Section 5: Arrays & Other Data Structures

- Array allocation and access in memory
- Multi-dimensional or nested arrays
- Multi-level arrays
- Other structures in memory
- Data structures and alignment

Multi-Level Array Example

```
zip_dig cmu = { 1, 5, 2, 1, 3 };
zip_dig uw  = { 9, 8, 1, 9, 5 };
zip_dig ucb = { 9, 4, 7, 2, 0 };
```

```
#define UCOUNT 3
int *univ[UCOUNT] = {uw, cmu, ucb};
```

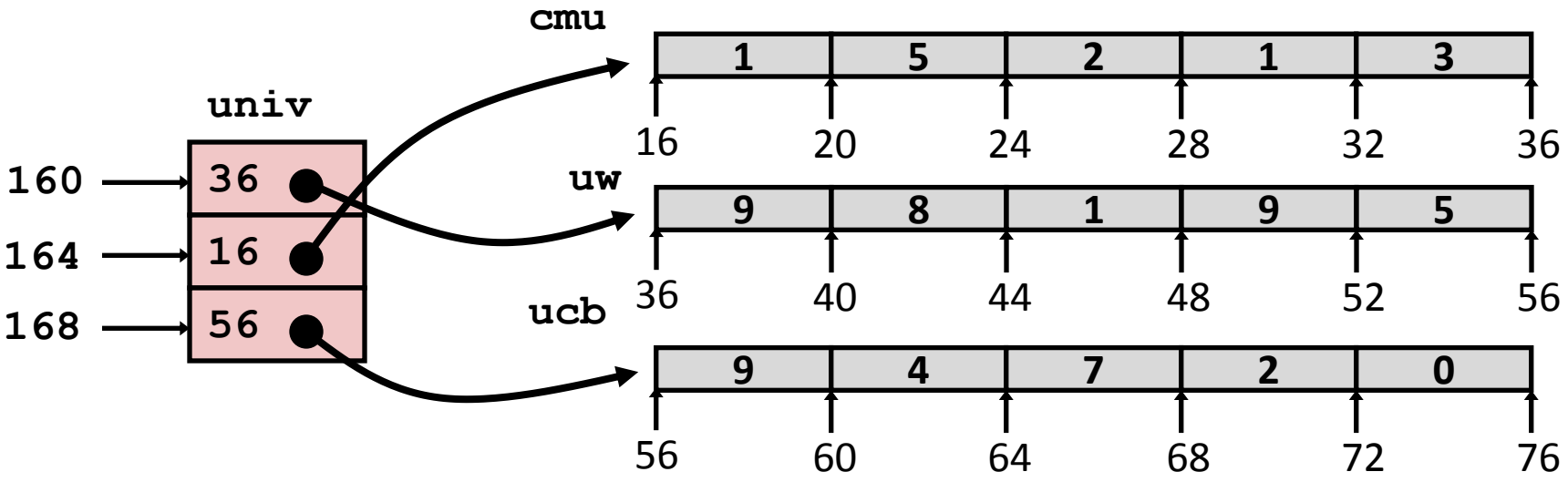
Same thing as a 2D array?

Multi-Level Array Example

```
zip_dig cmu = { 1, 5, 2, 1, 3 };
zip_dig uw  = { 9, 8, 1, 9, 5 };
zip_dig ucb = { 9, 4, 7, 2, 0 };
```

```
#define UCOUNT 3
int *univ[UCOUNT] = {uw, cmu, ucb};
```

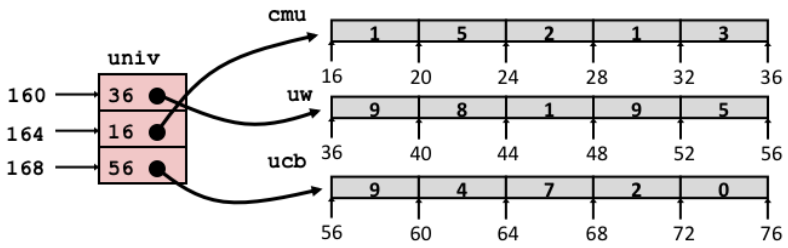
- Variable `univ` denotes array of 3 elements
- Each element is a pointer
  - 4 bytes
- Each pointer points to array of `int`s



Note: this is how Java represents multi-dimensional arrays.

Element Access in Multi-Level Array

```
int get_univ_digit
   (int index, int dig)
{
   return univ[index][dig];
}
```



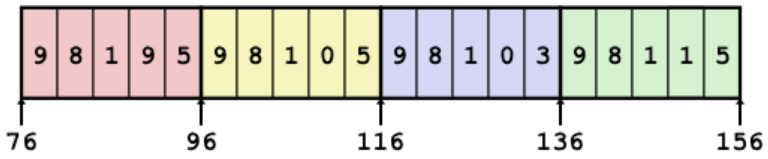

```
   # %ecx = index
   # %eax = dig
   leal 0(,%ecx,4),%edx     # 4*index
   movl univ(%edx),%edx     # Mem[univ+4*index]
   movl (%edx,%eax,4),%eax  # Mem[...+4*dig]
```

- **Computation (IA32)**
  - Element access **`Mem[Mem[univ+4*index]+4*dig]`**
  - Must do two memory reads
    - First get pointer to row array
    - Then access element within array

Array Element Accesses

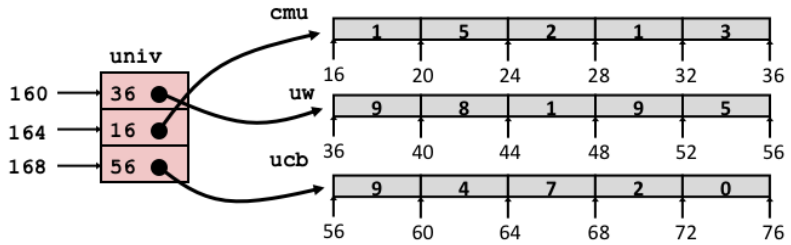
Nested array

```
int get_sea_digit
  (int index, int dig)
{
  return sea[index][dig];
}
```



Multi-level array

```
int get_univ_digit
  (int index, int dig)
{
  return univ[index][dig];
}
```

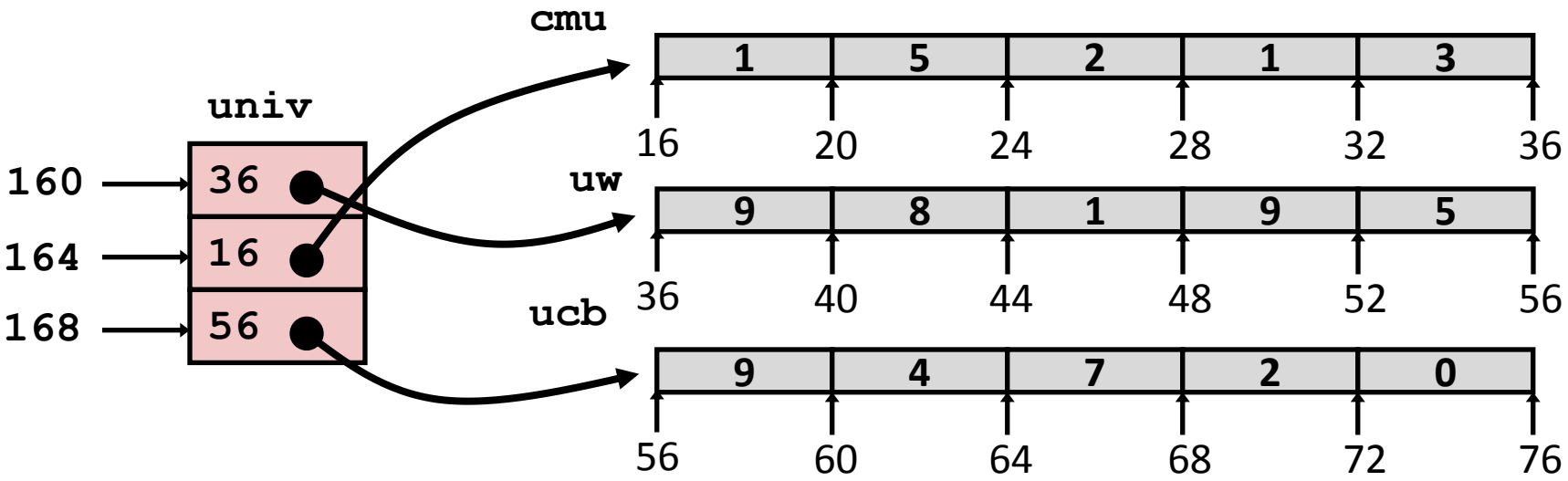


Access looks similar, but it isn't:

`Mem[sea+20*index+4*dig]`

`Mem[Mem[univ+4*index]+4*dig]`

Strange Referencing Examples



Reference	Address	Value	Guaranteed?
`univ[2][3]`	`56+4*3 = 68`	2	Yes
`univ[1][5]`	`16+4*5 = 36`	9	No
`univ[2][-1]`	`56+4*-1 = 52`	5	No
`univ[3][-1]`	`??`	`??`	No
`univ[1][12]`	`16+4*12 = 64`	7	No

- Code does not do any bounds checking
- Location of each lower-level array in memory is not guaranteed

Arrays in C

- Contiguous allocations of memory
- No bounds checking
- Can usually be treated like a pointer to first element (elements are offset from start of array)
- Nested (multi-dimensional) arrays are contiguous in memory (row-major order)
- Multi-level arrays are not contiguous (pointers used between levels)