Memory Ideas

BASIC ASSEMBLY

Assembly language programming
By xorpd

xorpd.net

Objectives

- We will see examples of interesting memory constructs:
  - Array of structures.
  - Two dimensional table.
  - Binary tree.

The memory is not just bytes

- The memory is made of bytes.
- Don’t let it limit your thinking.
- Big ideas could be implemented with little bytes.

Array of structs

```
struct DOG
    color     dd ?
    age       dd ?
ends
```

```
NUM_DOGS = 12
section '.bss' readable writeable
         my_dogs   db   NUM_DOGS*sizeof.DOG dup (?)

section '.text' code readable executable
start:
          ...
          ; Access dog number ecx
          mov       esi,my_dogs
          lea       esi,[esi + ecx*sizeof.DOG]

          mov       eax, dword [esi + DOG.color]
          mov       edx, dword [esi + DOG.age]
```

Higher dimensions

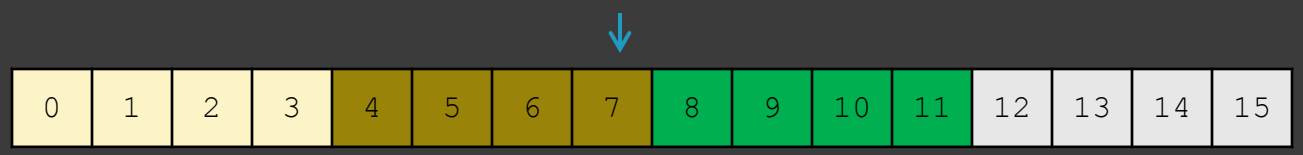
- Assume that we want to remember the multiplication table in memory.

X	1	2	3	4	…
1	1	2	3	4	
2	2	4	6	8	
3	3	6	9	12	
4	4	8	12	16	
…					…

- How can we store a two dimensional object in our one dimensional memory landscape?
  - We use our imagination!

Higher Dimensions (Cont.)

	0	1	2	3
0	0	4	8	12
1	1	5	9	13
2	2	6	10	14
3	3	7	11	15



- The cell in row 3 and column 1 is in location 7.
  - $1 \cdot 4 + 3 = 7$
- The cell in row r and column c is in location:
  - $c \cdot 4 + r$

Higher Dimensions (Cont.)

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- The cell in row 3 and column 1 is in location 13.
  - $1 + 3 \cdot 4 = 13$
- The cell in row r and column c is in location:
  - $c + r \cdot 4$

Higher Dimensions (Cont.)

- Where is 11?
  - $11 / 4 = 2$ (Row)
  - $11 \% 4 = 3$ (Column)
- Where is k?
  - $k / 4$ (Row)
  - $k \% 4$ (Column)

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
?

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 0

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ecx →

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 0

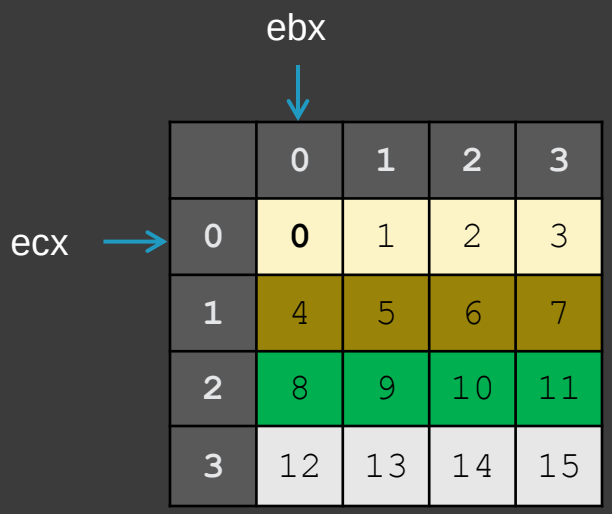
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 0

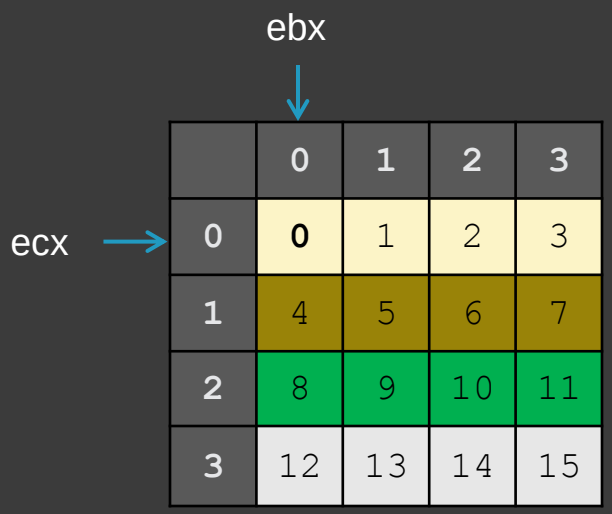
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 1

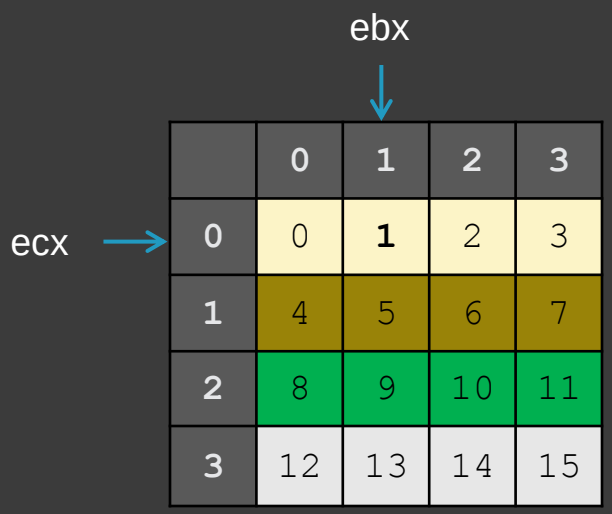
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 1

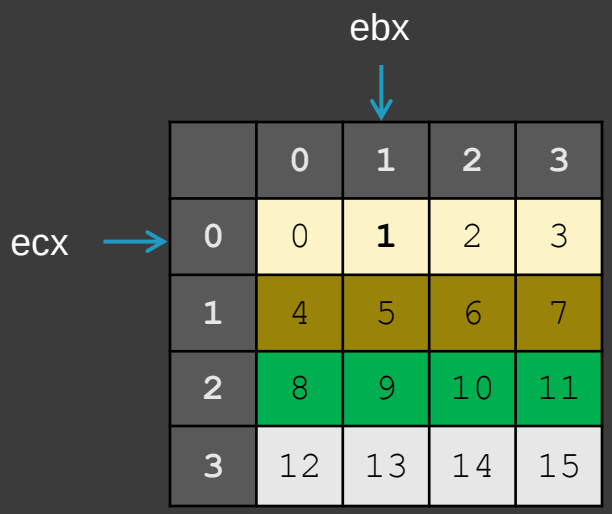
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 1

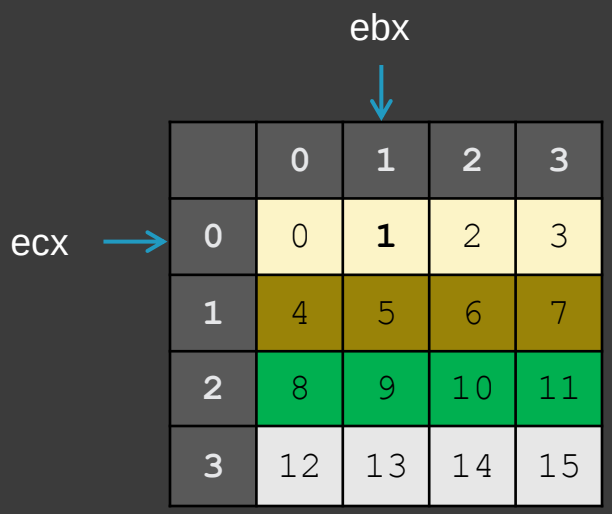
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 1

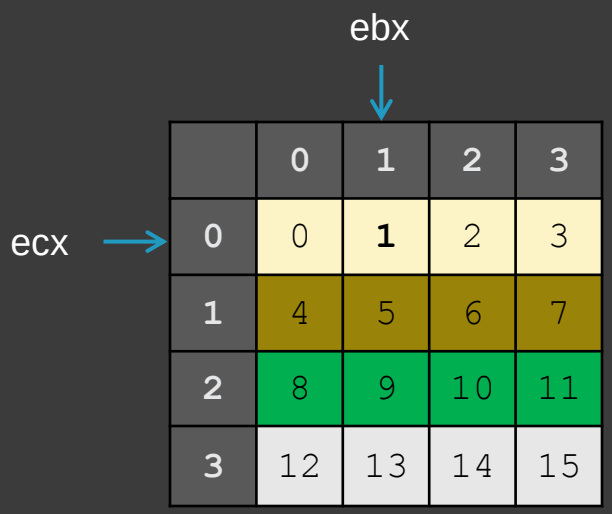
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 2

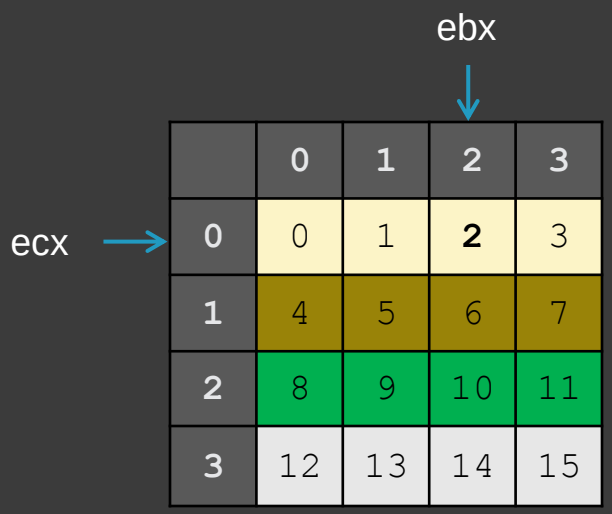
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 2

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx

ecx

	0	1	2	3
0	0	1	**2**	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 2

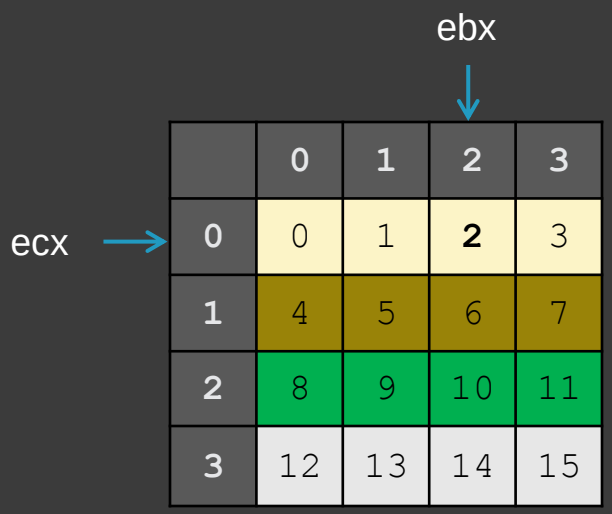
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 2

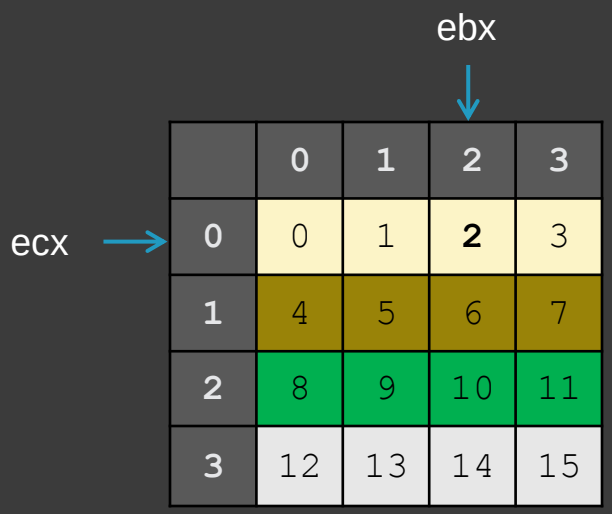
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 3

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx

ecx

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 3

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx

ecx

	0	1	2	3
0	0	1	2	**3**
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 3

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx

ecx

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 3

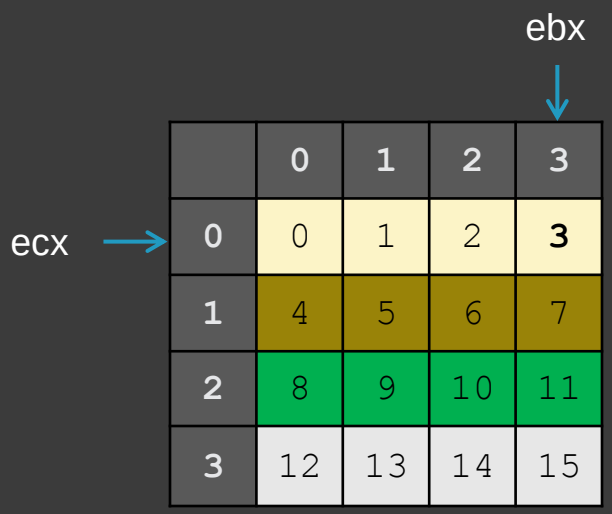
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 4

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx

ecx

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 4

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx

ecx

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 4

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

→   inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx ↓

ecx →

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 4

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx

ecx

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 4

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
→   jnz     next_row
```

ebx ↓

ecx →

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 4

Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```

ebx

ecx

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

esi
mul_tbl + 4 · 4

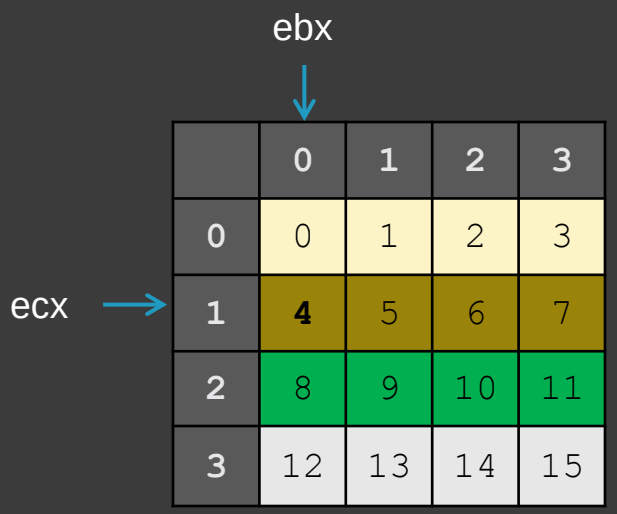
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 4

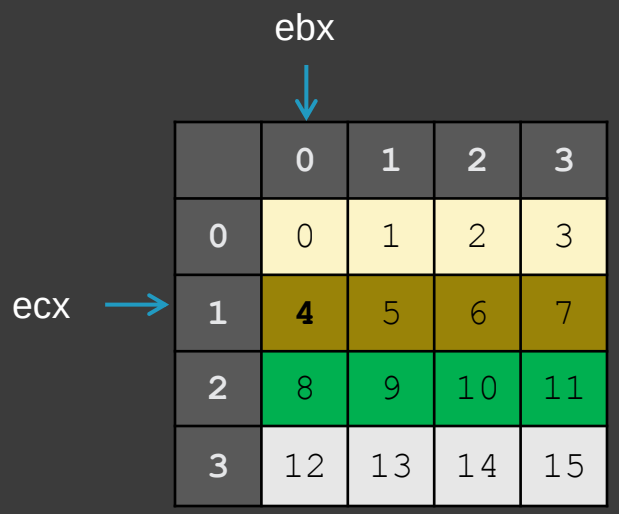
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 5

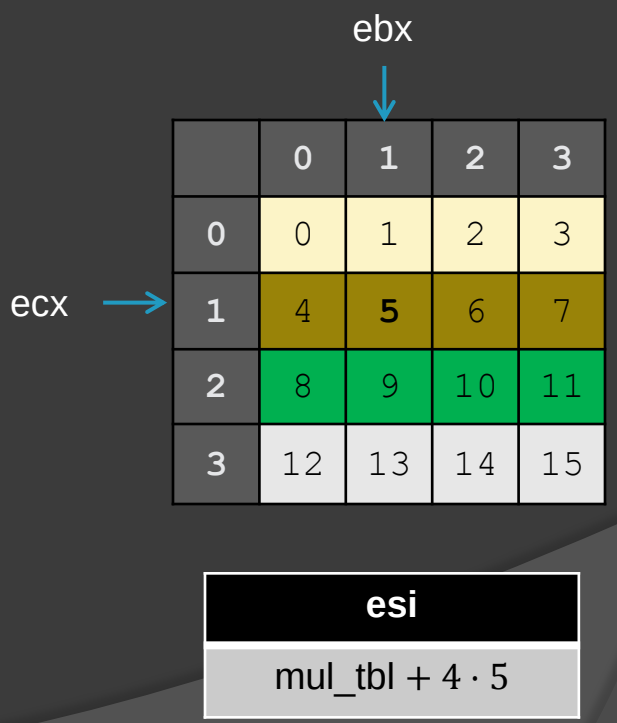
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



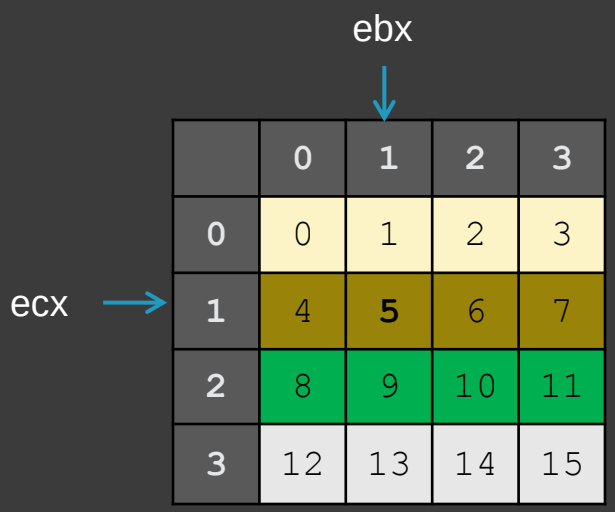
Multiplication table

- Declaring the table:

```
mul_tbl     dd      WIDTH*HEIGHT dup (?)
```

- Filling in the table:

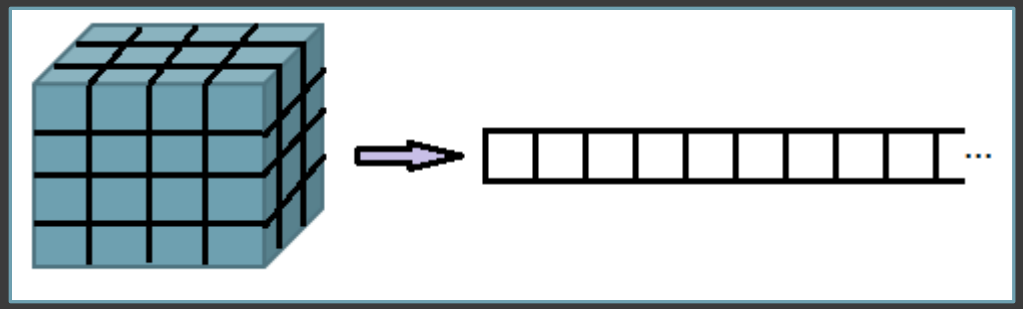
```
    mov     esi,mul_tbl ; cell ptr.
    mov     ecx,0   ; row counter.

next_row:
    mov     ebx,0   ; Column counter.
next_column:
    mov     eax,ecx
    mul     ebx
    mov     dword [esi],eax

    add     esi,4
    inc     ebx
    cmp     ebx,WIDTH
    jnz     next_column

    inc     ecx
    cmp     ecx,HEIGHT
    jnz     next_row
```



esi
mul_tbl + 4 · 5

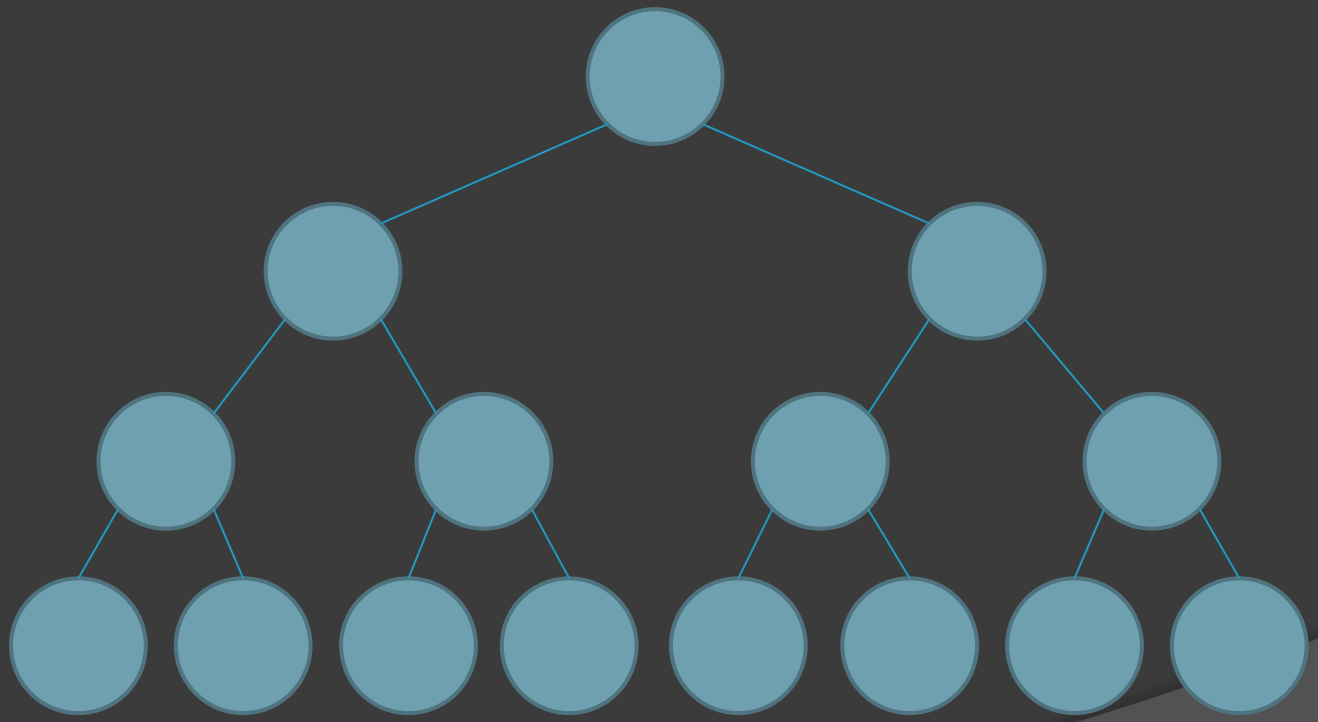
More dimensions

- The same techniques apply to more than 2 dimensions.
- Think about how to translate (x,y,z) coordinates into a linear memory location.



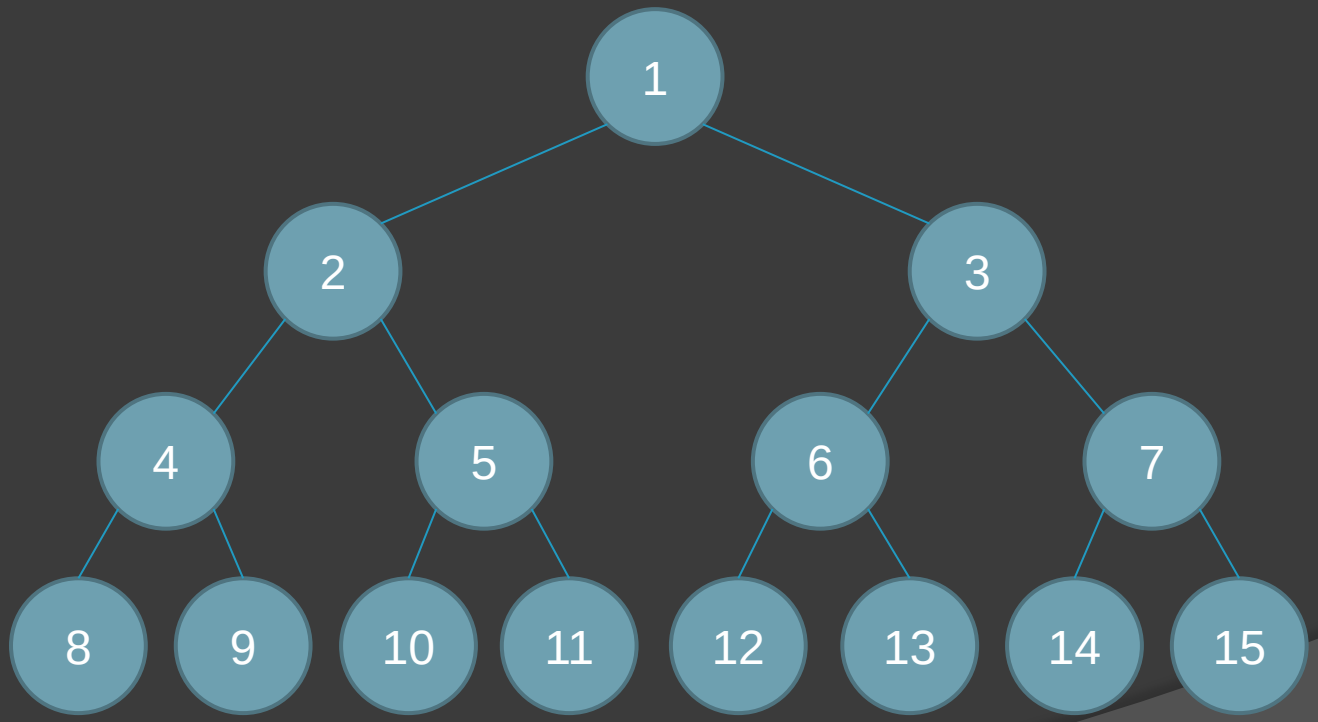
Binary tree

- Every node has at most two sons.
- How to represent it using linear memory?



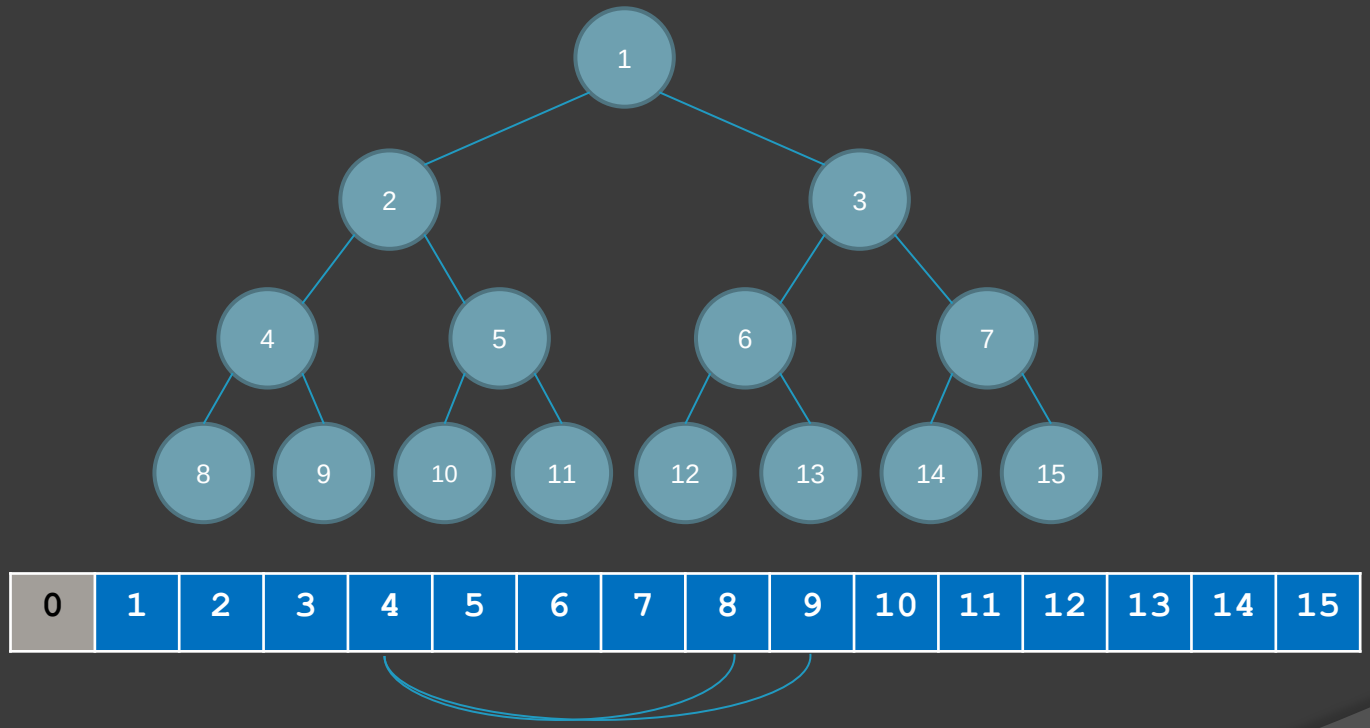
Binary tree (Cont.)

- Let's number the nodes:



Binary tree (Cont.)

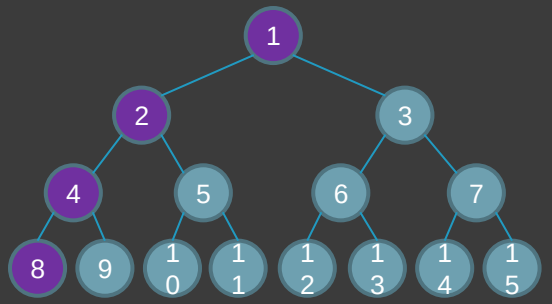
- Finally we flatten the tree into linear memory:



- k's sons are: $2k, 2k+1$.

Binary tree (Cont.)

- Traversing the tree.

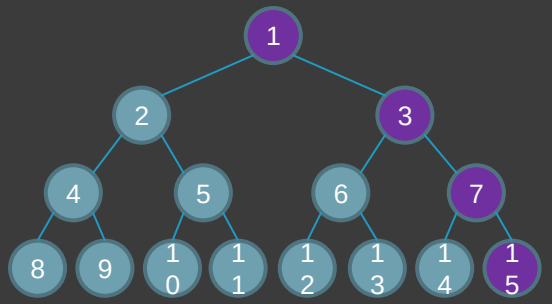



```
TREE_SIZE = 15
my_tree   dd   TREE_SIZE+1 dup (?)
```

```
...
    mov     esi,my_tree
    mov     ecx,1 ; Root.

next_son:
    ; print contents:
    mov     eax,dword [esi + 4*ecx]
    call    print_eax
    ; Calculate the left son's location:
    lea     ecx,[2*ecx]
    cmp     ecx,TREE_SIZE
    jbe     next_son
```

Binary tree (Cont.)

- Traversing the tree.




```
TREE_SIZE = 15
my_tree   dd   TREE_SIZE+1 dup (?)
```

```
...
    mov     esi,my_tree
    mov     ecx,1 ; Root.

next_son:
    ; print contents:
    mov     eax,dword [esi + 4*ecx]
    call    print_eax
    ; Calculate the right son's location:
    lea     ecx,[2*ecx+1]
    cmp     ecx,TREE_SIZE
    jbe     next_son
```

Summary

- We have seen the following memory constructs:
  - Array of structures.
  - Two dimensional table.
  - Binary Tree.
- Much more could be achieved.
  - Use your imagination.

Exercises

- Code reading.
- Code writing.
- Have fun :)