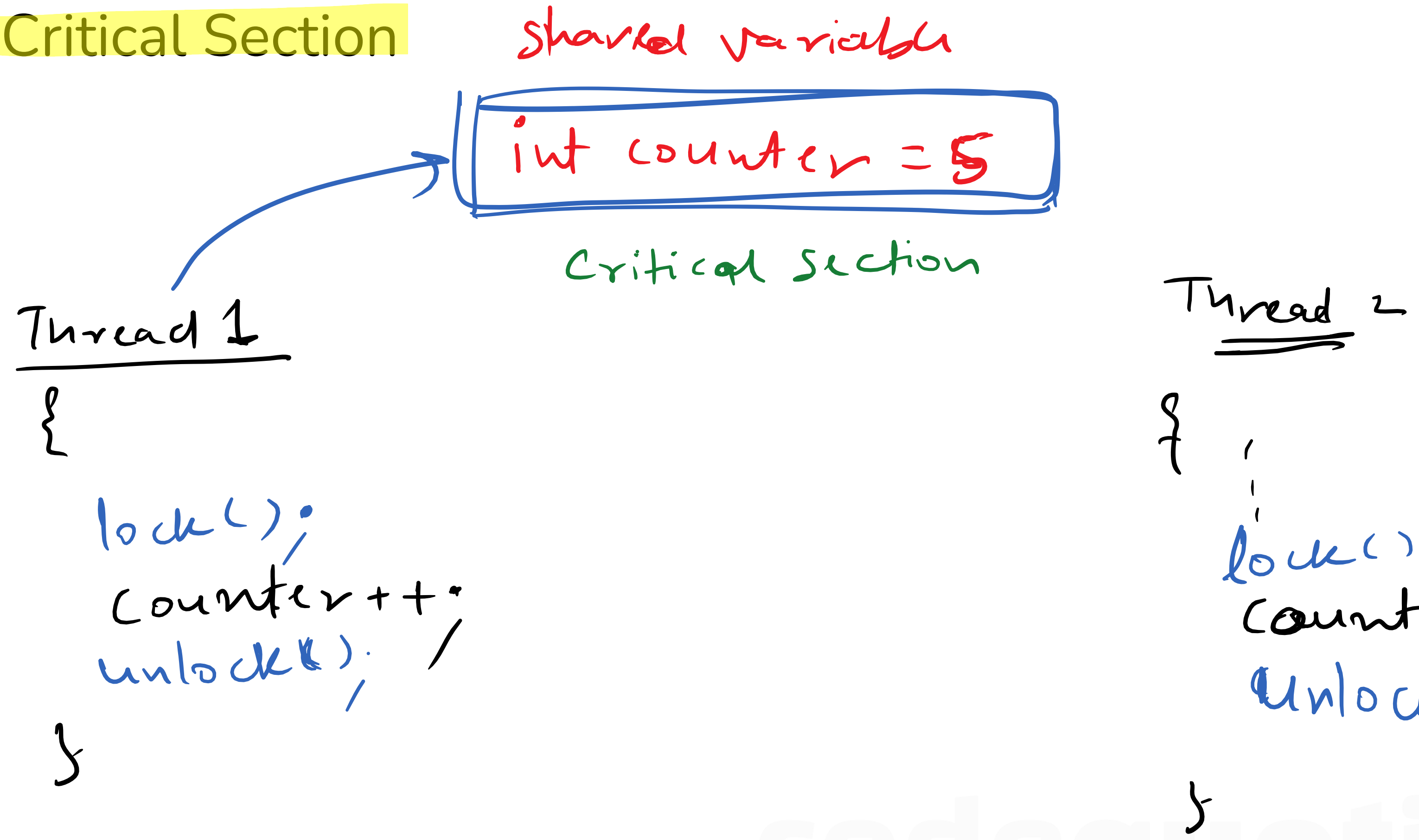
Critical Section



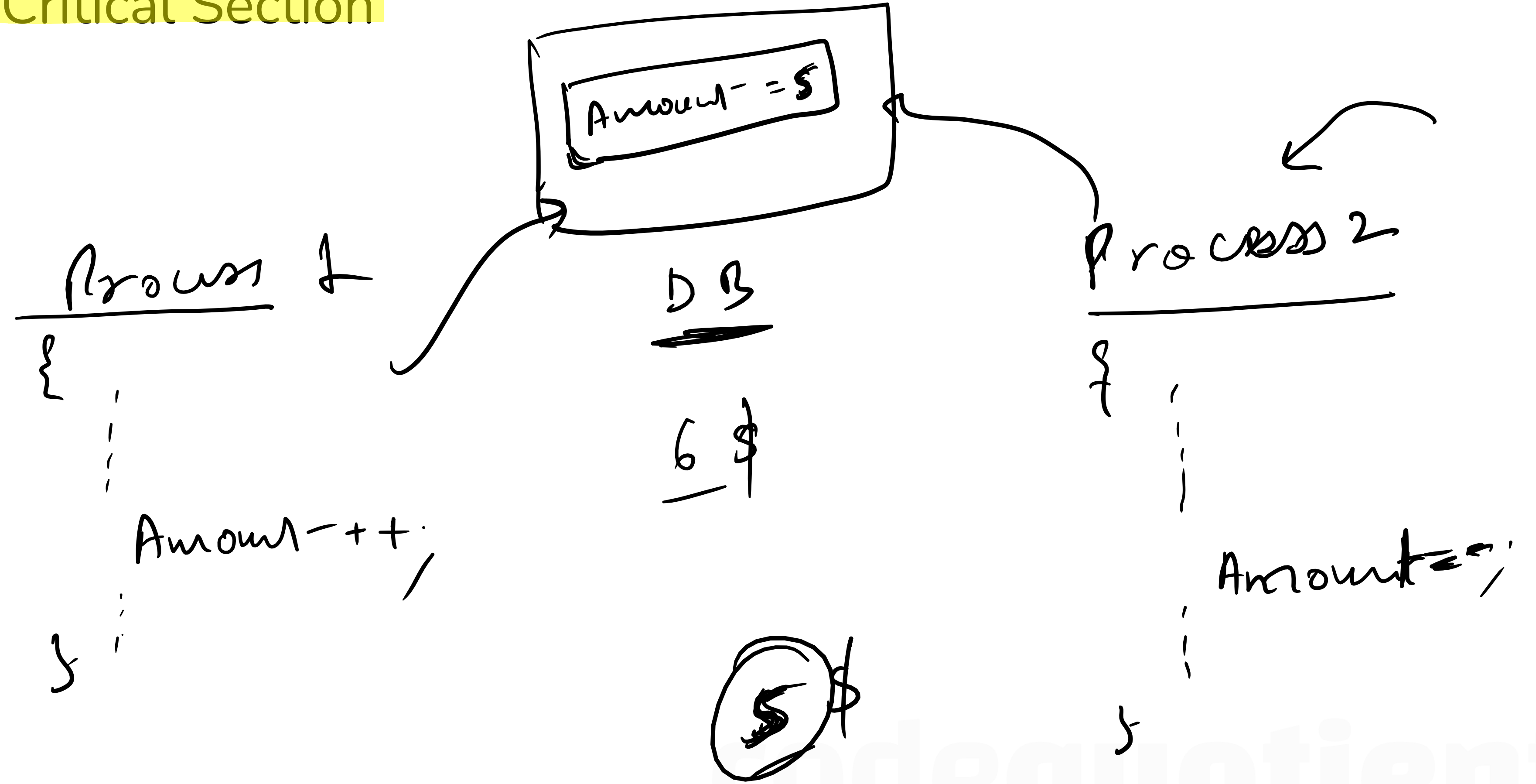
Thread 1

```
{
    lock();
    counter++;
    unlock();
}
```

Thread 2

```
{
    lock();
    counter--;
    Unlock();
}
```

Critical Section



Critical Section

shared variable

$\boxed{\text{int counter} = 5}$

critical section

Thread 1
```
{
    :
    counter++;
    :
}
```

Thread 2
```
{
    :
    counter--;
    :
}
```

$R_1 \leftarrow \text{counter}$

$R_1 \leftarrow R_1 + 1$

$\text{counter} \leftarrow R_1$

$R_2 \leftarrow \text{counter}$

$R_2 \leftarrow R_2 - 1$

$\text{counter} \leftarrow R_2$

$\boxed{5}$

$R_1 \leftarrow \text{counter}$

$R_2 \leftarrow \text{counter}$

$R_2 \leftarrow R_2 - 1$

$\text{counter} \leftarrow R_2$

$R_1 \leftarrow R_1 + 1$

$\text{counter} \leftarrow R_1$

$\boxed{6}$

$R_2 \leftarrow \text{counter}$

$R_1 \leftarrow \text{counter}$

$R_1 \leftarrow R_1 + 1$

6 $\text{counter} \leftarrow R_1$

$R_2 \leftarrow R_2 - 1$

$\text{counter} \leftarrow R_2$

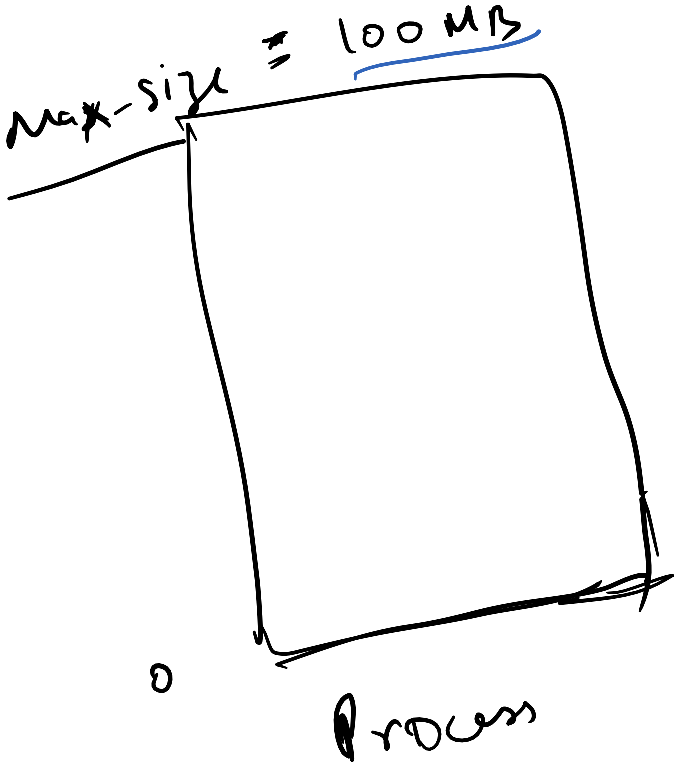
$\boxed{4}$

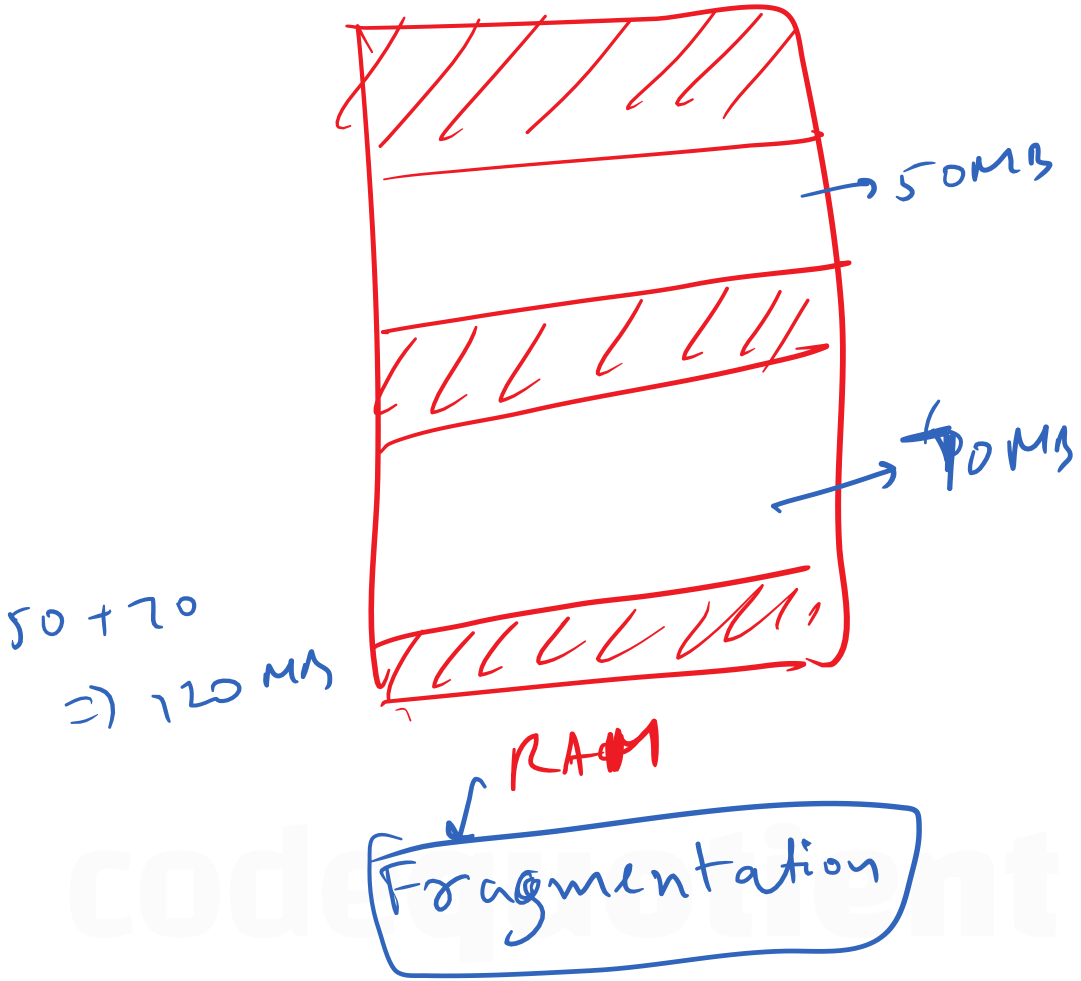
Race Condition

- A situation where several processes access and manipulate the same data (critical section)
- The outcome depends on the order in which the access takes place

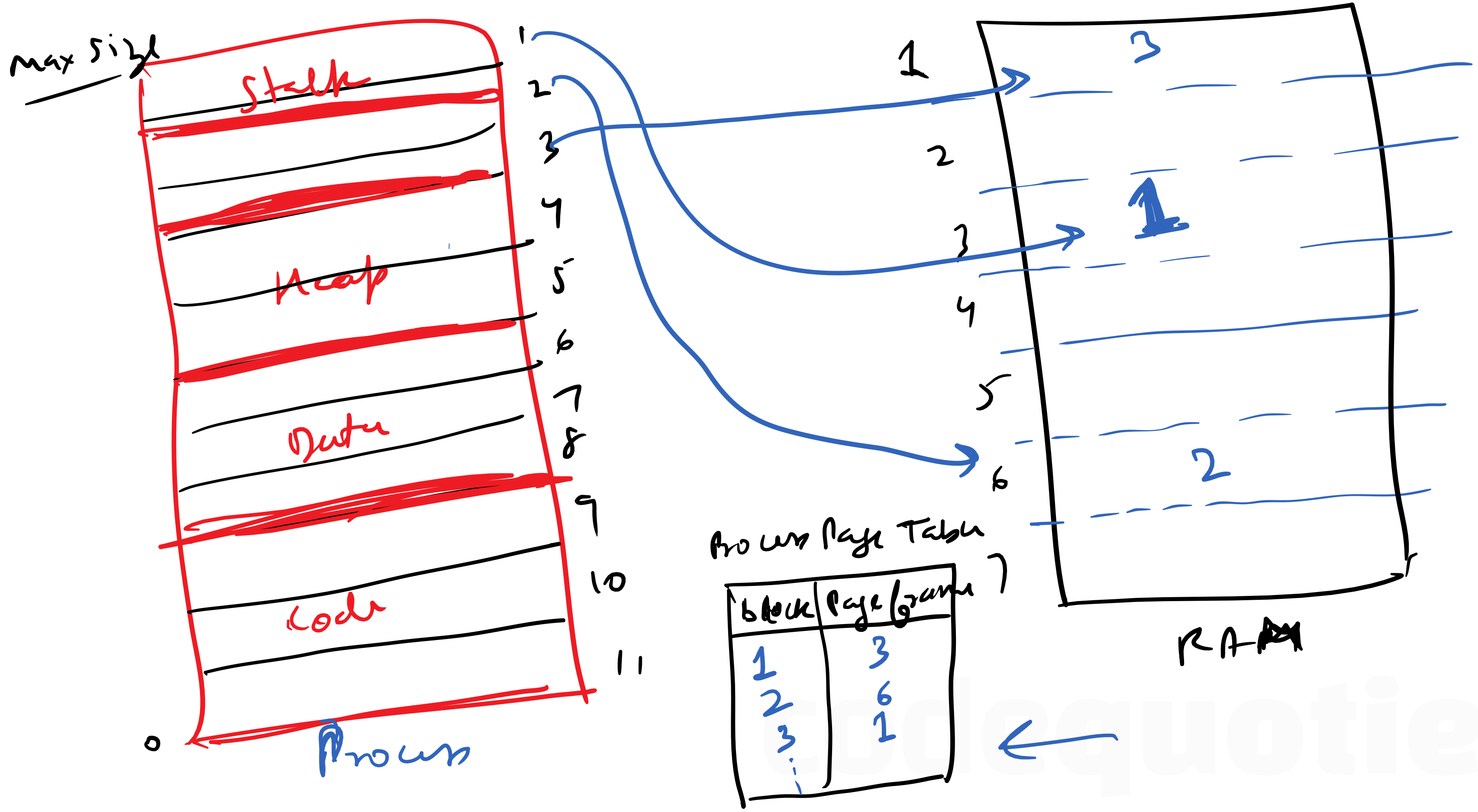
- Prevent race conditions by synchronization using locks

Virtual Memory (Paging)





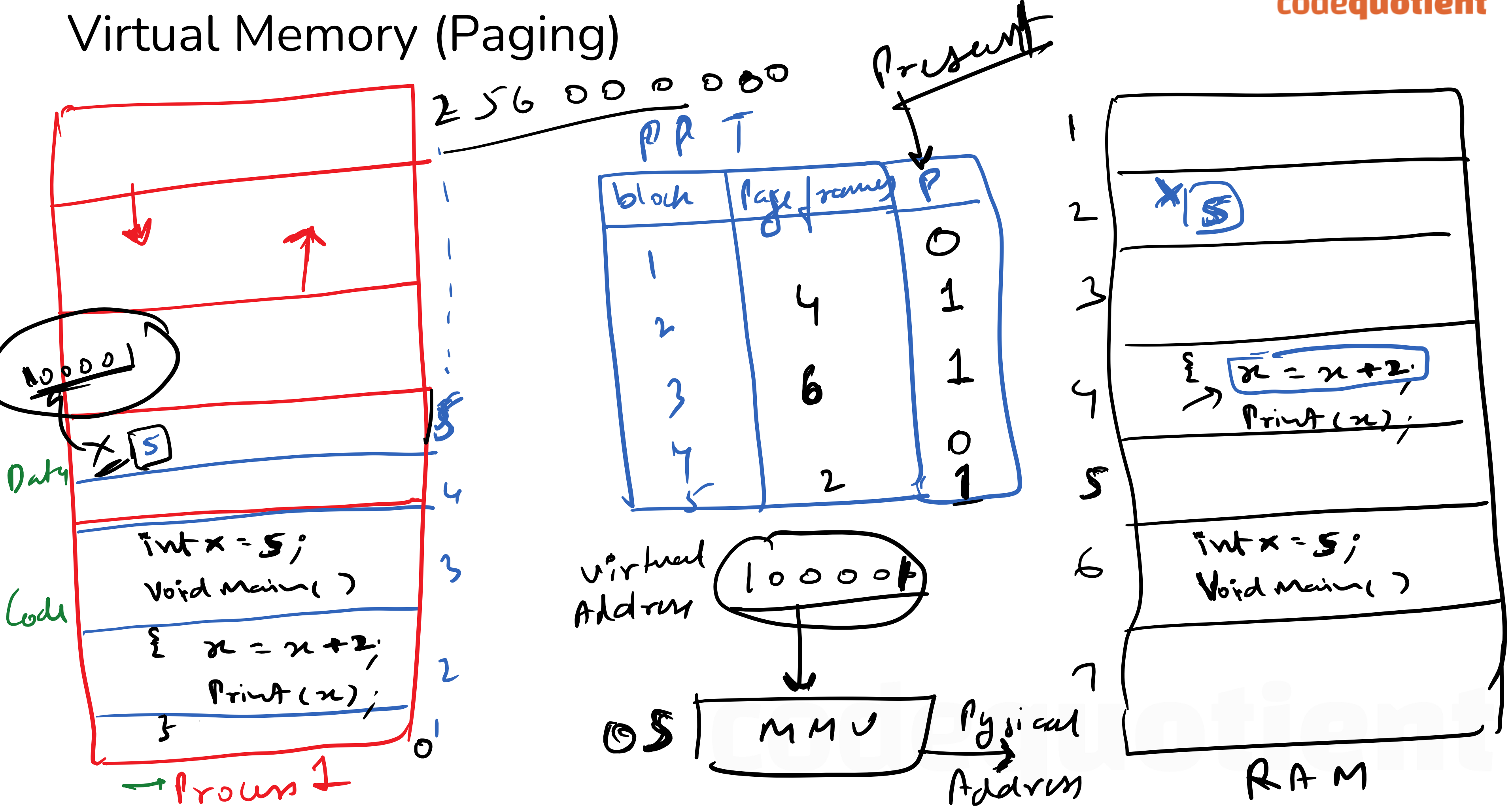
Virtual Memory (Paging)



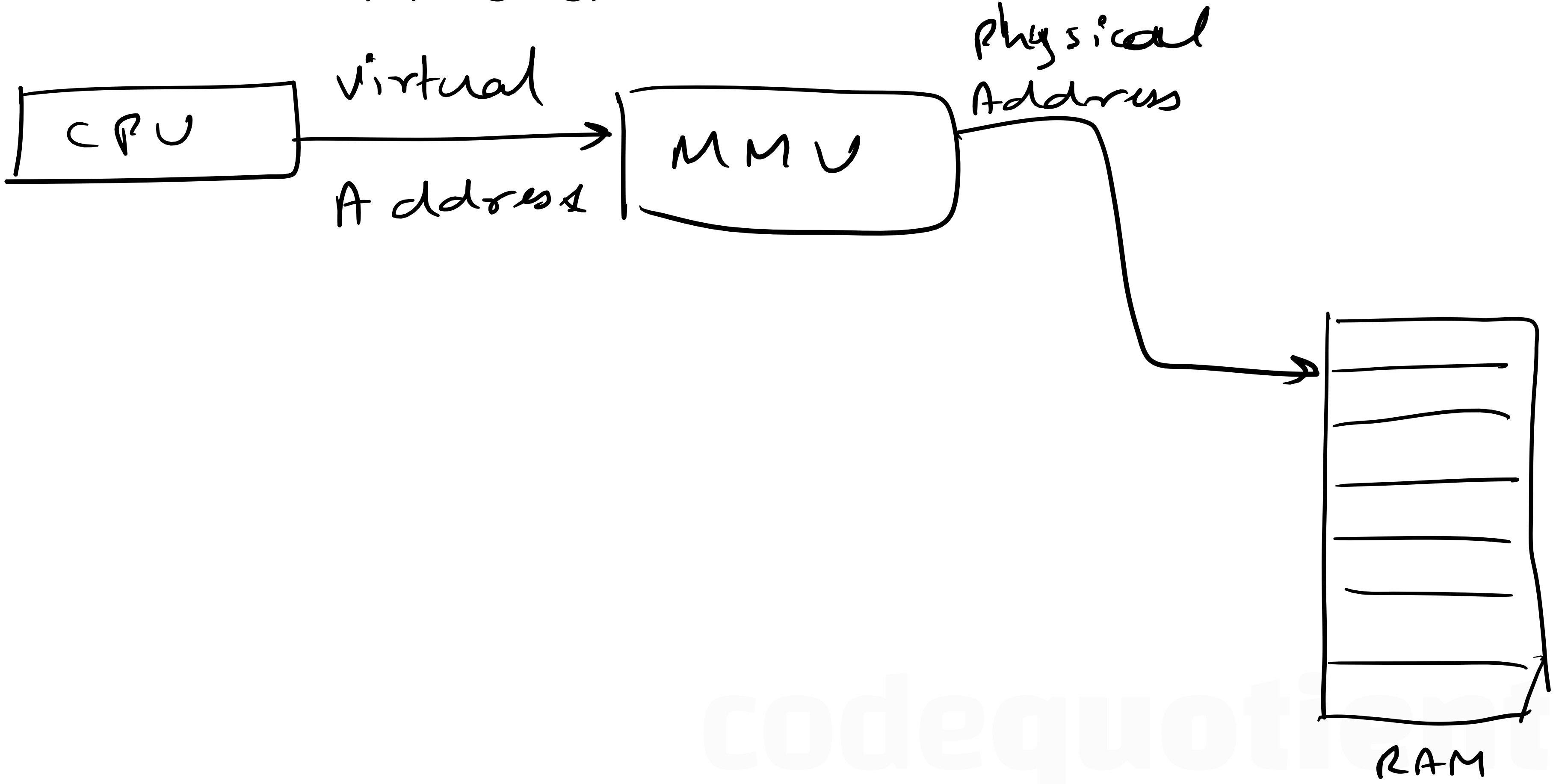
Process Page Table

block	Page frame
1	3
2	6
3	1
⋮	

Virtual Memory (Paging)



Virtual Memory (Paging)


CPU
Virtual Address
MMU
Physical Address
RAM