



DICo

Sistemi  
Operativi

Bruschi  
Monga

Concetti  
generali

La macchina  
fisica

Hardware  
Concetti di base  
Perché un s.o.

# Sistemi Operativi<sup>1</sup>

Mattia Monga

Dip. di Informatica e Comunicazione  
Università degli Studi di Milano, Italia  
[mattia.monga@unimi.it](mailto:mattia.monga@unimi.it)

a.a. 2011/12



DICo

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 (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: *Operating System Design and Implementation*, di Tanenbaum e Woodhull, III ed.
- <http://homes.dico.unimi.it/sisop/>

# Things A Computer Scientist Rarely Talks About



DICo

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
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-lib-1.4beta6
god:~ # ./configure --prefix=/usr/local --with-pain=/usr/local --includedir
god:~ # make
god:~ # make install
```



## Cos'è un sistema operativo

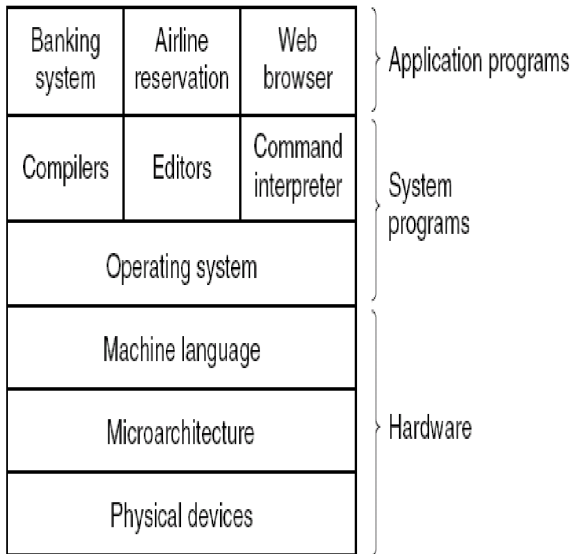
Un insieme di programmi che:

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

# The onion model



DICo



Sistemi Operativi

Bruschi Monga

Concetti generali

La macchina fisica

Hardware  
Concetti di base  
Perché un s.o.

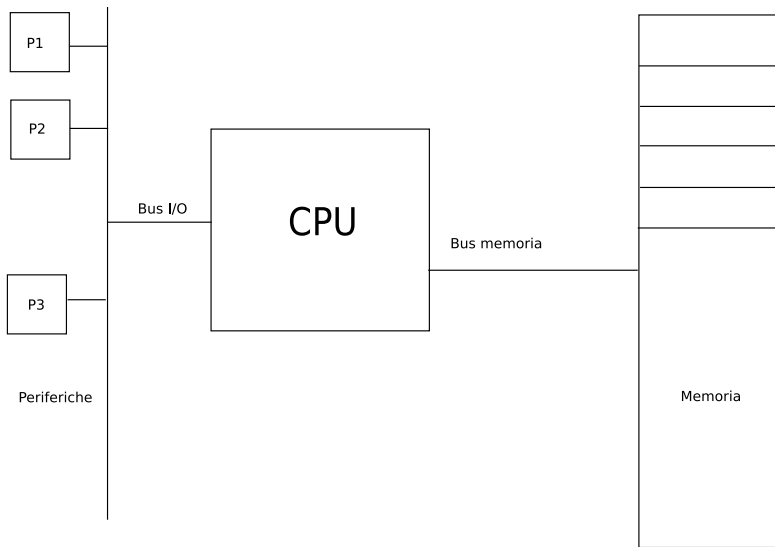


- Il s.o. è l'unico programma che esegue con il totale controllo delle risorse hardware (**kernel mode**).
- Gli altri programmi si appoggiano unicamente sui servizi del s.o. e la loro esecuzione è gestita e controllata dal s.o. (**user mode**)
- In molti processori questa separazione è imposta via hardware

# La macchina di Von Neumann



DICo



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 ne usa 3):

- 1 kernel (supervisor) mode
- 2 user mode

# Real mode, Protected mode



DICo

Sistemi Operativi

Bruschi Monga

Concetti generali

La macchina fisica

Hardware  
Concetti di base  
Perché un s.o.

	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)



- NASM, <http://nasm.sourceforge.org>
- PC Assembly Language, by Paul A. Carter  
<http://www.drpaulcarter.com/pcasm/>
- Minix usa un altro assembler (x86)  
<http://www.users.csbsju.edu/%7Ecburch/cs/portfolio/x86.pdf>

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

Gli assembleri x86 si distinguono per la famiglia sintattica

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



Qemu <http://fabrice.bellard.free.fr/qemu> PC (x86 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]





Cosa succede quando si accende un PC?



Cosa succede quando si accende un PC?

- 1 Inizia l'esecuzione del programma contenuto nel firmware (BIOS)



Cosa succede quando si accende un PC?

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



Cosa succede quando si accende un PC?

- 1 Inizia l'esecuzione del programma contenuto nel 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 programma contenuto nel 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



DICO

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  $\times$  sort .PC

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

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

(B)  $f = 2|K \cdot a|(u \cdot a)$

# Programming the iron



DICo

Sistemi Operativi

Bruschi Monga

Concetti generali

La macchina fisica

Hardware  
Concetti di base  
Perché un s.o.

```
1 ;Copyright (C) 2008, 2009 by Mattia Monga <mattia.monga@unimi.it>
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
```