



# Sistemi Operativi<sup>1</sup>

Mattia Monga

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

a.a. 2016/17

<sup>1</sup> © 2008–17 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. <http://creativecommons.org/licenses/by-sa/4.0/deed.it>. Immagini tratte da [?] e da Wikipedia.



## Lezione VIII: Shell 2



## Astrazioni fornite dal s.o.

Per risolvere il suo problema Ada *deve* fare uso delle astrazioni fornite dal s.o. perché l'accesso diretto allo *hardware* è interdetto. Tipicamente:

- *System call*
- Memoria virtuale
- Programma in esecuzione: Processo
- Persistenza: *File*
- *Shell* (interprete comandi)

L'insieme di queste costituisce una macchina virtuale piuttosto differente dal dispositivo elettronico i386.



## Riscaldamento...

Scrivere in assembly un programma che stampa la somma di due numeri interi.

```
extern scanf, printf, exit
global main
segment .text
main:
    ; TODO

segment .rodata
imsg: db 'Inserisci due numeri interi: ',0
ifmt: db '%d',0
ofmt: db 'Somma: %d',10,0
segment .bss
x: resd 1 ; 4-byte int (dd) non inizializzato
y: resd 1 ; 4-byte int (dd) non inizializzato
```

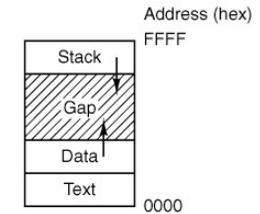


- Memoria allocata staticamente dall'inizio dell'esecuzione (generalmente nei segmenti `.data` `.rodata` `.bss`)
- Memoria allocata e liberata durante l'esecuzione secondo un protocollo imposto dallo *hardware* (`push`, `pop`  $\rightsquigarrow$  *stack*)
- Memoria allocata e liberata durante l'esecuzione tramite il sistema operativo (`malloc`, `free`  $\rightsquigarrow$  *heap*)



Il programmatore è libero di considerare un unico spazio di memoria, interamente dedicato al suo programma. Questo spazio può anche essere superiore alla memoria fisicamente disponibile.

Generalmente la memoria virtuale è divisa in *segmenti*: testo (codice), dati inizializzati, stack e heap.



- Gestito tramite `push` `pop` `call` `ESP` `EBP`
- Occorre adottare una *calling convention* nella gestione delle procedure. Una delle più diffuse è `cdecl`
  - parametri passati sullo *stack* ( $f(a, b) \rightsquigarrow \text{push } b \text{ push } a$ ), valore di ritorno in `EAX`
  - I registri `EAX` `ECX` `EDX` devono essere salvati dal *chiamante*
  - A ogni nuova `call` si alloca un corrispondente *stack frame*
    - 1 `call f` è equivalente a `push eip+len(call)` (salva sullo *stack* l'indirizzo di ritorno) e `jmp f`
    - 2 prologo: Si salva `EBP` (`push ebp`) e vi si assegna il nuovo valore `mov ebp, esp`
    - 3 Si alloca spazio sullo *stack* per le variabili locali `sub esp, ...`
    - 4 epilogo: `mov esp, ebp pop ebp`
    - 5 `ret` Equivalente a `pop eip`



```

...
mov eax, [ebp+16]
mov eax, [ebp+12]
mov eax, [ebp+8]
mov eax, [ebp+4]
mov eax, [ebp]
mov eax, [ebp-4]
mov eax, [ebp-8]
mov eax, [ebp-0xc]
...
    
```

...  
 terzo parametro della procedura  
 secondo parametro della procedura  
 primo parametro della procedura  
 indirizzo di ritorno  
 il vecchio `EBP`  
 prima variabile locale  
 seconda variabile locale  
 terza variabile locale  
 ...

Indipendentemente dalle variazioni di `ESP`!

## Esercizio



Sistemi Operativi

Bruschi Monga Re

Astrazioni

Memoria  
Stack  
Heap

Scrivere in assembly un programma che stampa il fattoriale di un numero intero positivo tramite una procedura ricorsiva.

```
fact(int n){
    int t;
    if (n == 1)
        return 1;
    else {
        t = fact(n - 1);
        return n * t;
    }
}
```

Istruzioni utili: mul sub cmp jz jmp ...

174

## Esercizio



Sistemi Operativi

Bruschi Monga Re

Astrazioni

Memoria  
Stack  
Heap

Scrivere in assembly un programma che, dopo aver chiesto all'utente il numero di interi che intende immettere, stampa tutti gli interi immessi in ordine inverso. Conservare gli interi immessi in memoria creata dinamicamente tramite malloc.

```
extern scanf, printf, exit, malloc
global main
segment .text
main:
    ; TODO

segment .rodata
msg1: db 'Inserisci il numero di elementi: ',0
msg2: db 'Inserisci i %d elementi: ',0
ifmt: db '%d',0
ofmt: db ' %d ',0
segment .bss
n: resd 1 ; 4-byte int per il numero di elementi
p: resd 1 ; 4-byte pointer per i dati immessi
```

175



Sistemi Operativi

Bruschi Monga Re

Astrazioni

Memoria  
Stack  
Heap

418