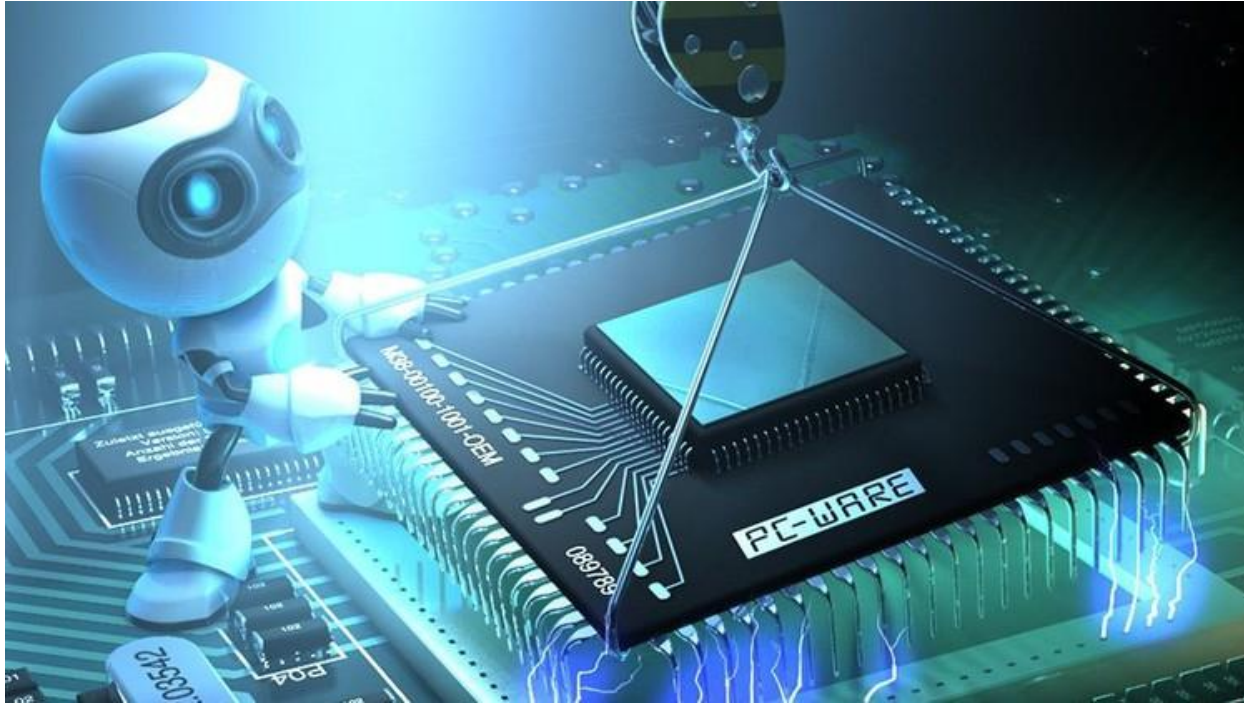


# Vector Practise

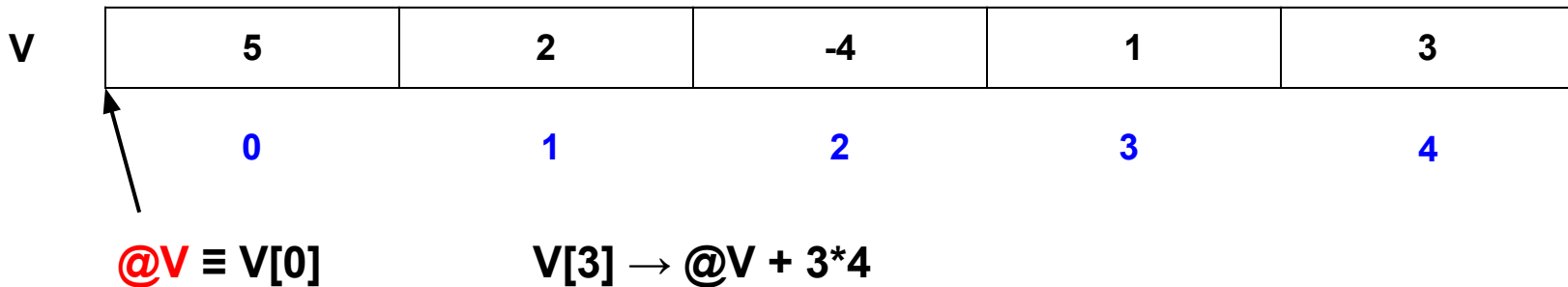


# Structured Data Types



## Vectors

- Declaration in C:  
`type name[size];` // indexed starting at 0
- Storage in consecutive memory locations
  - Access element  $V[i]$ : **@start V + i\*size** (size: size of the elements of V)

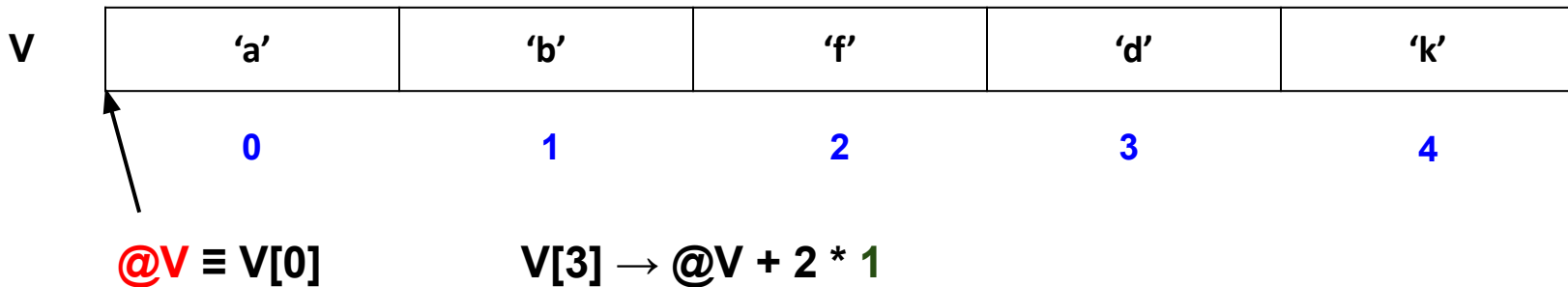


# Structured Data Types



## Vectors

- Declaration in C:  
`type name[size];` // indexed starting at 0
- Storage in consecutive memory locations
  - Access element  $V[i]$ : **@start V + i\*size** (size: size of the elements of V)



# Structured Data Types



## Vectors

- Examples:

Declaration in C	Size of Element	Size of Vector	@element i
char <b>A</b> [12];	1B	12B	@start A + i
char * <b>B</b> [80];	4B	320B	@start B + 4·i
double <b>C</b> [1024];	8B	8KB	@start C + 8·i
int * <b>D</b> [5];	4B	20B	@start D + 4·i
int <b>E</b> [100];	4B	400B	@start E + 4·i

# Vectors



## Example:

```
int Vi(int V[100], int i) {
    return V[i];
}
```

$\%ecx \leftarrow @V$

$\%edx \leftarrow i$

$@V + i * 4$

## Traduction:

```
Vi: pushl %ebp
    movl %esp, %ebp
    movl 8(%ebp), %ecx    # @V → 8[%ebp]
    movl 12(%ebp), %edx   # i → 12[%ebp]
    movl (%ecx, %edx, 4), %eax
    popl %ebp            # result in %eax
    ret
```

$\%ecx + \%edx * 4 \equiv @V + i * 4$