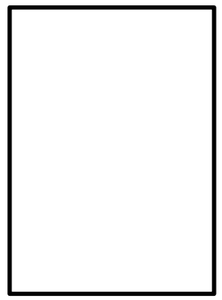
.c



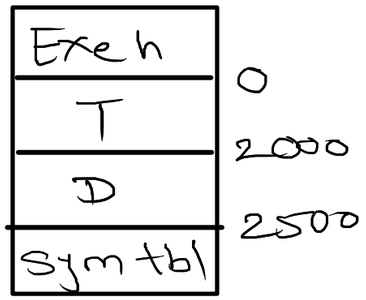
Source code

(Harddisk)

compiler

i386 + 4Gb

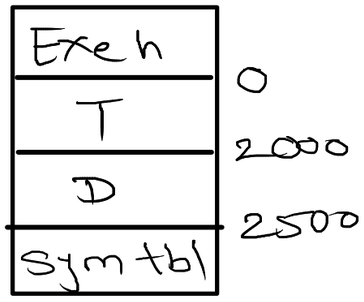
.out



Machine code

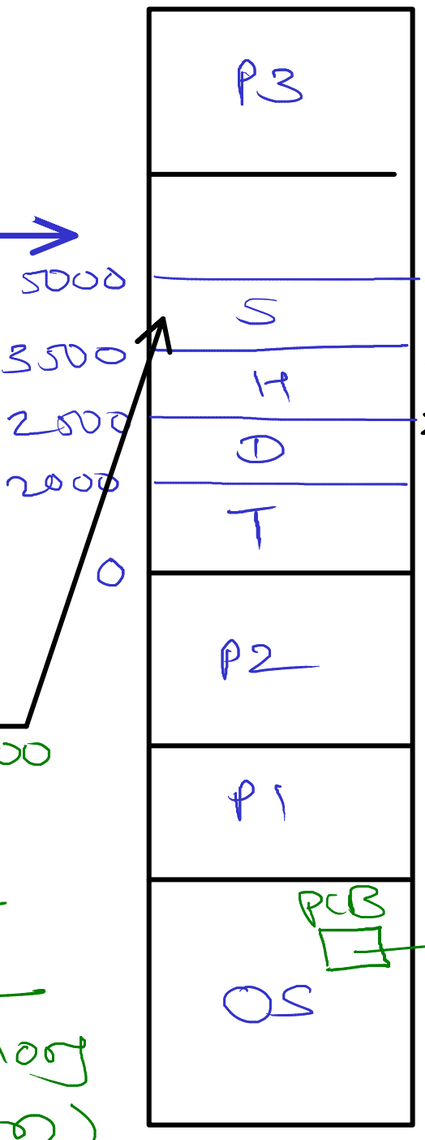
} logical/virtual addresses
logical/virtual address space
↳ set of logical addresses

.out

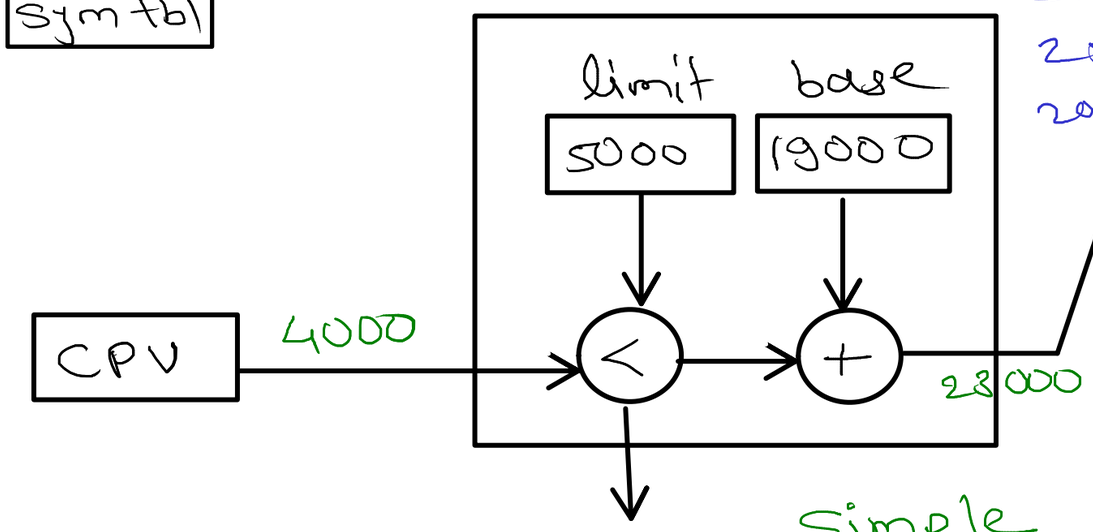


Loader

(RAM)

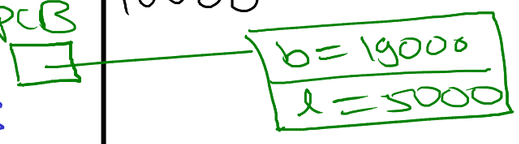


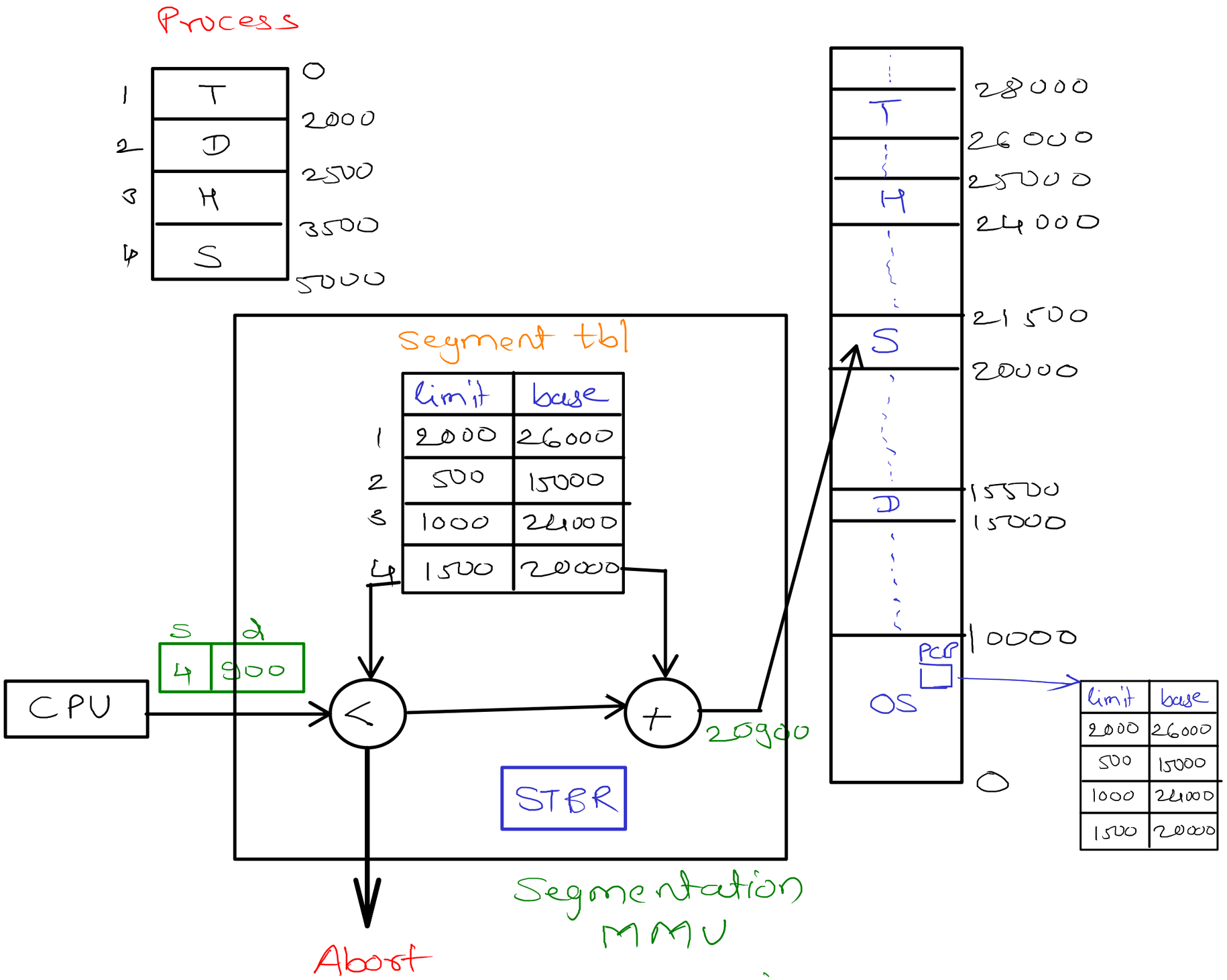
Physical addresses
Physical addr space
↳ set of physical addresses



Simple MMU

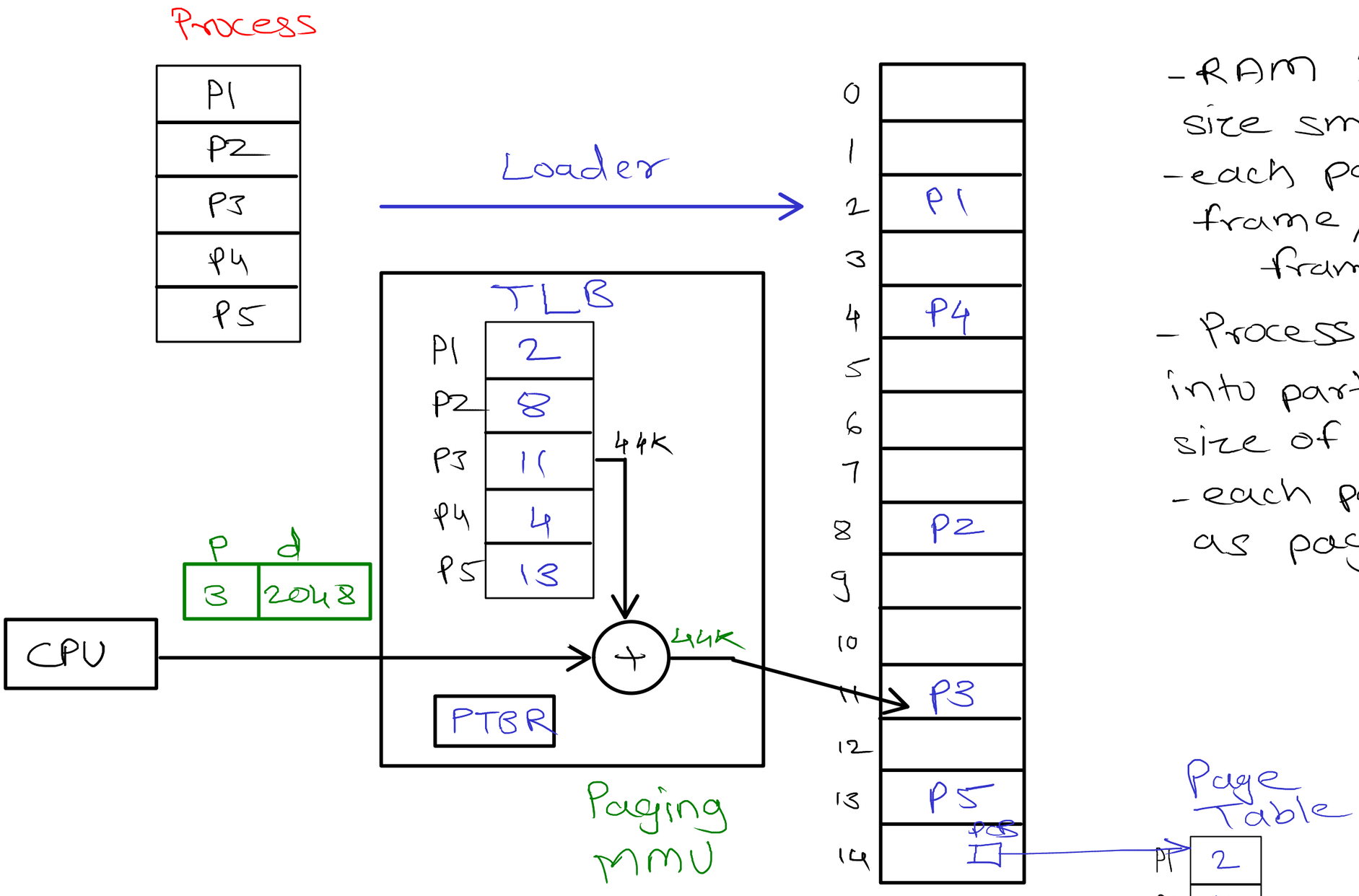
(Contiguous Memory Allocation)





Segmentation MMU

(Segment wise memory allocation)



- RAM is divided into equal size small partitions
- each partition is known frame / physical page
  frame size = 4 Kb
- Process is also divided into partitions equal to size of frame
- each partition is known as page / logical page

Swap area

- some part of harddisk is treated as RAM to store inactive processes.
- as we are treating it as RAM it is known as "virtual memory".
- we can create swap area by two methods

1. Swap file
    - file is created into OS partition and treated as swap area
    - eg. Windows (pagefile.sys)

2. Swap partition
    - seperate partition is created and treated as swap area
    - eg. Linux
    - swap area size = 2 * RAM size

- moving of process from RAM to swap area - swap out
- moving of process from swap area to RAM - swap in
- process never executes into swap area

P1
P2
P3
P4
P5

Loader

TLB

P1	2	V
P2	8	V
P3	11	V
P4	~~49~~ i	V
P5	13	V

P | d

4 | 2048

CPU

+

PTBR

Paging MMU

0	
1	
2	P1
3	
4	P4
5	
6	
7	
8	P2
9	P4
10	
11	P3
12	
13	P5
14	~~P5~~

swap out

swap in

P4

Swap area

Page Table

P1	2
P2	8
P3	11
P4	~~49~~
P5	13

- if page is not available into RAM, that page is invalid page
- if CPU request for address of invalid page then page fault is generated
- when page fault is generated into system, page fault handler of OS is called

```
page_fault_handler()
{
    1. check if address is valid or not
        if address is not valid, terminate the process
    2. check if address has read or write permissions
        if no permission then terminate the process
    3. means page is on swap area, find free frame in which we can swap in
       the page
    4. swap in the page into free frame and update same into page table
       and TLB
    5. re execute the instruction for which page fault is occured
}
```