Review Session 4. March 23. Print handout!

☑ 0. Recording and attendance.

□ 1. Background knowledgo

a. virtual memory

b. kernel

c. Process Control Block (PCB)

□ 2. Lab 4 Overview

a. important data structure

b. MACROS

c. important functions

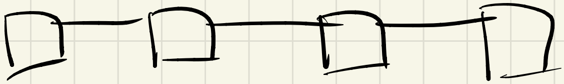
d. lab steps.

□ 3. Tips.

□ 4. Q&A

1. Background knowledge

a. Virtual memory.



b. Kernel

c. Process Control Block.

2. Lab Overview

a. data structures

pageinfo array

kernel.c

```
physical_pageinfo {
        owner      ←  PID.
        refcount   ←
}.
physical_pageinfo   pageinfo[PAGENUM
                             (MEMSIZE_
                              PHYSICAL)
```

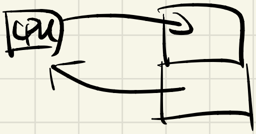
process array

kernel.h

```
proc {
      pid;
      p_registers;
      p_state.
      pagetable.
}
```

state

{ L1 pagetable. registers. }



b. Macros.

see handout.

C. Steps.

- virtual memory manage
- implement fork.

step 1:

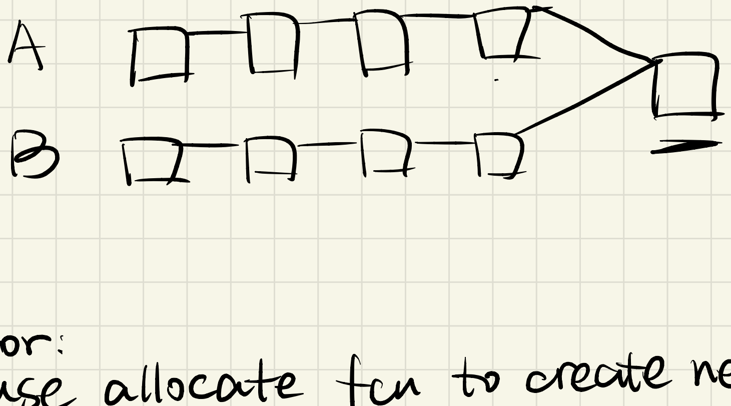
vmap.

PTE_U
PTE_P
PTE_W.

console.

stop 2:

- copy-pagetable.



Allocator:

- use allocate fcn to create new pages.
  - iterate all physical pages
  - find free ones

- assign to the process.

- allocate a new pagetable.
  Look up $PA^{\text{in } A}$ and map the new $VA_{\text{in } B}$ to PA.

INT_SYS_PAGE_ALLOC

step 3:

- Virtual memory allocation
- allocate the first free physical page

step 4:

- allow overlapping VA.

Step 5:

- fork.
  - create child process

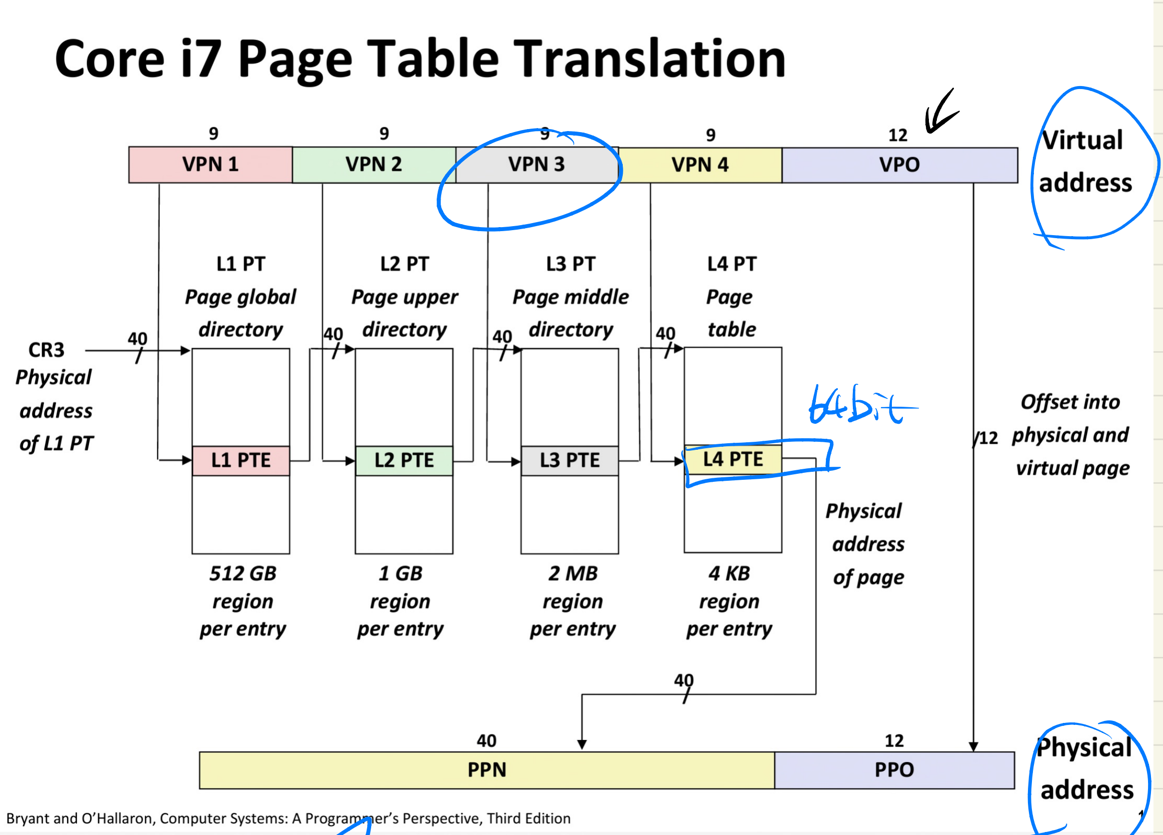
- copy memory. from parent to children
- set return value
  - set rax register.

High level.

- copy_page_table .
- copy accessible data by memcpy

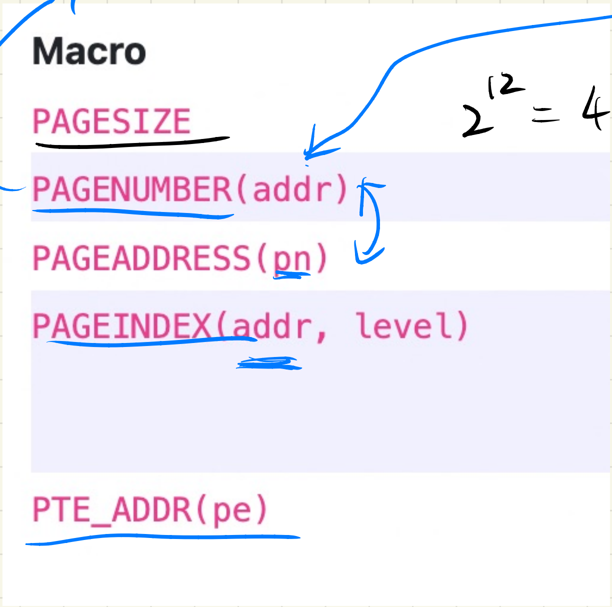
hand out

PAGEINDEX(va, 2).



Core i7 Page Table Translation

Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective, Third Edition



Macro

`PAGESIZE`

`PAGENUMBER(addr)`

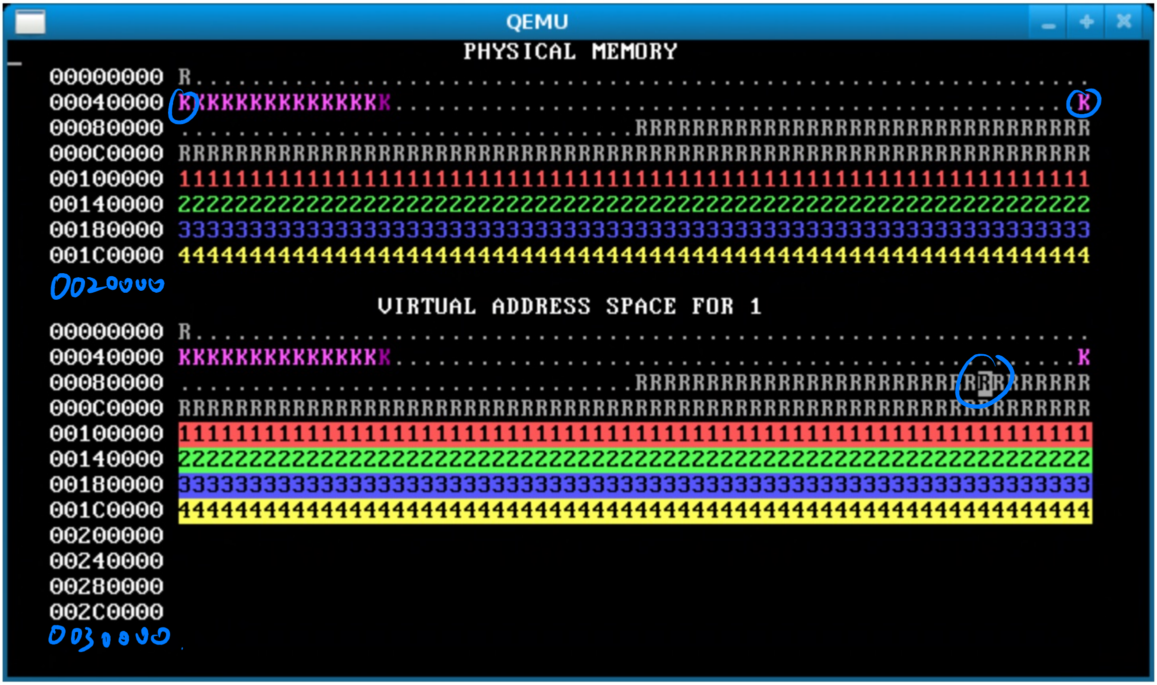
`PAGEADDRESS(pn)`

`PAGEINDEX(addr, level)`

`PTE_ADDR(pe)`

$2^{12} = 4kb.$

Constants.



```
                         PHYSICAL MEMORY
_
  00000000 R...............................................................
  00040000 KKKKKKKKKKKKKKK................................................K
  00080000 ................................RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
  000C0000 RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
  00100000 1111111111111111111111111111111111111111111111111111111111111111
  00140000 2222222222222222222222222222222222222222222222222222222222222222
  00180000 3333333333333333333333333333333333333333333333333333333333333333
  001C0000 4444444444444444444444444444444444444444444444444444444444444444
  00200000

                   VIRTUAL ADDRESS SPACE FOR 1
  00000000 R...............................................................
  00040000 KKKKKKKKKKKKKKK................................................K
  00080000 ................................RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
  000C0000 RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
  00100000 1111111111111111111111111111111111111111111111111111111111111111
  00140000 2222222222222222222222222222222222222222222222222222222222222222
  00180000 3333333333333333333333333333333333333333333333333333333333333333
  001C0000 4444444444444444444444444444444444444444444444444444444444444444
  00200000
  00240000
  00280000
  002C0000
  00300000
```

Hints:

Constant	Meaning
KERNEL_START_ADDR	Start of kernel code.
KERNEL_STACK_TOP	Top of kernel stack. The kernel stack is one page long.
console	Address of CGA console memory.
PROC_START_ADDR	Start of application code. Applications should not be able to access memory below this address, except for the single page at console.
MEMSIZE_PHYSICAL	Size of physical memory in bytes. WeensyOS does not support physical addresses ≥ this value. Defined as 0x200000 (2MB).
MEMSIZE_VIRTUAL	Size of virtual memory. WeensyOS does not support virtual addresses ≥ this value. Defined as 0x300000 (3MB).