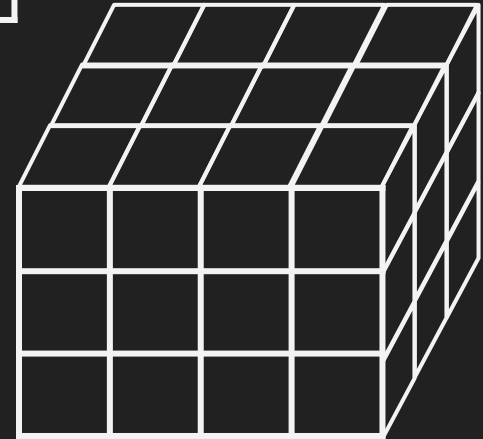
- **2D array (row-major)**

- $M \times N$
- M 1D arrays (rows) of size N
- $\text{Array}[i, j] = \text{Array}[N*i + j]$



- **3D array (row-major)**

- $M \times N \times P$
- M 2D arrays of dimension $N \times P$
 - $\text{Array}[i, j, k] = \text{Array}[i, j*P + k] = \text{Array}[i * N*P + j * P + k]$
- An $M \times N$ 2D array of 1D arrays of size P
 - $\text{Array}[i, j, k] = \text{Array}[i*N + j, k] = \text{Array}[(i * N + j) * P + k]$



- **ND array (row-major)**

- $M_1 \times M_2 \times \dots \times M_n$
- $\text{Array}[i_1, i_2, \dots, i_n] = \text{Array}[M_2 * \dots * M_n * i_1 + M_3 * \dots * M_n * i_2 + \dots + M_n * i_{n-1} + i_n]$

ND Arrays



- ND array (row-major)
 - $M_1 \times M_2 \times \dots \times M_n$
 - $\text{Array}[i_1, i_2, \dots, i_n] = \text{Array}[M_2 * \dots * M_n * i_1 + M_3 * \dots * M_n * i_2 + \dots + M_n * i_{n-1} + i_n]$
- $j_1 = i_1$
- $j_2 = M_2 * j_1 + i_2$
- $j_3 = M_3 * j_2 + i_3$
- $:$
- $j_n = M_n * j_{n-1} + i_n$
- $\text{Array}[i_1, i_2, \dots, i_n] = \text{Array}[j_n]$

number of + and * operations?