



Sistemi
Operativi





Bruschi
Monga

Sistemi Operativi¹

Mattia Monga

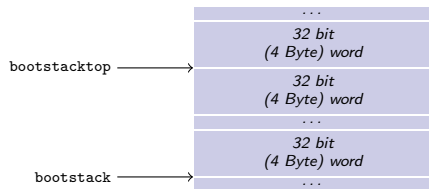
Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2012/13

¹  2011–13 M. Monga. Creative Commons Attribution-Condividi allo stesso modo 3.0 Italia License.
<http://creativecommons.org/licenses/by-sa/3.0/it/>. Immagini tratte da [?] e da Wikipedia.    1



Lezione XVIII: Gestione della memoria in JOS



- $ESP == bootstacktop$
- $bootstacktop == bootstack + KSTKSIZE$
- 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*)



Nei manuali x86 si parla di 3 tipologie di indirizzi

virtuali quando sono relativi ad un segmento: un puntatore C è un *offset*

lineare selettore di segmento + offset permette di calcolare un indirizzo nello spazio di indirizzamento (virtuale) lineare 0–4GB

fisico l'indirizzo lineare è "mappato" su un indirizzo fisico dalla MMU (che non può essere saltata!)



Segmentazione e MMU non possono essere saltati: il programmatore “vede” esclusivamente indirizzi virtuali.

- JOS configura tutti i segmenti (in `boot/boot.S` tramite la prima GDT) in `0-0xffffffff` (0-4GB), quindi il segmento può essere ignorato
- Quando serve manipolare indirizzi fisici (che **non** possono essere dereferenziati) devono essere usati *numeri* che sarà utile contrassegnare con il tipo `physaddr_t`
- Un numero che può essere dereferenziato (perché si tratta di un indirizzo virtuale) verrà contrassegnato con `uintptr_t` e per dereferenziarlo come `T` va interpretato come `T*`.



I kernel sono generalmente caricati a un indirizzo (lineare) alto, p.es. `0xf0100000` (3,75GB), che potrebbe perfino non esistere nello spazio fisico.

- il programmatore del kernel (e il programma!) usa `0xf0100000` (virtuale)
- il boot loader carica il kernel all'indirizzo `0x00100000`
- il boot loader istruisce la MMU perché mappi `0xf0100000`
→ `0x00100000`

le prime page table

- La page table 'zeresima' in boot/boot.S configura il mapping *identità*, quindi indirizzi lineari uguali a fisici.
- La prima vera page table è in kern/entrypgdir.c

lineare	fisico
0xf0000000 (KERNBASE)	0x00000000
...	...
0xf0400000	0x00400000 (4MB)
0x00000000	0x00000000
...	...
0x00400000	0x00400000 (4MB)
*	eccezione

Macro che sostituiscono la MMU



Sistemi
Operativi

Bruschi
Monga

$0xf0000000 == \text{KERNBASE} \rightarrow 0x00000000$

$0xf0100000 == \text{KERNBASE} + 1\text{MB}$

$0xf0400000 == \text{KERNBASE} + 4\text{MB} \rightarrow 0x00400000$

Alla fine del lab2 verranno mappati 256MB. Si noti che esiste una relazione semplice fra fisico e lineare: quando serve il programmatore può calcolare l'indirizzo lineare aggiungendo KERNBASE al fisico. Per farlo meglio usare KADDR (e PADDR per l'inverso) che controllano che il numero cui si applica sia sensato.