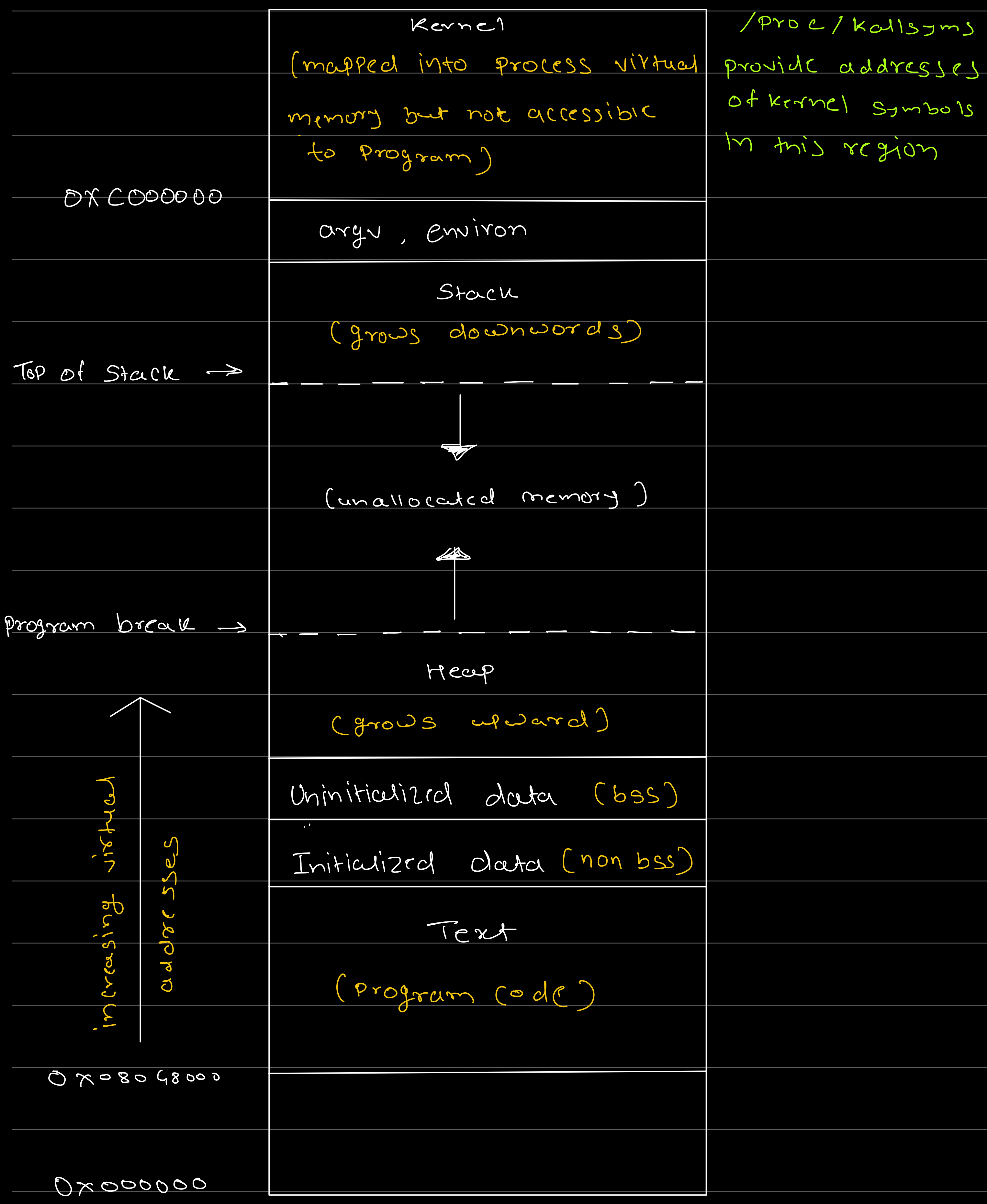
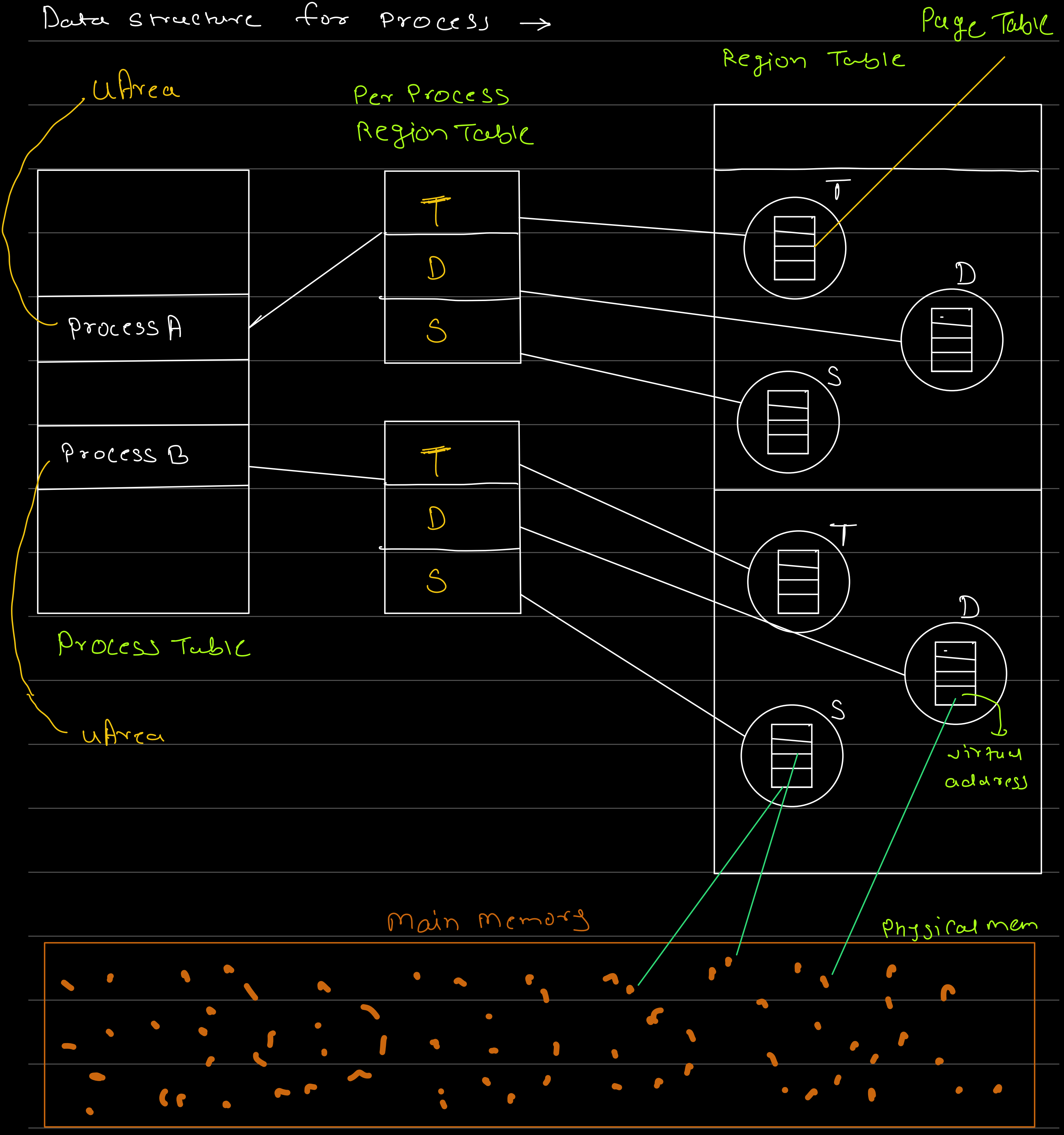
Process layout →



Kernel
(mapped into process virtual memory but not accessible to Program)
/Proc/kallsyms provide addresses of kernel symbols in this region
0XC000000
argv , environ
Stack
(grows downwords)
Top of Stack →
(unallocated memory)
Program break →
Heap
(grows upward)
Uninitialized data (bss)
Initialized data (non bss)
Text
(Program code)
increasing virtual addresses
0x08048000
0x000000

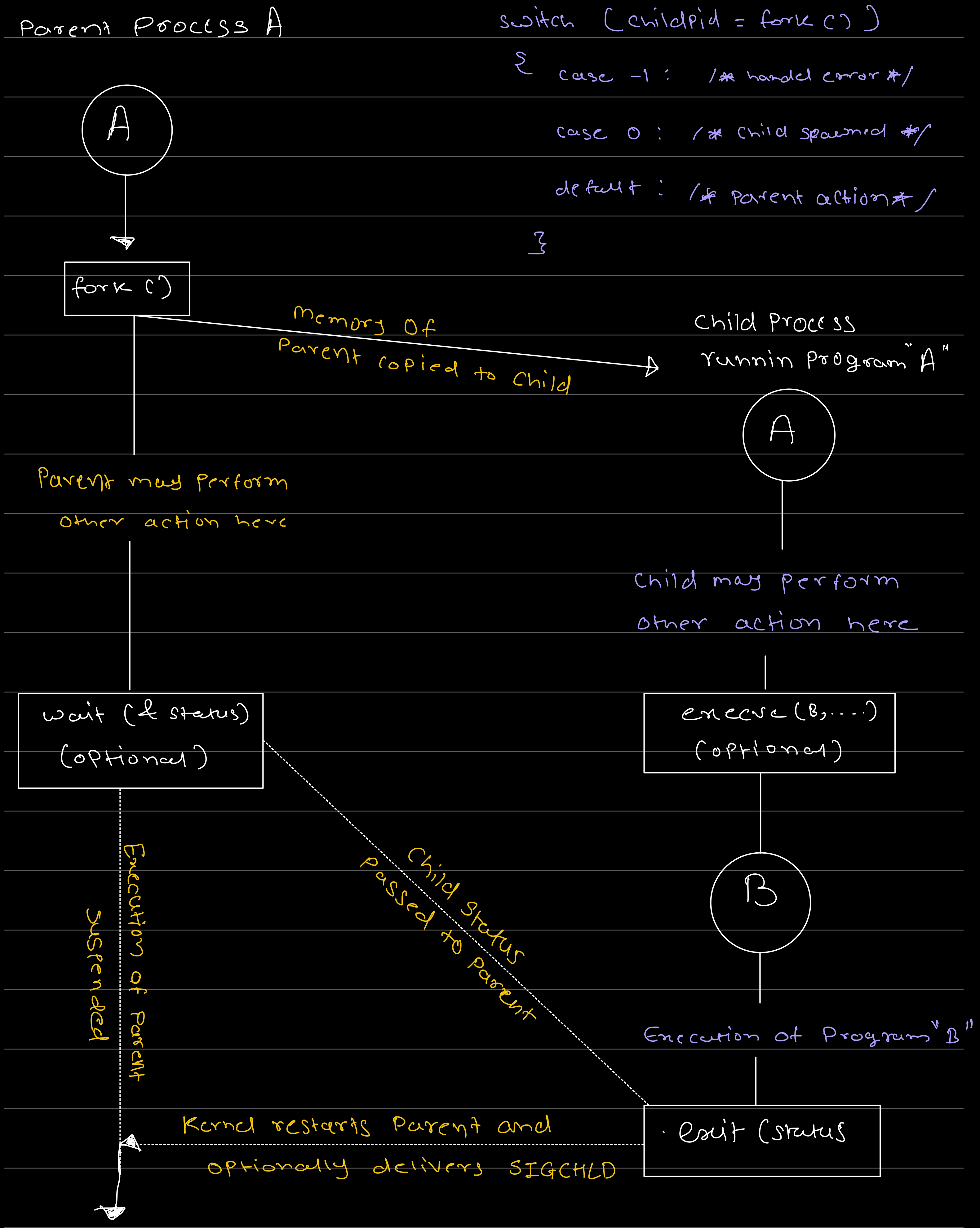
Data structure for process →

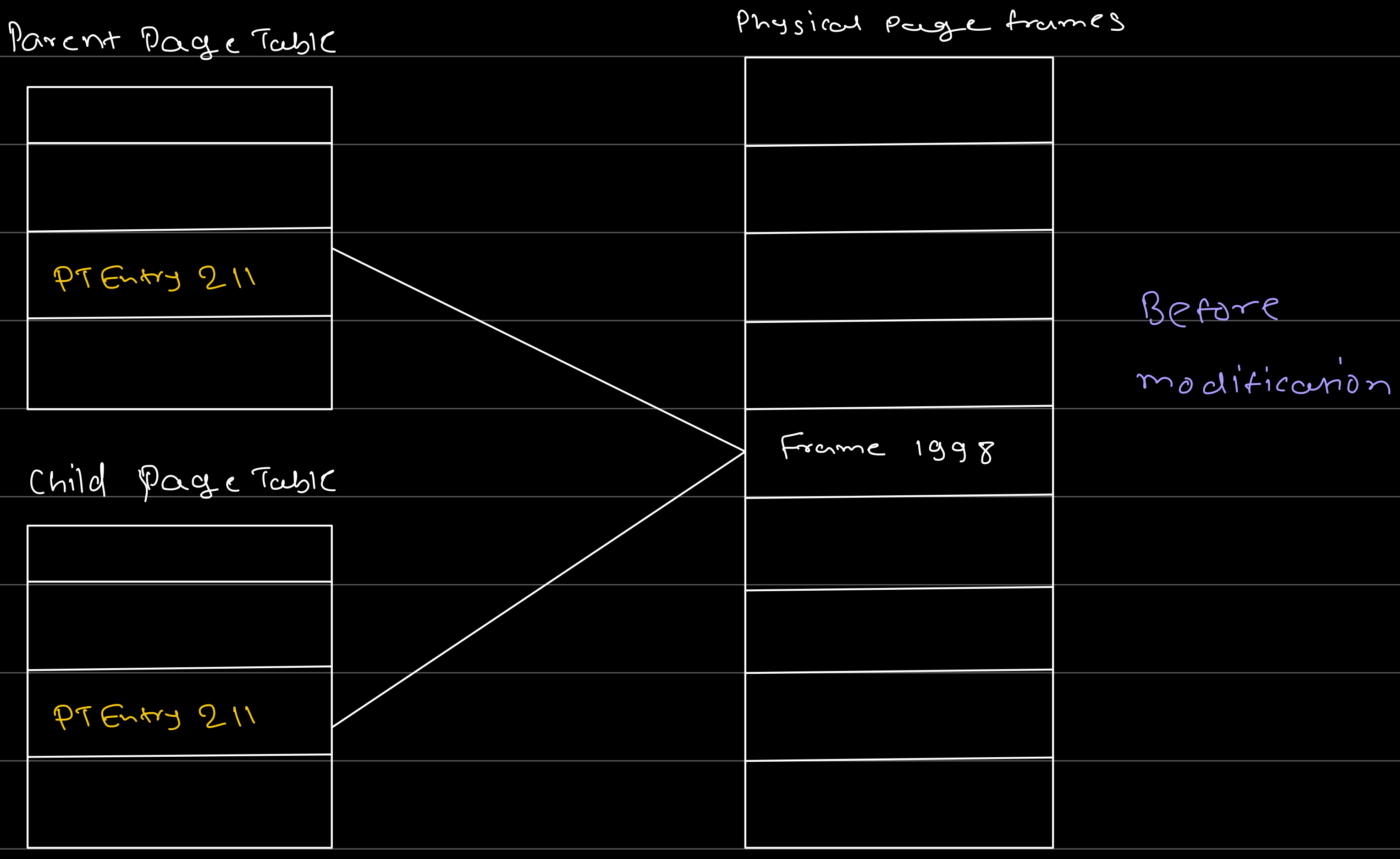


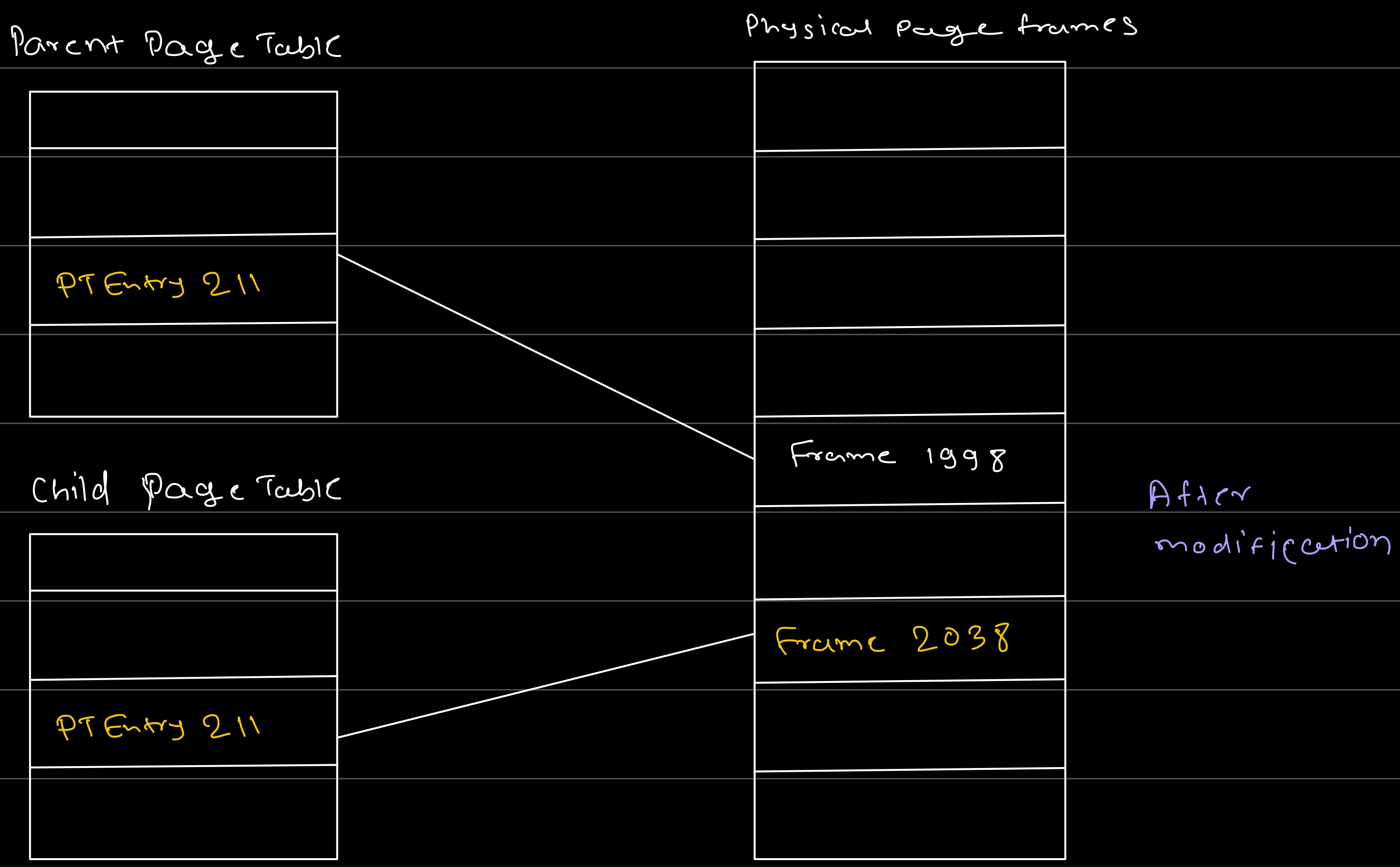
fork(), exec(), wait(), exit() flow

```
pid_t   childpid;

switch  (childpid = fork())
{  case -1 :   /* handel error */
   case 0 :   /* child spawned */
   default :   /* parent action */
}
```

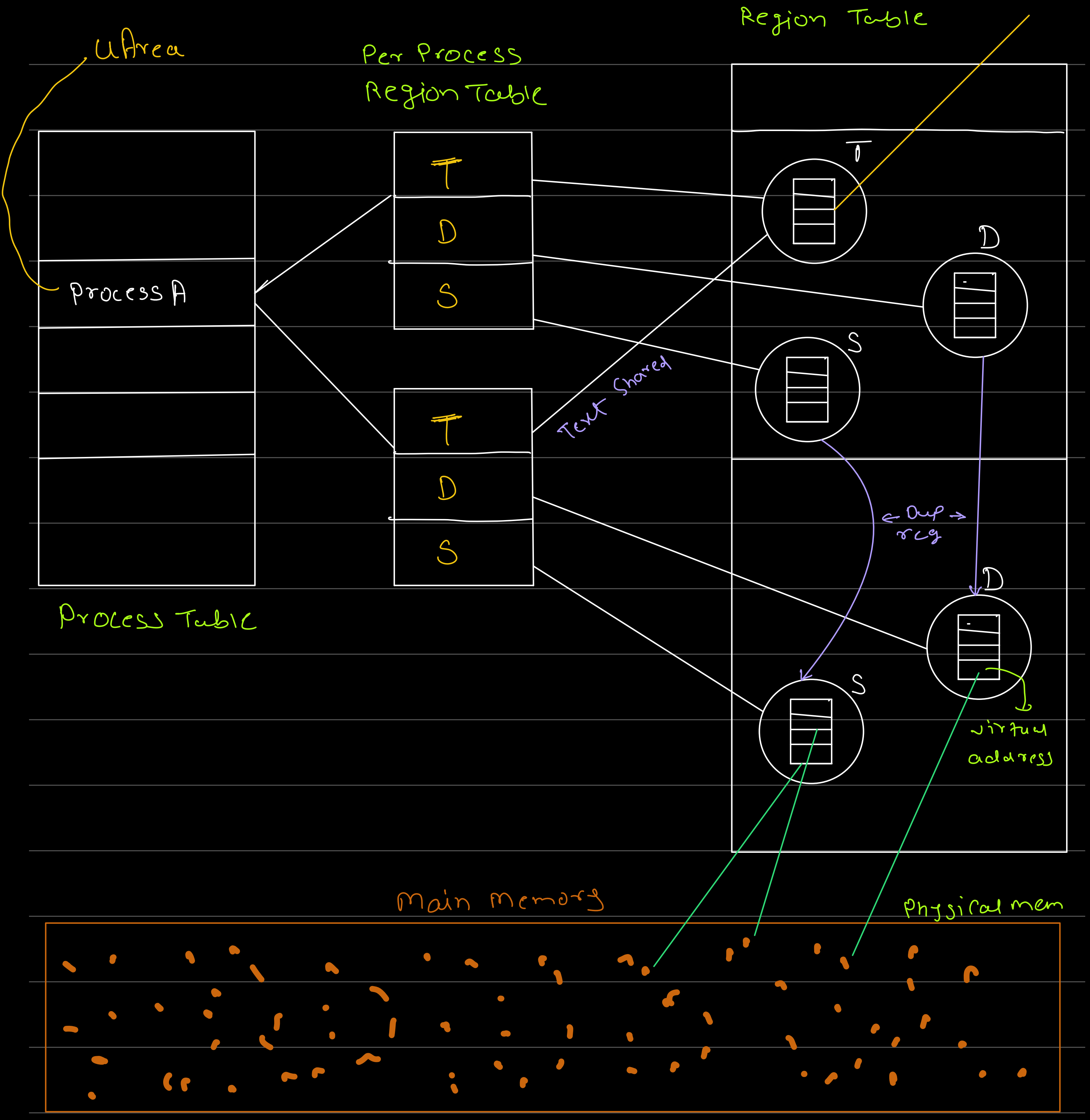

Parent Process A
A
fork()
Memory of Parent copied to Child
Child Process runnin program "A"
A
Parent may perform other action here
Child may perform other action here
wait (&status) (optional)
execve(B,...) (optional)
B
Child status passed to parent
Execution of Parent suspended
Execution of Program "B"
exit (status
Kernel restarts Parent and optionally delivers SIGCHLD



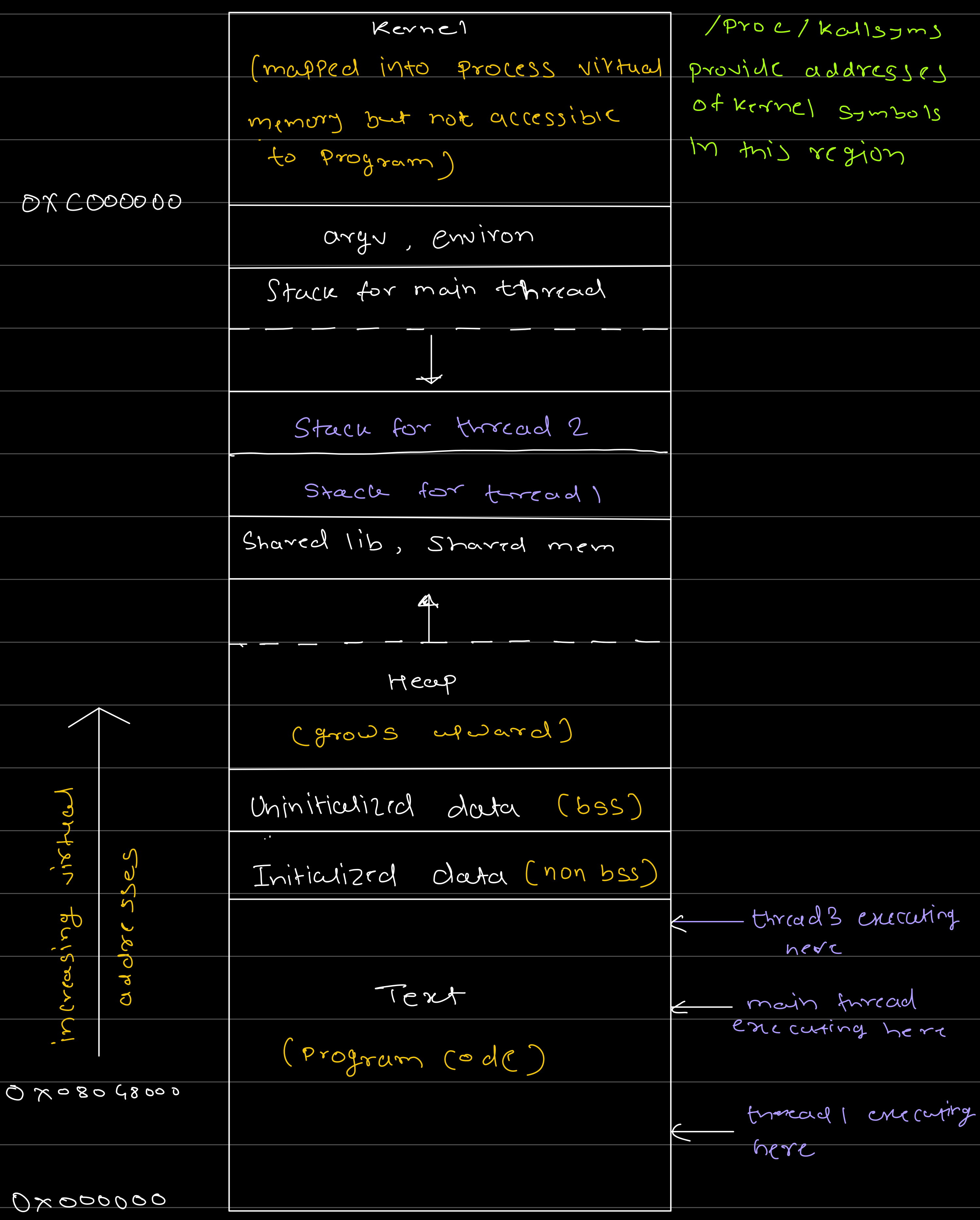


Copy on write → સમાન page પર parent & child write કરાવવાની હોય

fork() internals →



2 threads running in a Process →



Thread	process
⊖ Sharing data between threads is easy	for process need to use IPC
⊖ Thread creation is faster	process creation is heavy
⊖ Content switching is faster	content switching is slower
⊖ Thread synchronization is required	synchronisation ची गरज नाही.
⊖ एका जरी thread execution मध्ये bug आला तर सगळ्यांचे execution stop	Processes are isolated

① Process explain →① process layout

② OS ds to manage process → single process

② fork () → light weight