Sistemi Operativi[1]

Mattia Monga

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

a.a. 2012/13

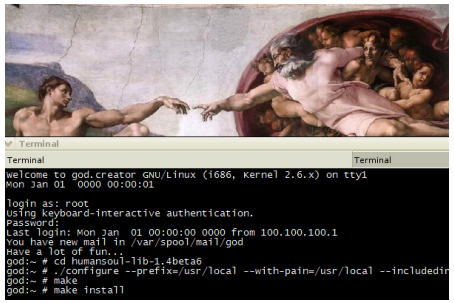
Lezione II: Introduzione laboratorio

Informazioni sul corso

- 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: *Modern Operating Systems* 3/e by Andrew S. Tanenbaum, Pearson/Prentice Hall
- http://homes.di.unimi.it/sisop/
- https://mameli.docenti.di.unimi.it/solab

Things A Computer Scientist Rarely Talks About

"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]



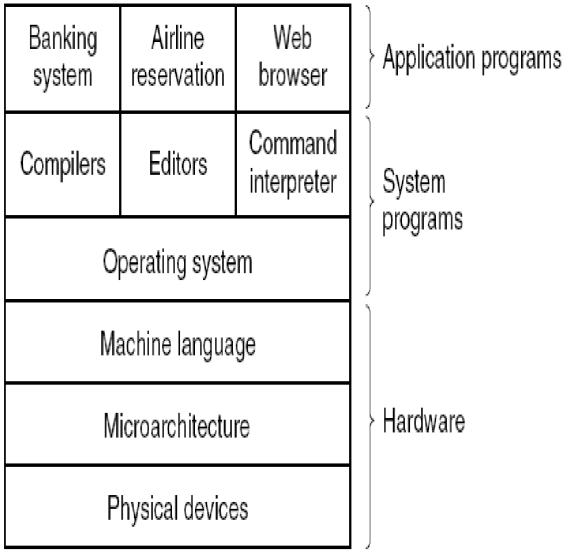
Il sistema operativo

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



Sistemi Operativi
Bruschi Monga
Concetti generali
La macchina fisica
Hardware
Concetti di base
Perché un s.o.

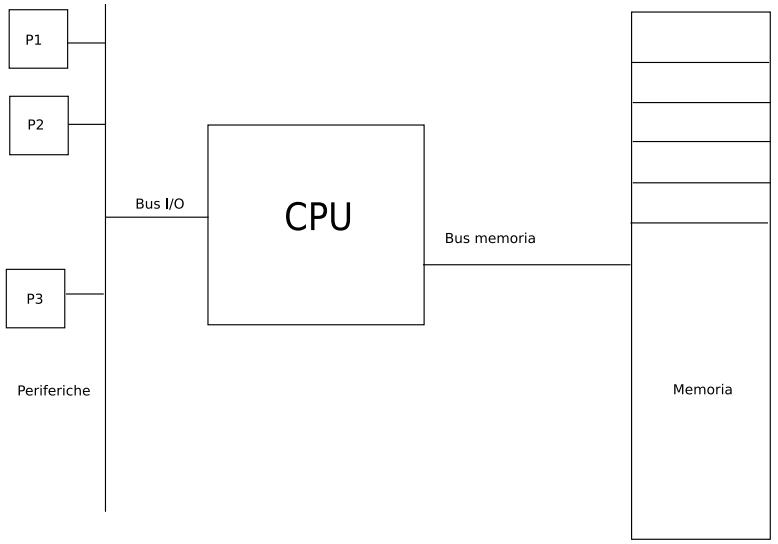
Kernel/User mode

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

Sistemi Operativi
Bruschi Monga
Concetti generali
La macchina fisica
Hardware
Concetti di base
Perché un s.o.



La macchina i386

- 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

Linguaggio macchina

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

Protezioni hardware

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

	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 $= 16 * \textit{selettore} + \textit{offset}$
  - 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)

Assembly

- NASM, http://nasm.sourceforge.org
- PC Assembly Language, by Paul A. Carter http://www.drpaulcarter.com/pcasm/
- Un altro assemblatore molto diffuso è gas (http://www.ibm.com/developerworks/linux/library/l-gas-nasm/index.html)

```
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
```


Sintassi

Gli assemblatori x86 si distinguono per la famiglia sintattica

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

Hardware virtuale

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.

I/O

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]

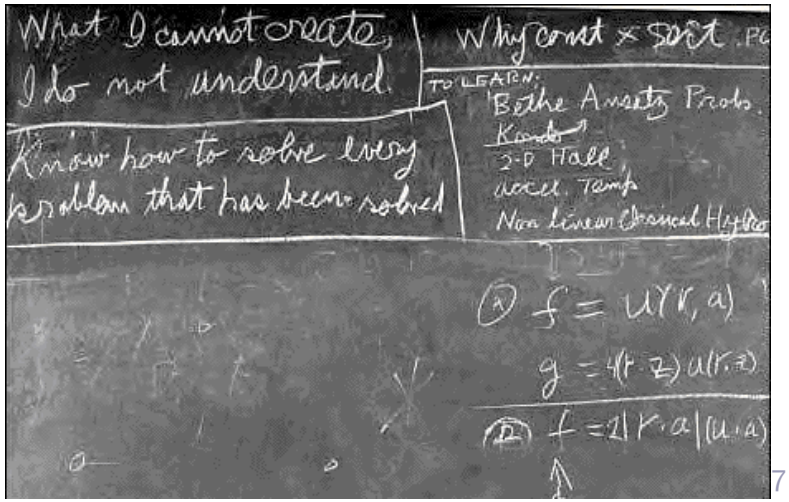
Sequenza di boot

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

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



Programming the iron


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


Programming the iron (2)


```
1
2  message:
3          lodsb ; carica un byte da [DS:SI] in AL e inc SI
4          cmp al, 0
5          jz done
6          stosw ; memorizza una word da [ES:DI] in AL e inc SI
7          jmp message
8  done: ret
9
10
11 msg db "Hello world from the bare machine!!!", 0
12
13         times 510-($-$$) db 0
14         dw 0xAA55
```

Programming the iron (3)

```
1  message:
2          lodsb ; carica un byte da [DS:SI] in AL e inc SI
3          cmp al, 0
4          jz done
5          mov ah, 0x0E ; write char to screen in text mode
6          mov bx, 0 ; BH page number BL foreground color
7          int 0x10 ; write AL to screen (BIOS)
8          jmp message
9  done: ret
10
11 boot: db "Loading unuseful system...." , 10, 13, 0
12 work: db "I've done my unuseful stuff!" , 10, 13, 0
13 cont: db "Hit ENTER to continue...", 10, 13, 0
14 wow: db "Great! Hello world!" , 10, 13, 0
15
16 waitenter: mov si, cont
17              call message
18              mov ah, 0
19              int 0x16 ; Wait for keypress (BIOS)
20              cmp al, 'm'
21              jz egg
22              cmp al, 'b'
23              jz basic
24              cmp al, 13
25              jnz waitenter
26              ret
27 egg: mov si, wow
28              call message
29              jmp waitenter
30 basic: int 0x18  ; basic (BIOS)
31              hlt
32
33         times 510-($-$$) db 0
34         dw 0xAA55
```


Sistemi Operativi

Bruschi Monga

Concetti generali

La macchina fisica

Hardware

Concetti di base

Perché un s.o.