GESTIONE MEMORIA – os161 memory

A chi serve la memoria fisica?
Una volta che si è caricato il kernel nella memoria fisica al bootstrap, la memoria fisica può servire a

- **K malloc**
- **Dumbvm** → memoria virtual base implementa in os161

Nota: entrambi allocano memoria contigua.
Memoria logica viene allocata per multipli di pagina di 4096byte in frame contigui.

- Getppages→ chiama ram_stealmem
- Ramstealmem: alloca RAM contigua partendo da firstpaddr (che è incrementato)

Allocatore è comune a:

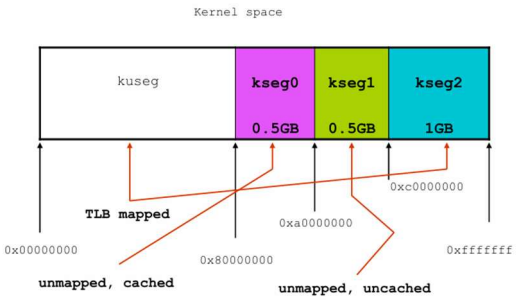
- User memory
- Dymanic kernel memory

NOTA: kernel e indirizzi user sono riconoscibile all'interno di un unico spazio di indirizzamento.
se indirizzo inizia con 0 (in esadecimale ≤7) → indirizzo logico **USER** (prima dei 2Giga)

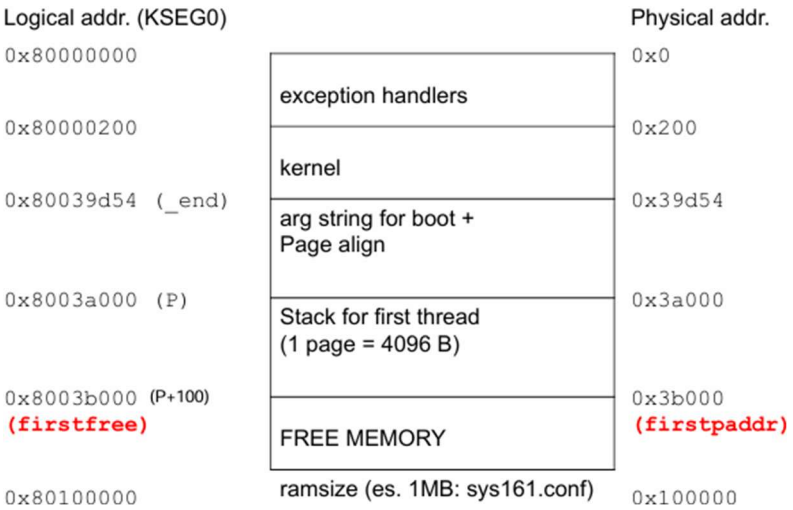
- gli indirizzi di user sono mappati in tlb quindi la MMU prova a lavorare su tlb

se inidirizzo inizia con 1 → inidirizzo di **KERNEL**: in parte mappati su tlb, in parte no

- **primo 0.5Gb**: non mappato su tlb perchè non si vuole paginazione -> vogliono memoria continua
- **secondi 0.5Gb**: oltre a non avere la tlb sono anche non cachati perchè non mi interessa velocizzare questi indirizzi perchè a questi indirizzi corrispondende un dispositivo I/O memory mapped *(è lento di suo quindi non ha senso velocizzarlo; quando la CPU scrive un dato alla scheda di rete ad un certo indirizzo, ci si aspetta di leggere quello che arriva da quella porta e non quello che ci hai scritto -> il dispositivo non deve memorizzarmi quel dato ma altro; in alcuni casi, in presenza di una cache, leggeresti solo quello che hai messo tu → la cache va bene davanti una memoria, non sempre quando è davanti ad un dispositivo I/O)*.
  Non possono avere cache perché IO.
- **ultimo segmento da 1Gb**: mappato su tlb



In OS/161, user programs live in kuseg, kernel code and data structures live in kseg0, devices are accessed through kseg1, and kseg2 is not used.



Vettore che gestisce interrupt

Argomenti per boot

Stack kernel

Nota: ad un certo momento bisognerà cambiare la *confiugurazione* di sys161.conf in modo tale da farle usare un po' più di ram

2Gb= 1, seguito da 31 zeri → 0x80000000

La memoria libera è caratterizzata da due variabili globali:

- **Firstfree** → primo libero logico *(0x8003b00)*
- **Firstpaddr** → primo libero fisico *(0x3b00)*

DUMBVM: parte ad allocare da firstfree

```
void
ram_bootstrap(void) {
  /* Get size of RAM. */
  size_t ramsize = mainbus_ramsize();
  if (ramsize > 512*1024*1024) {
    ramsize = 512*1024*1024;
  }
  lastpaddr = ramsize;  ultimo indirizzo disponibile
  /* Get first free virtual address from where
     start.S saved it. Convert to physical address. */
  firstpaddr = firstfree - MIPS_KSEG0;  primo indirizzo libero fisico
                                        indirizzo logico - 2GB
}
```

(kern/arch/mips/vm/ram.c)

```
paddr_t ram_stealmem(unsigned long npages) {
  paddr_t paddr;
  size_t size = npages * PAGE_SIZE;

  if (firstpaddr + size > lastpaddr) {
    return 0;   controllo che si sta senza sfondare
  }             RAM

  paddr = firstpaddr;
  firstpaddr += size;

  return paddr;
}
```

(kern/arch/mips/vm/dumbvm.c)

```
                chiama ram_stealmem in mutua esclusione
static paddr_t
getppages(unsigned long npages) {
  paddr_t addr;

  spinlock_acquire(&stealmem_lock);

  addr = ram_stealmem(npages);

  spinlock_release(&stealmem_lock);

  return addr;
}
```

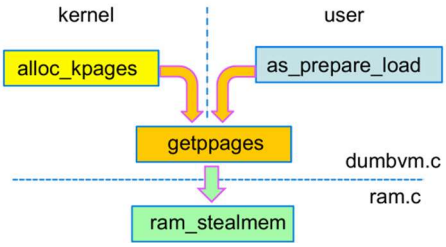
→ static in quanto è una funzione dumbvm interna

→ prende in mutua esclusione

La getppages viene chiamata da:

- as_prepare_load in user per preparzione
- in kernel con alloc_kpages.

dumbvm.c (alloc)



De-allocazione per ora non esiste, bisogna implementare:

- soluzione A1, **free in ram.c**:
  - freeppages come interfaccia a ram_freemem
  - struttura dati e gestione di memoria in ram.c
- soluzione A2, **free in ram.c**:
  - memoria non ritornata alla RAM
  - struttura dati e gestione memoria in dumbvm.c
  - freeppages si cordina con getppages
- soluzione B, **paging in user space**:
  - 2 allocatori:
    - Pagin per processi utente
    - Memoria contigua per kernel