

Avviamento di un PC

Appunti sulla scrittura di programmi avviabili
da un PC

Dopo la fase di reset

Alla fine della propria inizializzazione il processore carica nella parte visibile di CS 0xf000, in quella nascosta imposta la base a 0xffff0000 e il limite a 0xffff, EIP a 0xffff0; inizialmente CR0 vale 0x60000010 (modo reale). La prima istruzione eseguita è quella all'indirizzo fisico 0xfffff0, su EPROM, a cui si trova la prima istruzione dell'inizializzazione del BIOS.

BIOS

Basic Input Output System

E' il sistema (residente su ROM) che gestisce l' inizializzazione del PC e fornisce al sistema operativo un insieme di routine per l' I/O in modo reale.

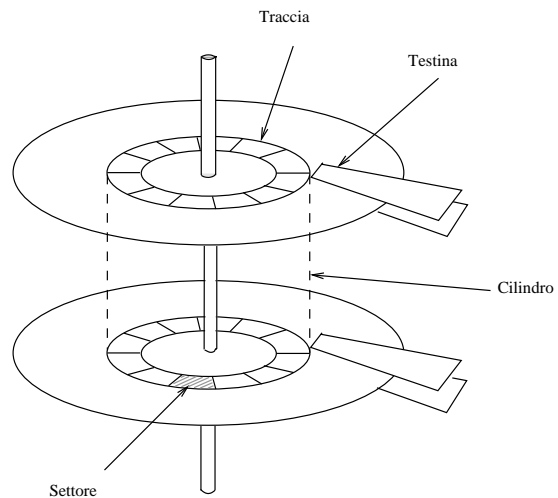
In particolare il BIOS inizializza alcune componenti hardware e lascia installata una propria IDT e dei propri gestori di interruzione che possono essere usati dal sistema operativo.

Stampa di un carattere a schermo

Con l' interrupt 0x10 è possibile stampare su schermo un carattere;
i parametri da passare sono:

- AH = 0x0e
- AL = <carattere da stampare>
- BH = <numero di pagina>

Struttura di un disco



Lettura da disco

La lettura da disco, può essere eseguita tramite l' interrupt 0x13:

- AH = 02
- AL = <settori da leggere>
- CH = <cilindro (8 bit meno significativi)>
- CL[6:7] = <cilindro (2 bit più significativi)>
- CL[0:5] = <settore>
- DH = <testina>
- DL = <drive, 0 indica il primo floppy>
- ES:BX = <buffer>

Caricamento del settore di avviamento

E' l' ultima operazione compiuta dal BIOS, al termine della propria inizializzazione: seleziona un dispositivo (di solito una unità a disco) in base alla propria configurazione e controlla se contiene un settore d' avviamento valido. Se è così lo carica all' indirizzo 0x07c0:0000 e lo mette in esecuzione.

In genere il settore di avviamento contiene un bootloader che carica il sistema operativo e lo mette in esecuzione, oppure effettua la scansione della tabella delle partizioni di un disco rigido in cerca di una partizione da avviare (caricandone il primo settore a 0x07c0:0000).

Settore di avviamento

Sono i primi 512 byte di una unità avviabile, contengono il codice (nel formato in cui deve apparire in memoria, senza intestazioni e informazioni di caricamento) che viene caricato e messo in esecuzione dal BIOS. Il settore di avviamento deve avere negli ultimi due byte i valori 0x55 e 0xaa per essere riconosciuto come valido.

Nel caso sia il primo settore di un hard-disk si chiama Master Boot Record (MBR) e contiene anche la tabella delle partizioni, all' offset 446 (decimale).

Settore di avviamento minimale

Vedi `es1.s`, codice di un settore di avviamento per floppy disk che stampa un messaggio di benvenuto tramite BIOS.

Compilazione

Il settore di avviamento viene assemblato con il comando

```
$ as es1.s -o es1.o
```

e convertito in formato binario con

```
$ objcopy -O binary es1.o es1.sect
```

Caricamento di un modulo

Il settore di avviamento in genere carica uno o più moduli esterni e li esegue. Per semplicità si considera un solo modulo da caricare, presente su floppy subito dopo il bootsector, di lunghezza nota e con indirizzo di caricamento noto.

Caricamento: esempio

Vedi `es2.s` e `mod2.s`, bootloader e modulo caricato da disco.

Compilazione dell' esempio

Il bootsector viene compilato come in precedenza, il modulo aggiuntivo può essere compilato nello stesso modo, dato che viene eseguito in modo reale (gli indirizzi sono relativi all' inizio del segmento).

Abilitazione linea A20

Per questioni di compatibilità con i PC precedenti al 286 all' avvio la ventunesima linea degli indirizzi è disabilitata. Infatti prima si assumeva che l' indirizzo 0x0000:0x0000 riferisse la stessa locazione che 0xffff:0x0010 (1M), dato che la linea A20 non esisteva. Per abilitarla e poter usare senza sovrapposizioni gli indirizzi superiori al megabyte si utilizza una serie di comandi per il controllore della tastiera.

L' esempio `es3.s` mostra il codice necessario per l' abilitazione.

Abilitazione del modo protetto

L'abilitazione del modo protetto avviene in genere subito prima di passare il controllo dell'esecuzione al modulo caricato (che può essere il sistema operativo o un programma che lo carica), in modo da permettergli di sfruttare le potenzialità che esso offre.

La procedura da seguire per l'abilitazione è descritta nel manuale Intel.

Abilitazione del modo protetto (segue)

I passi da compiere sono:

- disabilitare le interruzioni
- caricare in GDTR l'indirizzo base ed il limite della GDT
- impostare ad uno il bit PE di CR0
- eseguire una istruzione JMP inter-segment
- ricaricare i registri selettore
- caricare in IDTR l'indirizzo base e il limite della IDT del modo protetto
- riabilitare le interruzioni

Abilitazione del modo protetto: esempio

Vedi esempio `es3.s`, settore di avviamento che abilita il modo protetto e mette in esecuzione il modulo ottenuto da `mod3.s` e `mod3c.c`, caricato da disco.

Compilazione dei moduli

I singoli file sorgente sono compilati regolarmente, bisogna fare attenzione al collegamento. Nell' esempio si usa:

```
$ ld mod3.o mod3c.o -nostdlib -T ldscript --oformat  
binary  
-o mod3.bin -Map mod3.map
```

- `-nostdlib` specifica di non collegare file di libreria (che quindi non sono disponibili ai moduli)
- `--oformat binary` specifica il formato dell' oggetto prodotto
- `-T ldscript` specifica lo script da usare per il collegamento
- `-Map mod3.map` richiede la produzione di una mappa dei simboli

Linker script

Sono script contenenti la descrizione dell' output prodotto dal linker. Nell' esempio si nota:

- `<simbolo> = <valore>` crea una simbolo e gli assegna un valore (nei file sorgente per accedere a questi valori si deve usare l' indirizzo del simbolo)
- `<sezione> : { <descrizione> }` definisce una sezione nel file di uscita
- il simbolo speciale `.` indica la posizione corrente
- `*(SEZIONE)` indica le sezioni `SEZIONE` di tutti i file di ingresso

Informazioni sui moduli da caricare

Il bootloader deve conoscere la dimensione e l' indirizzo di caricamento dei moduli che carica. Se queste informazioni non si conoscono prima della compilazione (e non se ne può stimare un limite superiore) devono essere scritte su disco, in posizioni note al bootloader, da dove le possa ritrovare.

Un possibile approccio è scriverle nel settore di avviamento, ad offset occupati da variabili del bootloader (in modo che queste assumano il valore voluto quando il settore è caricato in memoria).

Copiare il settore di avviamento su disco

Il file binario ottenuto (che deve essere lungo 512 byte) deve essere copiato su disco. Per copiarlo su floppy in ambiente Unix basta:

```
$ cat bootsect > /dev/fd0
```

assumendo che `/dev/fd0` sia il dispositivo che si vuole rendere avviabile.

Per creare una immagine di floppy (da 1.44M) avviabile (per esempio per usarla con un emulatore) si può procedere così:

```
$ dd if=/dev/zero bs=512 count=2879 | cat bootsect - >  
fd.img
```

Copiare moduli aggiuntivi su disco

Per copiare sia il settore di avviamento (`bootsect`) che un modulo (`module.bin`) si possono usare diversi metodi, ad esempio:

- scrivere un programma che effettui la concatenazione, verificando la lunghezza del modulo e scrivendola nel bootsector all' offset opportuno e scrivere il risultato sul disco (anche con `cat`)
- determinare la lunghezza massima del modulo e fare caricare al bootloader un numero sufficiente di settori per coprirla; se è necessario creare l' immagine del floppy si può procedere così:

```
$ dd if=/dev/zero bs=512 count=2879 | cat bootsect  
module.bin - > fd.img.tmp  
$ dd if=fd.img.tmp of=fd.img bs=512 count=2880
```

Avviamento da cd

L'immagine di un floppy ottenuta può essere usata per rendere avviabile un cd, sfruttando, se il BIOS le supporta, le estensioni ElTorito.

All'avvio, se il BIOS è configurato per partire da cd e il cd è valido, si legge il primo settore dell'immagine del floppy presente sul cd e si avvia; tutte le letture successive da floppy (tramite int 0x13) vengono fatte in realtà dall'immagine usata per l'avviamento.

Creazione di un cd avviabile

Se `fd.img` è il nome dell'immagine avviabile creata in precedenza ci si può posizionare in una directory vuota e creare l'immagine di cd nel seguente modo:

```
$ mkdir boot
$ cp <path di fd.img> boot
$ mkisofs -b boot/fd.img -c boot/boot.cat -o
immagine.iso
```

L'immagine (che non contiene altro che la directory `boot`, `boot/fd.img` ed un file `boot/boot.cat`, che non deve essere presente nella directory di partenza), può essere scritta su cd (per esempio con `cdrecord`).

Riferimenti

- Lista dei servizi del BIOS:
www.cs.cmu.edu/afs/cs/user/ralf/pub/WWW/files.html
- Sistemi che comprendono un settore di avviamento:
 - Minix
 - Linux (tutte le versioni stabili fino al 2.4) www.kernel.org
- Manuale Intel per la programmazione di sistema
www.intel.com/design/pentiumii/manuals/243192.htm
- GRUB (bootloader completo) www.gnu.org/software/grub