



Sistemi
Operativi

Bruschi
Monga

Concetti
generali

La macchina
fisica

Hardware

Concetti di base

Perché un s.o.

Sistemi Operativi¹

Mattia Monga

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

a.a. 2019/20



Sistemi
Operativi

Bruschi
Monga

Concetti
generali

La macchina
fisica

Hardware
Concetti di base
Perché un s.o.

Lezione II: Introduzione laboratorio



- 6 (Bruschi) + 4 (Monga) ore di lezione settimanali (9 + 3 = 12 crediti)
- Lezioni di teoria e in laboratorio
- Esame:
 - Scritto con domande a risposta multipla + orale
 - Prova pratica per la parte di laboratorio
- Libro di testo: Remzi e Andrea Arpaci-Dusseau *Operating Systems — Three easy pieces*
Versione online: <http://ostep.org>
- <http://homes.di.unimi.it/sisop/>
- Laboratorio turno Monga
<https://mameli.docenti.di.unimi.it/solab>

Sistemi
Operativi

Bruschi
Monga

Concetti
generali

La macchina
fisica

Hardware

Concetti di base

Perché un s.o.

Things A Computer Scientist Rarely Talks About



Sistemi
Operativi

Bruschi
Monga

Concetti
generali

La macchina
fisica

Hardware

Concetti di base

Perché un s.o.

*“When I talk about computer science as a possible basis for insights about God, of course I’m not thinking about God as a super-smart intellect surrounded by large clusters of ultrafast Linux workstations and great search engines. That’s the user’s point of view.”
[Donald E. Knuth]*



```
Terminal
welcome to god.creator GNU/Linux (1686, kernel 2.6.x) on tty1
Mon Jan 01 0000 00:00:01

login as: root
using keyboard-interactive authentication.
password:
Last login: Mon Jan 01 00:00:00 0000 from 100.100.100.1
you have new mail in /var/spool/mail/god
Have a lot of fun...
god:~ # cd humansoul-11b-1.4beta6
god:~ # ./configure --prefix=/usr/local --with-pain=/usr/local --includedir
god:~ # make
god:~ # make install
```



Cos'è un sistema operativo

Un insieme di programmi (*software*) che:

- 1 Facilita a programmatori e utenti finali l'uso della sottostante macchina *hardware*
- 2 Gestisce in modo ottimale le risorse di un calcolatore;

Ottiene questi obiettivi **virtualizzando** il dispositivo di calcolo: programmatori e utenti finali interagiscono con una **macchina virtuale** con caratteristiche e proprietà largamente *indipendenti* da quelle dello *hardware*.

The onion model



- Sistema operativo: l'unico programma interpretato interamente dalla macchina reale;
- Applicazioni: programmi che, sfruttando il s.o., usano le risorse virtuali per fornire valore all'utente.

Sistemi Operativi

Bruschi Monga

Concetti generali

La macchina fisica

Hardware

Concetti di base

Perché un s.o.



Sistemi
Operativi

Bruschi
Monga

Concetti
generali

La macchina
fisica

Hardware

Concetti di base

Perché un s.o.

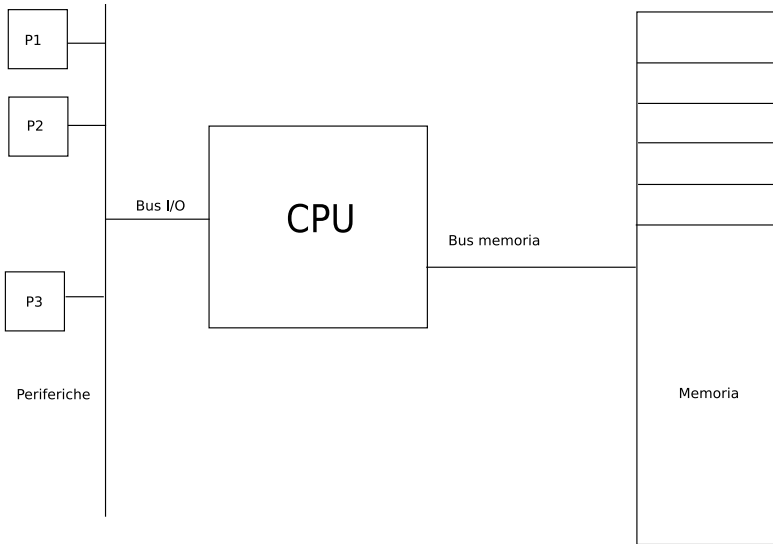
- Sistema operativo vs. *librerie*: entrambe le cose forniscono 'servizi' ai programmi, ma lo fanno in maniera *fondamentalmente* diversa. Le librerie sono a tutti gli effetti **componenti** di un programma, il sistema operativo costituisce il **contesto di esecuzione**.
- Esistono applicazioni chiamate 'macchine virtuali': java, QEmu, VMWare, VirtualBox, ... Il concetto è analogo a quello del s.o., ma **a livello applicativo** (con qualche eccezione).



- Il s.o. è l'unico programma che esegue con il totale controllo delle risorse *hardware* (**kernel mode**).
- Gli altri programmi si appoggiano unicamente sui servizi del s.o. e la loro esecuzione è gestita e controllata dal s.o. (**user mode**)
- In molti processori questa separazione è imposta via *hardware*



La macchina di Von Neumann



Sistemi Operativi

Bruschi Monga

Concetti generali

La macchina fisica

Hardware

Concetti di base

Perché un s.o.



- Registri a 32 bit
 - EAX, EBX, ECX, EDX,
 - ESI, EDI,
 - EBP, ESP,
 - EIP, EFLAGS
- Registri a 16 bit:
 - CS, DS, SS,
 - ES, FS, GS
- Real e Protected mode



- Si possono indirizzare direttamente porzioni di 8 bit, 1 byte ($AX = AH+AL$, $EAX = 16bit+AX$)
- Programmable Interrupt Controller (PIC): i8259 compatibile



I processori moderni hanno modalità di funzionamento in cui sono permesse operazioni diverse (**ring**), p.es. indirizzare tutta la memoria. i386 permette 4 ring diversi, di cui normalmente vengono usati solo 2 (Minix, p.es. ne usano 3):

- 1 kernel (supervisor) mode
- 2 user mode



Real mode, Protected mode

	Real mode	32-bit Protected mode
Protezioni hw	no	sí
Spazio di indirizzamento	2^{20}	2^{32}

- Real mode: memoria max 2^{20} byte, indirizzo ottenuto con due registri a 16 (SS:OFFSET)
*indirizzo = 16 * selettore + offset*
 - ci sono piú modi per riferirsi allo stesso indirizzo:
07C0:0000 e 0000:7C00 sono la stessa locazione fisica.
 - A20 gate
- Protected mode: il segmento è stabilito da un **descrittore** (che può essere cambiato solo in kernel mode)

Sistemi Operativi

Bruschi Monga

Concetti generali

La macchina fisica

Hardware

Concetti di base

Perché un s.o.



- NASM, <http://nasm.us>
- PC Assembly Language, by Paul A. Carter
<https://pacman128.github.io/pcasm/>
- Un altro assembler molto diffuso è gas
(<http://www.ibm.com/developerworks/linux/library/l-gas-nasm/index.html>)

```
mov    eax, 3 ; eax = 3
mov    bx, ax ; bx = ax
add    eax, 4 ; eax = eax + 4
add    al, ah ; al = al + ah
L8:db  "A"    ; *L8 = 'A'
mov    al, [L8] ; al = *L8
```



Gli assembleri x86 si distinguono per la famiglia sintattica

Intel (nasm)	AT&T (as86, gas)
<code>mov ebx, eax</code>	<code>movl %eax, %ebx</code>
<code>mov eax, 42</code>	<code>movl \$42, %eax</code>
<code>mov [ebx], eax</code>	<code>movl %eax, 0(%ebx)</code>
<code>mov [ebx+4], eax</code>	<code>movl %eax, 4(%ebx)</code>
<code>mov byte [ebx], al</code>	<code>movb %eax, 0(%ebx)</code>
<code>call eax</code>	<code>call *%eax</code>



Qemu <http://fabrice.bellard.free.fr/qemu> PC (i386 or x86_64 processor)

- i440FX host PCI bridge and PIIX3 PCI to ISA bridge
- Cirrus CLGD 5446 PCI VGA card
- PS/2 mouse and keyboard
- 2 PCI IDE interfaces with hard disk and CD-ROM support
- Floppy disk
- NE2000 PCI network adapters
- Serial ports
- PCI UHCI USB controller and a virtual USB hub.

Ogni periferica è dotata di un **controller**. Il controller avrà registri che conservano lo stato della periferica. Come accedere (leggere o scrivere) al contenuto dei registri?

- 1 Spazi di indirizzamento separati chiamati **port**. Vi si accede con istruzioni particolari:
 - `out port, eax`
 - `in eax, port`
- 2 Memory-mapped I/O, lo spazio di indirizzamento è unico
 - `mov [address], eax`
 - `mov eax, [address]`



Sistemi
Operativi

Bruschi
Monga

Concetti
generali

La macchina
fisica

Hardware

Concetti di base

Perché un s.o.

Cosa succede quando si accende un PC?



Sistemi
Operativi

Bruschi
Monga

Concetti
generali

La macchina
fisica

Hardware

Concetti di base

Perché un s.o.

Cosa succede quando si accende un PC?

- 1 Inizia l'esecuzione del firmware (BIOS)



Cosa succede quando si accende un PC?

- 1 Inizia l'esecuzione del firmware (BIOS)
- 2 Il BIOS carica il programma contenuto nel **boot sector**



Cosa succede quando si accende un PC?

- 1 Inizia l'esecuzione del firmware (BIOS)
- 2 Il BIOS carica il programma contenuto nel **boot sector**
- 3 Il programma di boot carica il sistema operativo



Cosa succede quando si accende un PC?

- 1 Inizia l'esecuzione del firmware (BIOS)
- 2 Il BIOS carica il programma contenuto nel **boot sector**
- 3 Il programma di boot carica il sistema operativo
- 4 A questo punto il controllo della macchina è affidato al s.o., a cui dovranno essere richiesti i caricamenti di altri programmi

Programming the iron



Sistemi Operativi

Bruschi Monga

Concetti generali

La macchina fisica

Hardware

Concetti di base

Perché un s.o.

What I cannot create I do not understand. [R. Feynman]

What I cannot create,
I do not understand.

Know how to solve every
problem that has been solved

Why const x sort. PG

TO LEARN:
Bethe Ansatz Probs.
Kondo
2-D Hall
local. Temp
Non linear Chemical Hydro

(A) $f = u(r, a)$
 $g = 4(r-z)u(r, z)$

(B) $f = 2|K, a|(u, a)$

Programming the iron



Sistemi
Operativi

Bruschi
Monga

Concetti
generali

La macchina
fisica

Hardware

Concetti di base

Perché un s.o.

```
2  bits 16      ; 16 bit real mode
3  org 0x7C00  ; origine indirizzo 0000:7C00
4
5  start:
6  mov ax, 0xb800      ; text video memory
7  mov ds, ax          ; ds non accessibile direttamente
8  mov bx, 10
9  write:
10  cmp bx, 0
11  jz end
12  mov byte [ds:bx], 'm' ; indirizzamento relativo a ds
13  mov byte [ds:bx+1], 0x0F ; attrib = white on black
14  sub bx, 2
15  jmp write
16  end:
17  hlt
18
19  times 510-($-$$) db 0 ; 0-padding
20  dw 0xAA55
```