Sistemi Operativi[1]

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2018/19

[1]

Lezione XVIII: Gestione della memoria in JOS

Iniziare con JOS

Servono almeno 512MB di ram (`-m 512` in Qemu) e `persistence-jos.qcow` (`-hda persistence-jos.qcow` in Qemu) in modo da salvare il proprio lavoro.

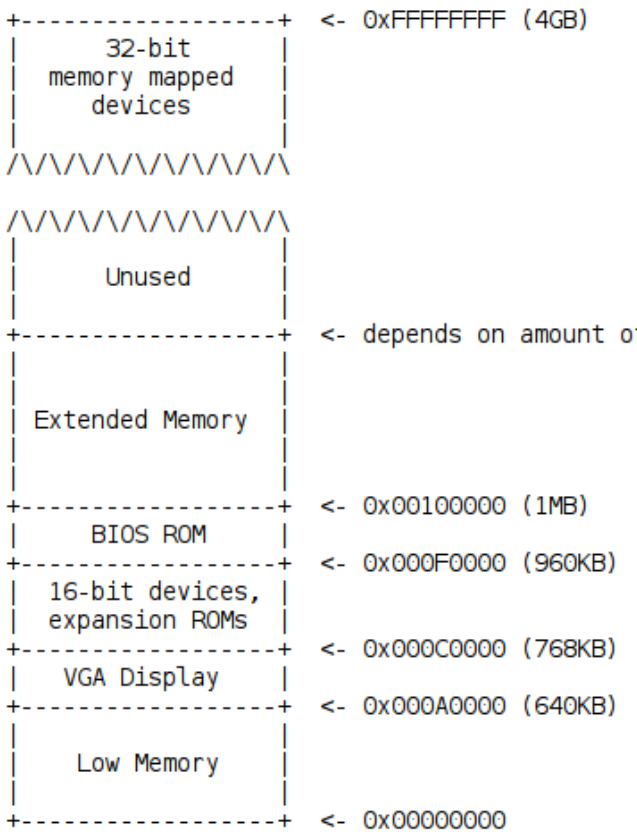
```
$ cd /home/user/joslab
$ make
$ make qemu-nox

K> kerninfo
Special kernel symbols:
  _start                  0010000c (phys)
  entry  f010000c (virt)  0010000c (phys)
  etext  f0101a6d (virt)  00101a6d (phys)
  edata  f0112300 (virt)  00112300 (phys)
  end    f0112944 (virt)  00112944 (phys)
Kernel executable memory footprint: 75KB
```

Per uscire `Ctrl-a+x`

Struttura dell'esercitazione

Seguiremo
http://pdos.csail.mit.edu/6.828/2016/labs/lab1/
(spesso semplificando per motivi di tempo: non è vietato cercare di seguire tutti gli spunti del corso MIT! Tenete conto che gli studenti MIT hanno circa 2 settimane per realizzare gli obiettivi di ogni lab)

Layout della memoria



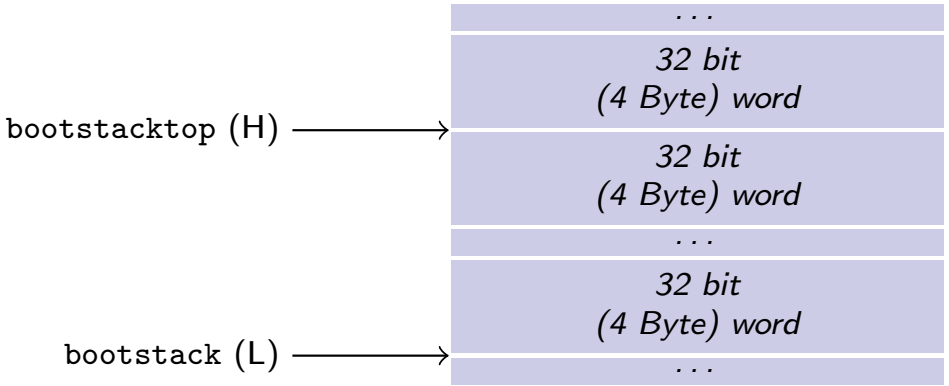
La mappa della memoria è definita dal costruttore. Generalmente accessibile via firmware o con tecniche di probing (GRUB2 fornisce un comando `lsmmap`)

Layout della memoria


```
[f000:fff0] 0xffff0:      ljmp   $0xf000,$0xe05b
```

L'indirizzo fisico è calcolato secondo il Real-Mode addressing (a 16 bit)

Lo stack



- `ESP == bootstacktop`
- `bootstacktop == bootstack + KSTKSIZE`
- Una `push` *sottrae* 4 Byte all'indirizzo ESP, una `pop` li *aggiunge*. (ESP è sempre divisibile per 4)
- Una `call` gestisce automaticamente il salvataggio dell'indirizzo di ritorno sullo stack, mentre EBP deve essere gestito a mano (salvandovi il vecchio ESP in modo da poter identificare facilmente il *record di attivazione* o *stack frame*)