Arrays

- a convenient organization of data
- efficient to access elements, because the array is a contiguous allocation of memory
- architectures generally offer no assembly language support for array element access

* except for the x86 do-everything addressing mode
int ar[5];

0 6 -1 15 102

in memory

0
6
-1
15
102

addr of \( \text{ar}[i] \) = \( \text{ar} + 4 \ast i \)

→ assumes 4-byte ints and byte addressable

→ probably assumes that \( \text{ar} \) is assigned an aligned address
Writing code that works correctly for any size of an integer... use

sizeof(int)
Alignment

as an example, assume 4-byte ints & byte addressable.

Where is the int at address 35?

shown w/ Little Endian numbering

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

start at byte addr

truncation (logical AND w/ 0xfffffffc)
int i, a[100];
for (i=0; i<100; i++)
    a[i] = -1;

movl $0, %edi
leal a, %ebx
movl $100, %edx
.L3: cmpq %edi, %edx
jle .L2
    movl $-1, (%ebx, %edi, 4)
    inc %edi
jmp .L3
.L2:
** movl $-1 (%ebp, %edi, 4)  
   shift left by 2 bits

1. calculates address
   shift
   add

2. store to address
change source code a little.

\[ a[i] = -1; \]

\[ a[i] = a[i] - 1; \]

decl (\%ebx, \%edi, 4)

1. calculate address
   shift
   add

2. load from address

3. subtract 1

4. store to address
2D arrays

```c
int DD[3][4];
```

HLL sets storage order within memory

row major
### Column Major

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### For Row Major Order

$$\text{addr of } [\text{row}][\text{col}] = \text{base} + \text{row} \times \#\text{cols} \times \frac{\text{size of elem in bytes}}{\text{bytes in row}} + \text{col} \times \frac{\text{size of elem in bytes}}{\text{bytes in row}}$$