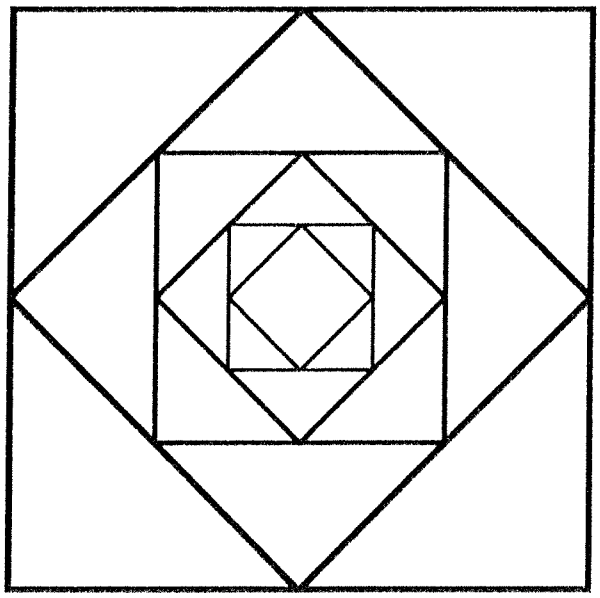
Note su
IBM Operating System/2

G. Vitillaro

Centro di Ricerca di Roma
Via Giorgione 159, 00147 ROMA

Marzo 1989



IBM Internal Use Only

Indice

ARCHITETTURA INTEL (tm)

ARCHITETTURA iAPX86 iAPX88 iAPX286-reale

General Purpose Registers
ax = ah||al - Accumulator
bx = bh||bl - Base
cx = ch||cl - Count
dx = dh||dl - Data
quattro registri a 16 bit

Segment Registers
cs - Code Segment
ds - Data Segment
es - Extra Segment
ss - Stack Segment
quattro registri a 16 bit

Other Registers
bp - Base Pointer
si - Source Index
di - Destination Index
sp - Stack Pointer
quattro registri a 16 bit

Special Registers
f - Status Flags
ip - Instructions Pointer
due registri a 16 bit

INDIRIZZAMENTO modo REALE

```
Indirizzamento                          Esempio
--------------                          -------

    segment:offset                      21F0:3500
 (physical address 32 bit)

    segment << 4 +                        21F00 +
          offset =                         3500 =

 ------------------                     --------
 physical address      (20 bit)           25400

 nota : ogni indirizzo fisico ha piu' di una rappresentazione
        ad esempio 25400 = 21F0:3500 = 2000:5400
```

- Segmento ed Offset lunghi 16 bit
- Il selettore di segmento individua un segmento fisico in memoria
- 1Mb di spazio di indirizzi 00000-FFFFF
- Possibilita' di sovrapporre i segmenti

INDIRIZZAMENTO modo REALE

Rappresentazioni di un indirizzo

$0 <= pa < 16*65536$ (16 segmenti di 64Kb)

Divisione Euclidea di pa per 16 :

$pa = q*16 + r$, con $0 <= q < 65536$, $0 <= r < 16$.

Possiamo scrivere

$pa = (q - a)*16 + (a*16 + r)$

ricordando che : $(q - a) >= 0$ e $(a*16 + r) < 65536$.

Allora posto $n = \min(q,(65536/16)-1) = \min(q,4095)$

deve essere $0 <= a <= n$.

Poniamo

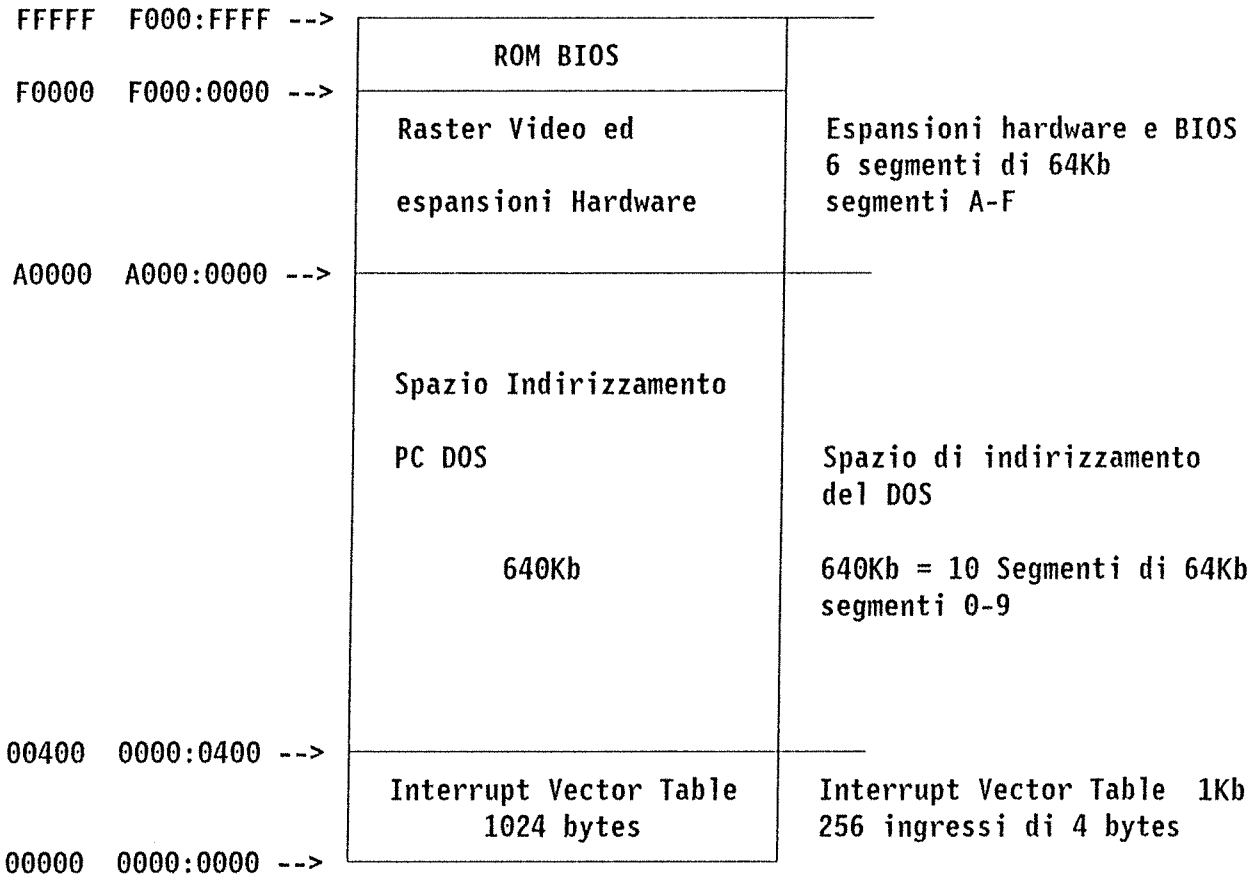
$segment = (q - a)$

$offset = (a*16 + r)$, con $a = 0,1,\ldots,n$.

Allora ogni indirizzo fisico pa ha al piu' 4096 rappresentazioni

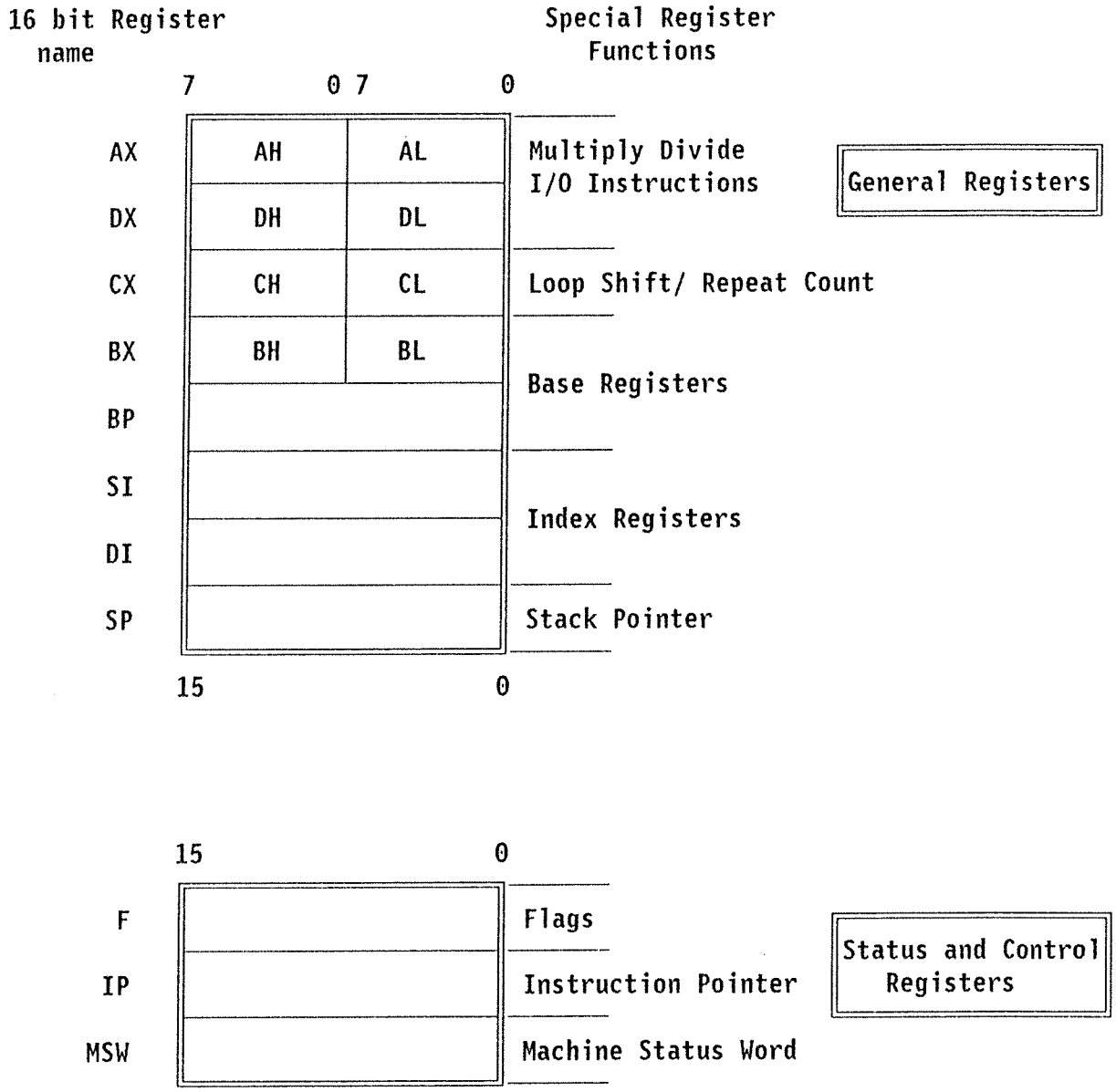
della forma : $pa = segment*16 + offset = segment:offset$.

Struttura della memoria

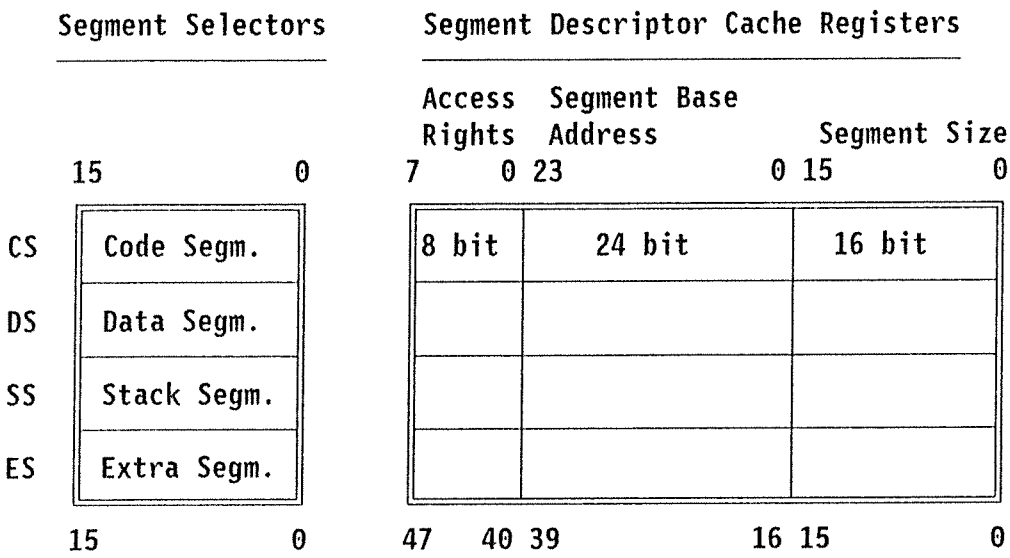


- Facilita' di indirizzamento : conoscendo un indirizzo fisico si puo' facilmente eseguire "aritmetica di segmento"
- Stretta dipendenza del DOS e della maggior parte dei programmi DOS da questa struttura

ARCHITETTURA iAPX 286-protetta

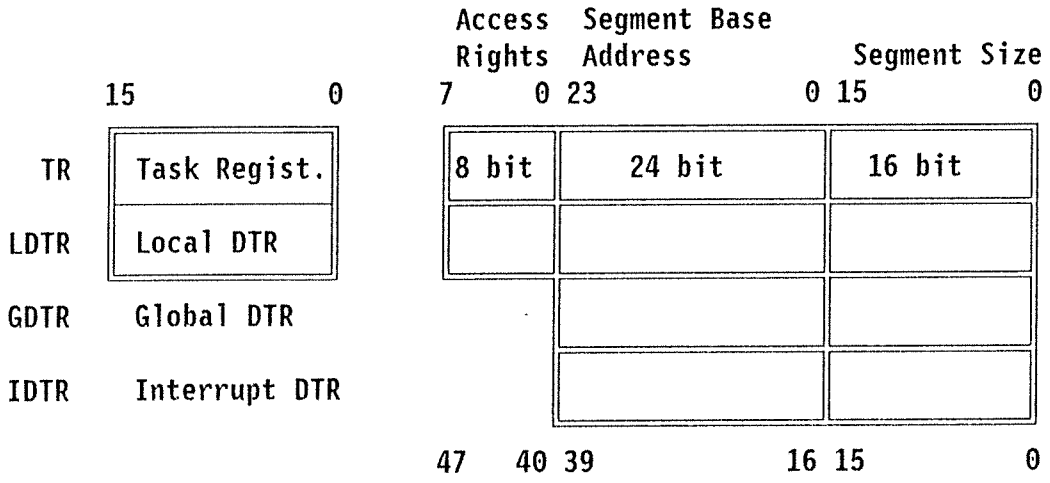


Segment Selectors and Descriptor Tables



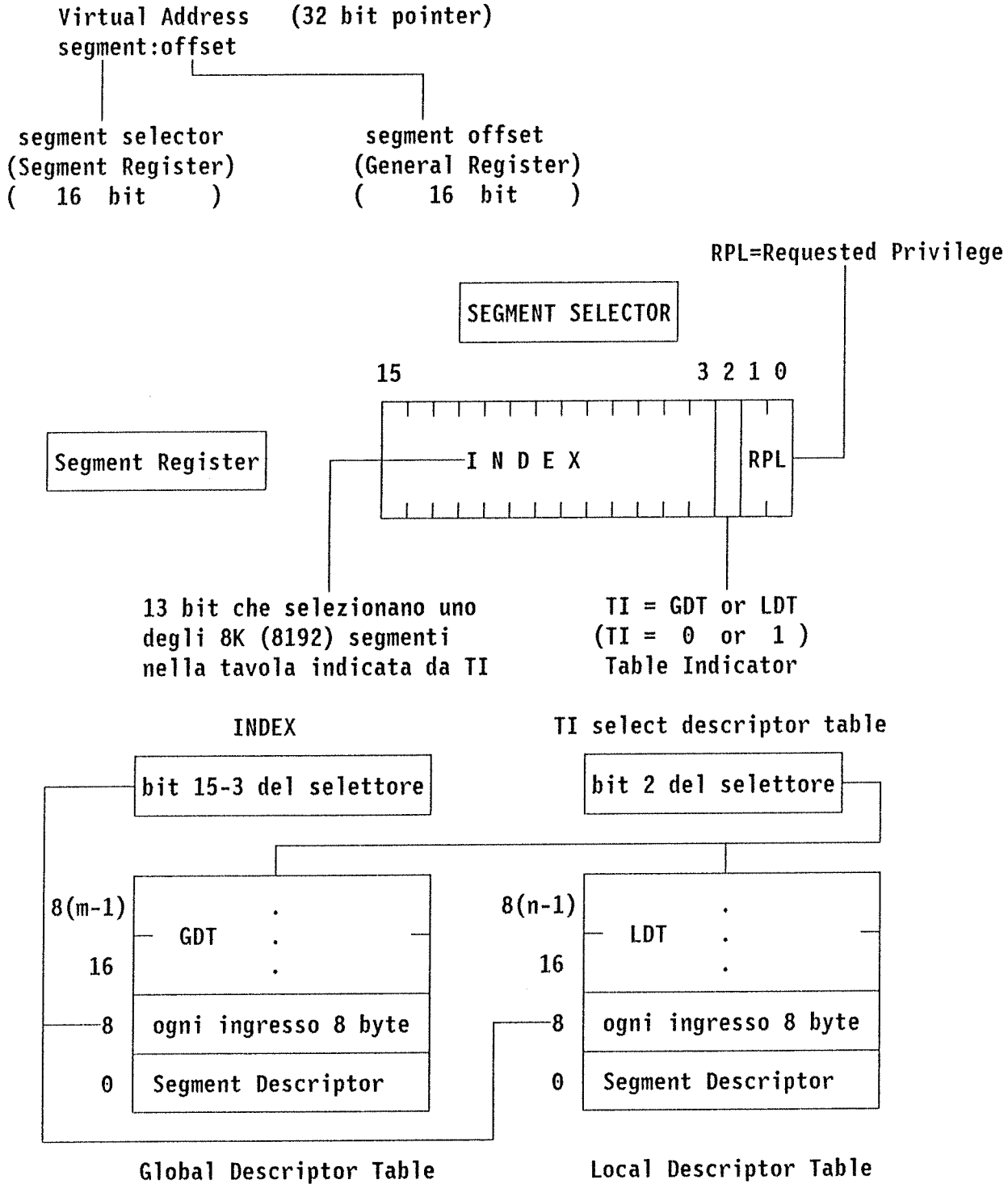
* Segment Registers loaded by program

* Segment descriptor cache registers CPU loads this explicit cache, which is invisible to programs

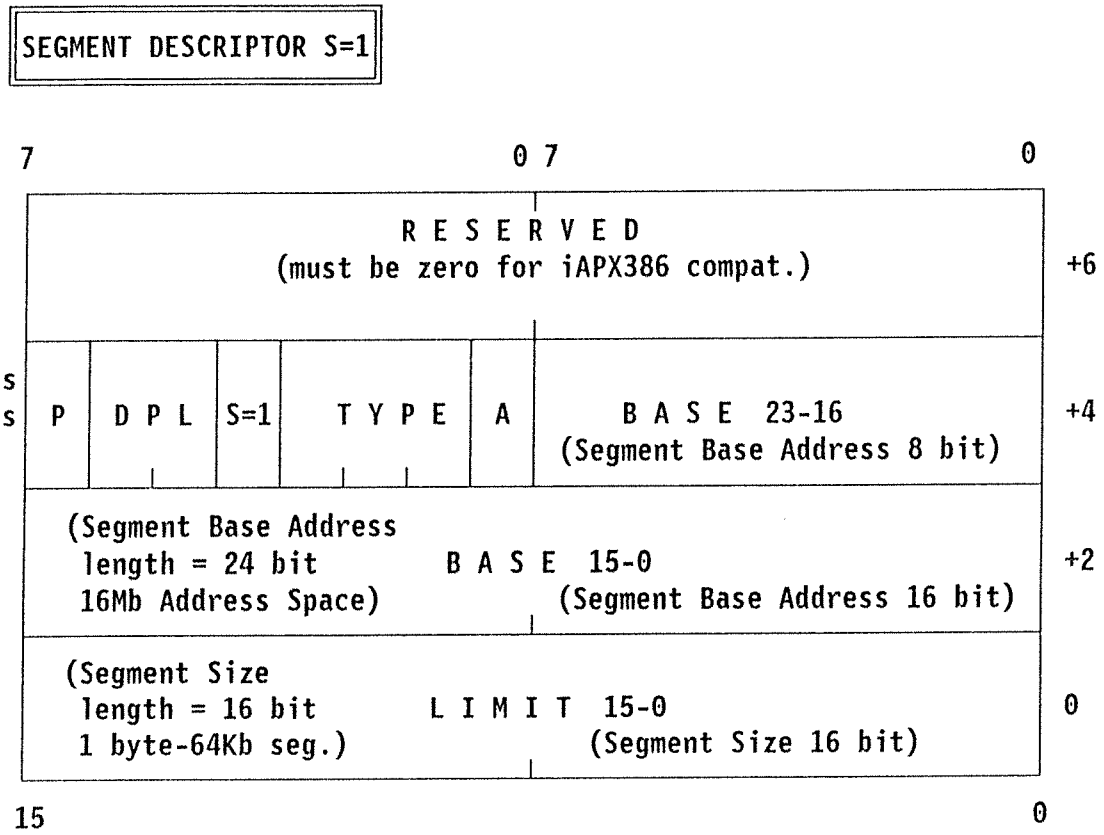


DTR = Descriptor Table Register

Indirizzi Virtuali Protected Mode



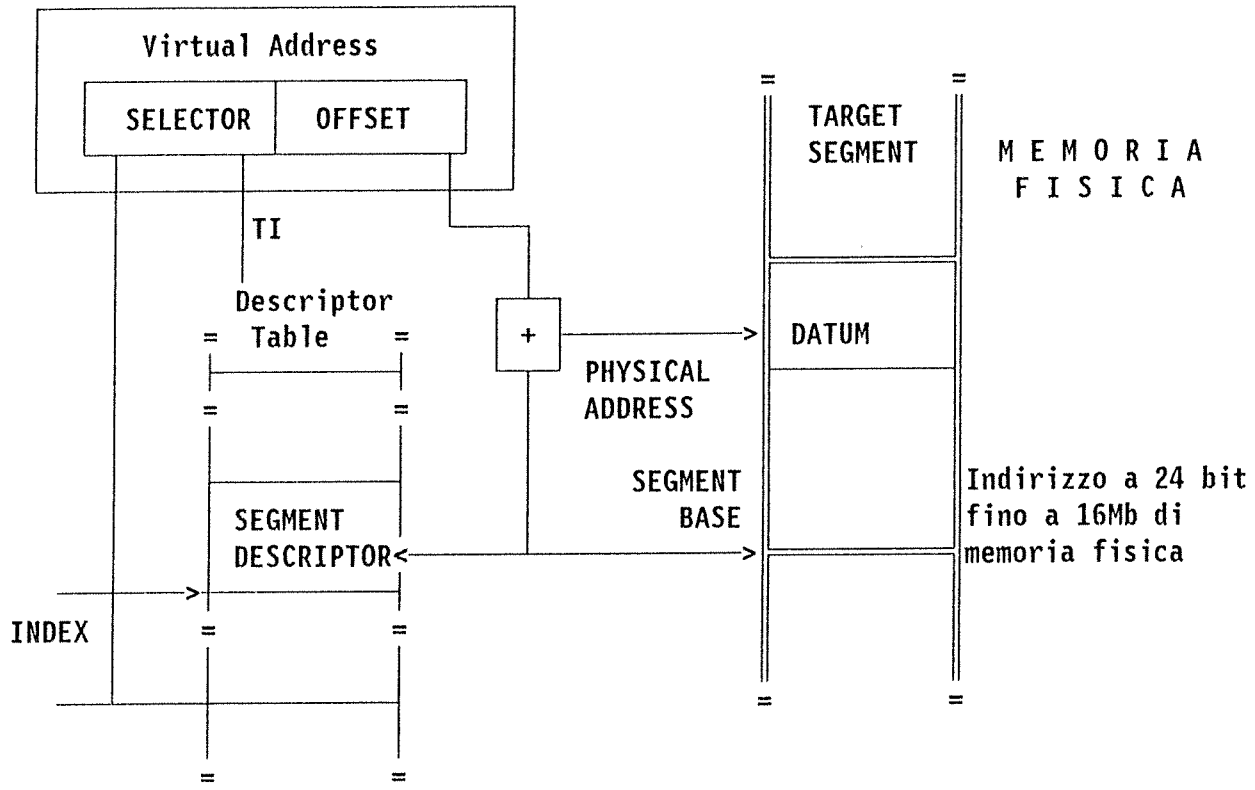
Segment Descriptors



Access Rights Byte

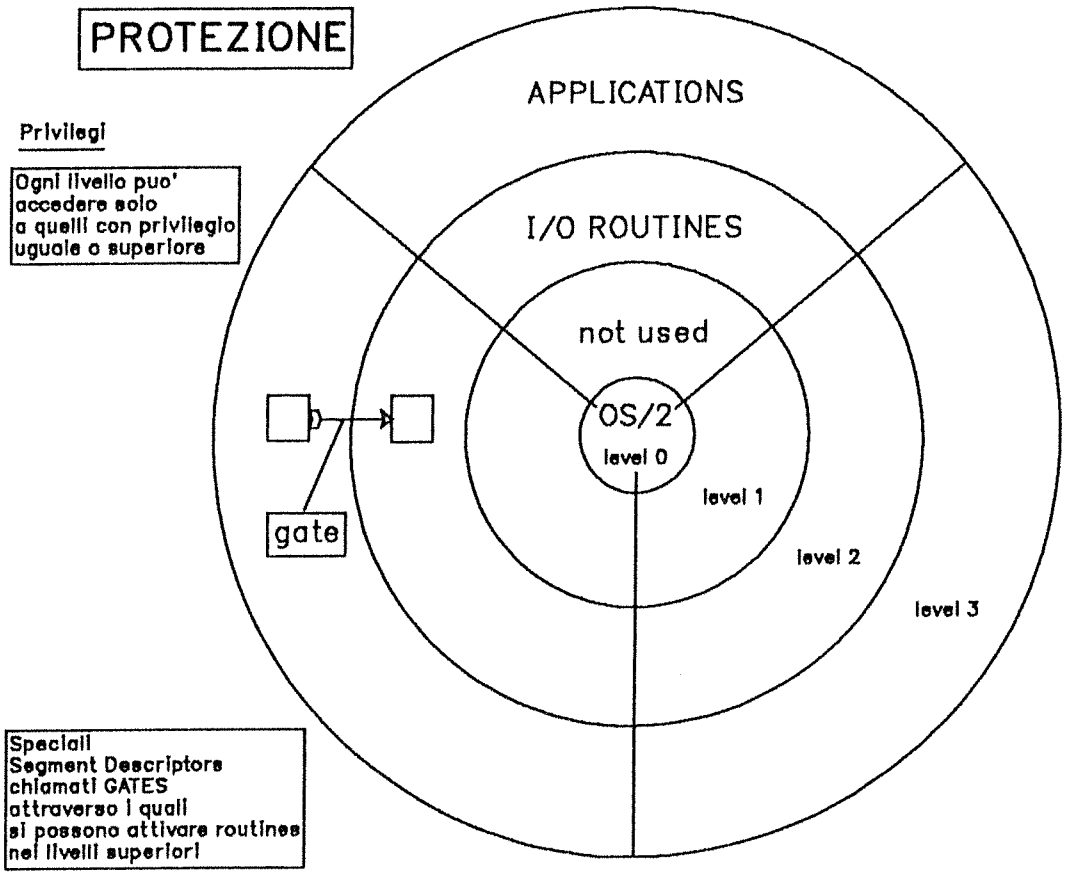
P	=	Present (1 bit)
DPL	=	Descriptor Privilege Level (2 bit)
S	=	Segment Descriptor
TYPE	=	Segment Type and access information (3 bit)
A	=	Accessed

Generazione INDIRIZZO FISICO



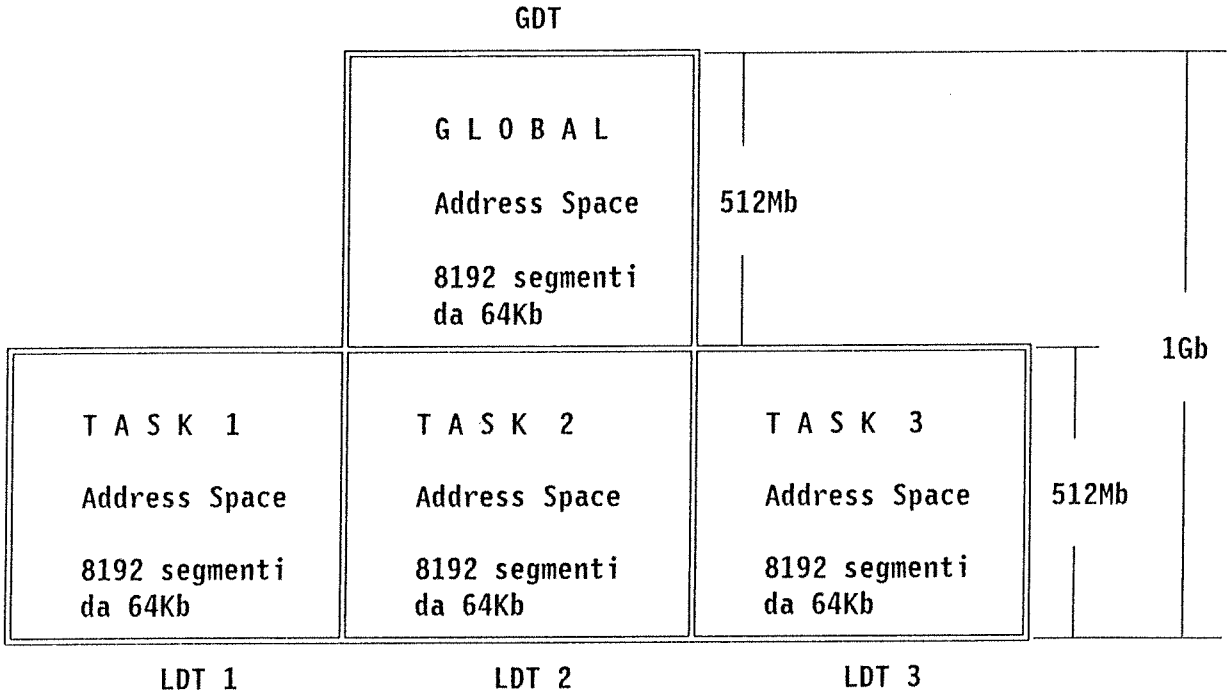
BASE	:	Valore a 24 bit che determina l'indirizzo base del segmento fisico - Accesso fino a 16Mb di memoria fisica
LIMIT	:	Valore a 16 bit che determina la dimensione (END) del segmento Fino a 64Kb
P	:	Determina se il segmento e' presente nella memoria fisica
A	:	Determina se il segmento e' stato acceduto
TYPE	:	Determina il tipo di segmento: CODE, DATA or SPECIAL
DPL	:	Valore a 2 bit che determina il livello di privilegio del segmento fino a 4 livelli di privilegio

LIVELLI DI PRIVILEGIO e PROTEZIONE



- Quattro livelli di privilegio
- Ogni livello accede solo a quelli con privilegio uguale o superiore
- Speciali Segment Descriptors, chiamati GATES, attraverso i quali si puo' accedere ai livelli inferiori
- Verifica che il CPL (Current Privilege Level - 2 bit meno significativi del CS corrente) sia congruente con il DPL del segm. acceduto - Accesso ai dati : EPL = max(CPL,RPL) <= DPL (Effective PL)

Address Spaces and Task Isolation



- Memoria virtuale : fino a 16384 segmenti di 64Kb - 1 Gigabyte
- Global Address Space : sistema operativo, subsystem, ...
- Meta' dello spazio di indirizzi e' globale, condivisa da ogni TASK
- L'altra meta' e' visibile solo dal TASK che possiede la LDT

ARCHITETTURA iAPX 386

- Processore a 32 bit
- Real Mode compatibile con le architetture iAPX 88, iAPX 86, iAPX 286-reale
- Protected Mode compatibile con la architettura iAPX 286-protetta
- Virtual Mode 8086 : permette di costruire macchine virtuali 8086
- 64-terabyte (2 alla 46 bytes) di memoria virtuale
- Segmenti fino a 4Gb : indirizzamento lineare a 32 bit
- Paginazione

PC DOS

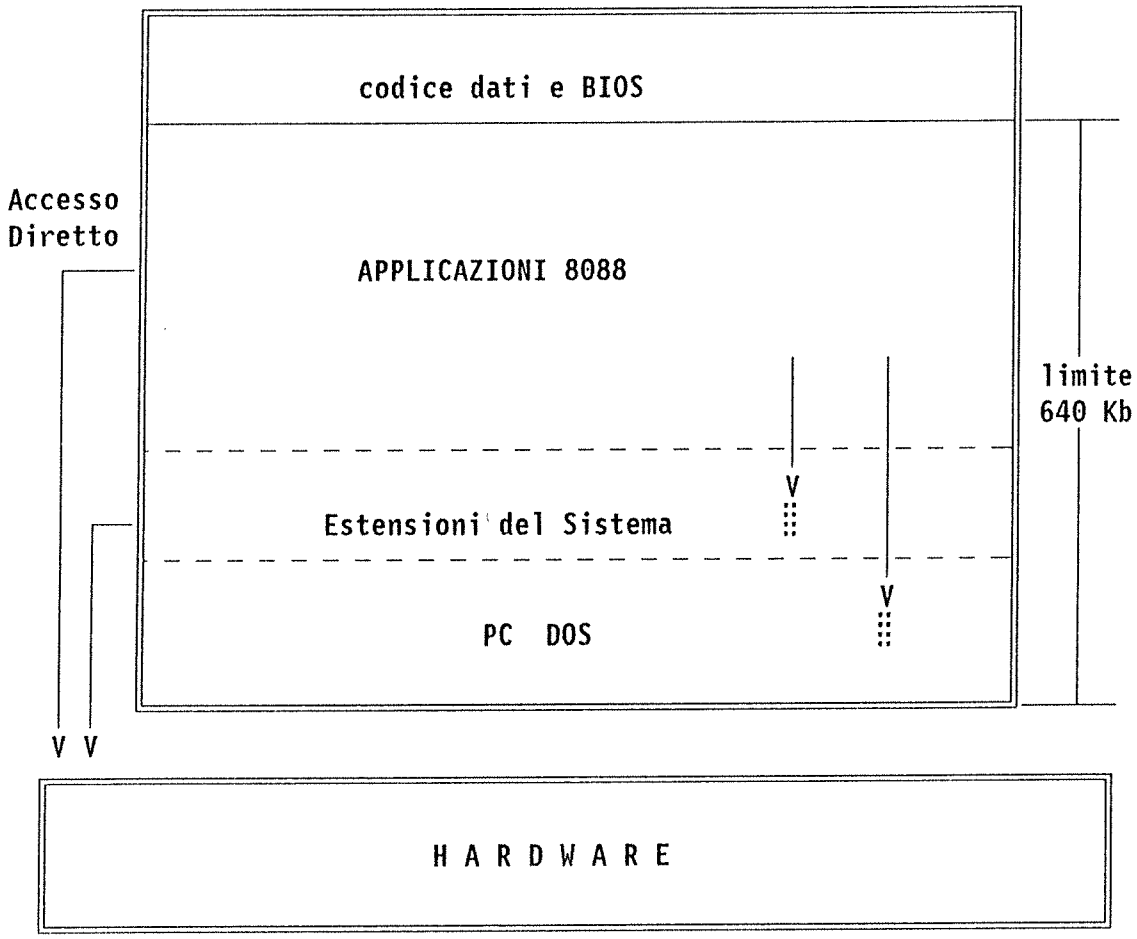
PC DOS : Caratteristiche

- Singolo Utente
- 640Kb di RAM
- Singolo Task
- Una Applicazione alla volta

PC DOS : Versioni

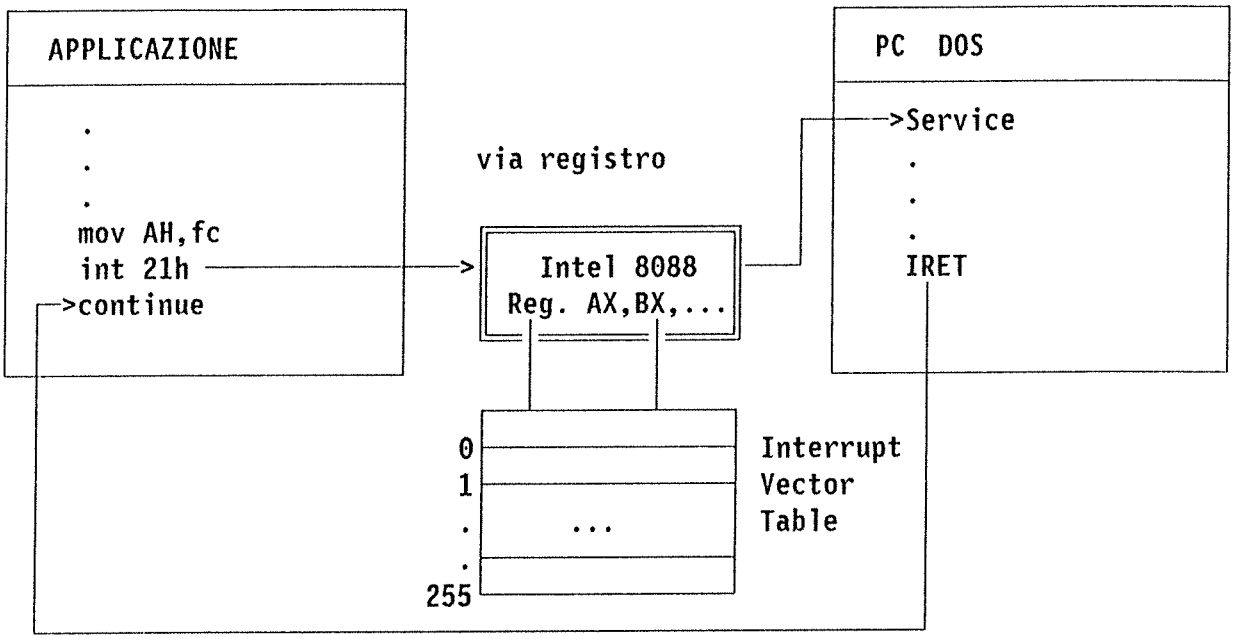
Versione	Anno	Funzioni
1.0	1981	Versione Iniziale
1.1	1982	supporto dischi 360Kb
2.0	1983	supporto dischi fissi, subdirectory, redirection
2.1	1984	support PCjr
3.0	1985	supporto PC AT
3.1	1985	supporto PC Network 1.0
3.2	1986	supporto PC Network 1.1 e dischi 720Kb
3.3	1987	supporto PS/2, Micro Channel, dischi 1.44Mb, 80386

Pc DOS : Struttura del sistema



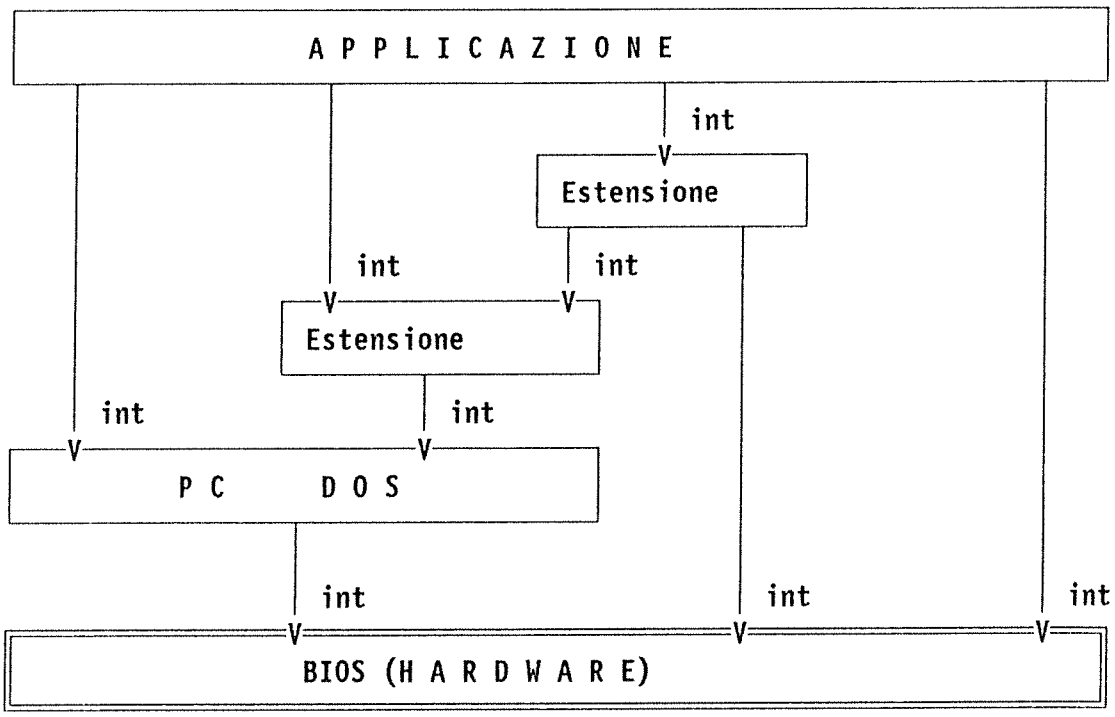
- Spazio di indirizzi come un unico blocco di 640Kb
- Nessuna distinzione tra il codice dell'applicazione e il codice di sistema
- Ogni programma puo' modificare qualunque porzione della memoria ed ha accesso diretto all'hardware

PC DOS : Application Program Interface



- API - Application Program Interface via Software Interrupt
- Manager "int 21h" non "rientrante"
- Passaggio Parametri via registro
- Facilmente riconfigurabile dalle applicazioni

PC DOS : Estensioni



- E' possibile estendere il DOS usando il meccanismo di interrupt software e la funzione Terminate and Stay Resident
- Il sistema non ha funzioni per fornire una struttura a questo approccio
- L'utente e' responsabile del caricamento delle estensioni

PC DOS : Limiti

- PC DOS : disegnato intorno al microprocessore Intel 8088 nei primi anni '80
- Spazio di indirizzamento limitato ad 1Mb : "barriera dei 640Kb"
- Memory Manager STATICO : non e' possibile muovere la zona di memoria in cui e' caricata una applicazione
- Nessuna garanzia di integrita' della memoria
- Le applicazioni possono accedere direttamente ai dispositivi di I/O
- Nessun supporto per il multitasking o il caricamento di piu' applicazioni
- Ambiente Aperto
- "Conflitti" fra prodotti programma
- Limite nel numero di funzioni contemporaneamente disponibili
- Complessita' nello sviluppo
- Facilita' di "deroga" dalle primitive del sistema

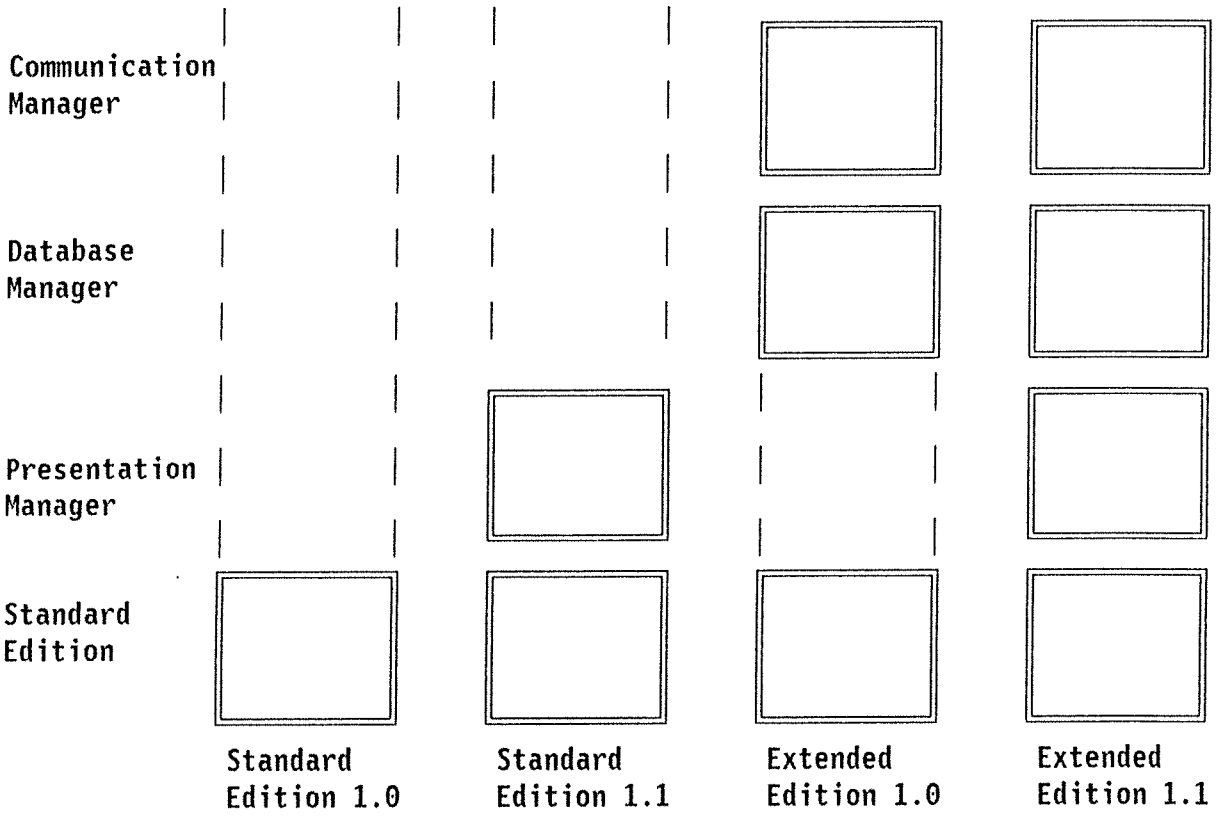
- Perfomances non accettabili per alcune funzioni di base (ex: screen I/O)
- Stretta dipendenza dall'architettura iAPX 88/iAPX 86
- Non rientrante
- Nessuna virtualizzazione della memoria
- In definitiva : UN PROGRAM LOADER

Operating System/2

OS/2 : Introduzione

- Disegnato per l'architettura protetta iAPX 286
- Gira sulle CPU 80286, 80386
- Sfrutta a pieno le possibilita' della CPU 80286
- Permette di usare fino a 16Mb di memoria fisica
- Memoria Virtuale
- Isolamento delle Applicazioni
- Gestione dell'accesso all'Hardware
- Gestione delle Risorse
- Multitasking ed Applicazioni Multiple
- Sistema Estendibile
- Coesistenza con l'ambiente "reale" PC DOS
- File System compatibile con PC DOS

OS/2 : Versioni



Versione	Data
OS/2 Standard Edition 1.0	12/87
OS/2 Standard Edition 1.1	10/88
OS/2 Extended Edition 1.0	07/88
OS/2 Extended Edition 1.1	11/88

OS/2 : Hardware

Operating System/2 gira su :

Bus	Modello	CPU
PC	PC AT 099, 239, 319, 339	80286
PC	PC XT 286	80286
PC	PS/2 mod 30-286	80286
MicroChannel	PS/2 mod 50, PS/2 mod 60	80286
MicroChannel	PS/2 mod 70	80386
MicroChannel	PS/2 mod 80	80386

OS/2 : Interprete dei Comandi

Interprete dei COMANDI
CMD.EXE COMMAND.COM

Shell OS/2 - Compatibility BOX

OS/2 : Comandi INTERNI RM=Real Mode PM=Protected Mode

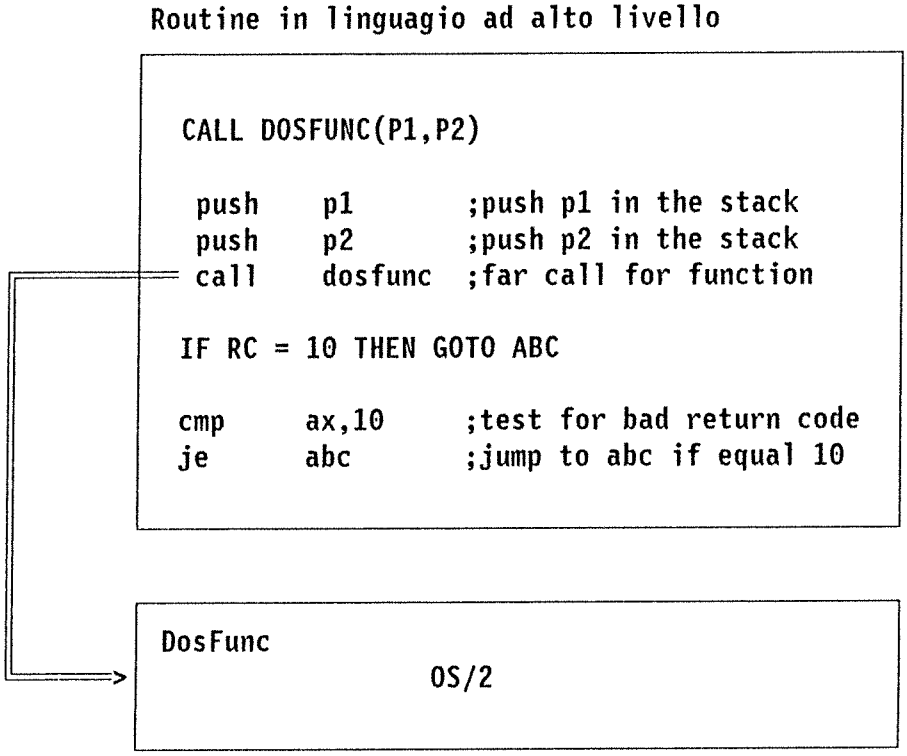
Comando	Modo	Descrizione
BREAK	RM	Check del BREAK on/off
CD	RM PM	Cambia la directory corrente
CHCP	RM PM	Cambia la System Codepage
COPY	RM PM	Copia dei files
DATE	RM PM	Cambia o visualizza la data del sistema
DIR	RM PM	Lista di una directory
DETACH	PM	Lancio di un programma in background
ERASE	RM PM	Cancellazione dei files
EXIT	RM PM	Uscita dall'interprete dei comandi
LABEL	RM PM	Cambia l'etichetta di un disco
MD	RM PM	Crea una directory
PROMPT	RM PM	Cambia il prompt
RD	RM PM	Rimuove una directory (vuota)
RENAME	RM PM	Rinomina i files
SET	RM PM	Set di una variabile di environment
START	PM	Start di una nuova Sessione
TIME	RM PM	Cambia o visualizza l'ora del sistema
VER	RM PM	Display della versione di OS/2
VERIFY	RM PM	Verifica scrittura su disco on/off
VOL	RM PM	Visualizza l'etichetta di un disco
d:	RM PM	Cambia il drive corrente (A:,B:,C:,...)

OS/2 : Application Programming Interface

OS/2 : API Model

- Il kernel di OS/2 gira al livello di privilegio 0 dell'80286
- Le funzioni *critiche* di OS/2 sono implementate nel kernel
- I programmi OS/2 comunicano con il sistema operativo attraverso delle **far calls** individuali
- Alcune di queste sono implementate attraverso dei **gates 80286** e implicano un cambiamento del livello di privilegio
- Ogni ingresso dell'API e' associato con un nome di funzione che corrisponde ad un *entry point* nel sistema
- L'API si accede con una istruzione **far call** simile alla chiamata di una subroutine (piuttosto che via interrupt software)
- Il sistema contiene una libreria denominata **DOSCALLS.LIB** che risolve i nomi simbolici

- L'API e' implementato attraverso una feature di OS/2 denominata **dynamic linking** : il link tra programma chiamante e routine chiamata viene risolto al momento del caricamento



- L'API e' ottimizzato per chiamate da linguaggio ad alto livello
  (**HLL calls** : C, PASCAL, ...)
- I parametri vengono passati via **stack** piuttosto che via registro
- L'API contiene un subset chiamato **DOS family API** (FAMAPI) che permette di scrivere programmi che girano sotto OS/2 (protected mode) e DOS (real mode)

- Convenzioni sui nomi :

  DOS Servizi di Sistema

  VIO Video Subsystem

  KBD Keyboard Subsystem

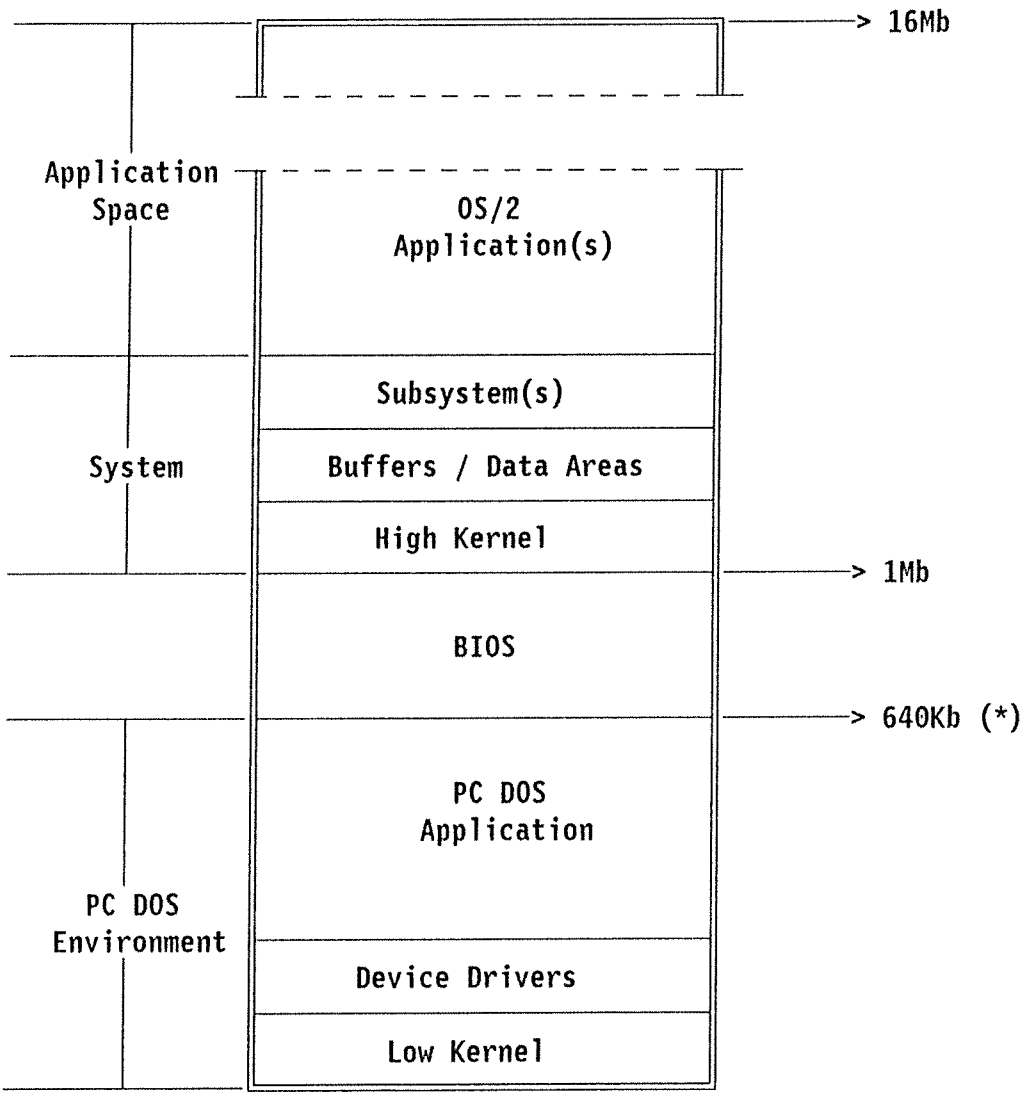
  MOU Mouse Subsystem

OS/2 : API - Vantaggi

- Usa un modello **far call** piuttosto che via **software interrupt**
- Passaggio dei parametri via stack piuttosto che via registro
- Nessuna neccessita' di un **router**
- Supporto ed ottimizzazione per linguaggi ad alto livello
- Modifiche dell'API non comportano modifiche dei programmi
- Le applicazioni non possono distinguere fra chiamate a routines *inline* e chiamate a *dynamic-linked* routines
- Facilita' di estensione e personalizzazione
- Implementazione del **DOS family API**

OS/2 : Memory Management

OS/2 : Mappa della Memoria

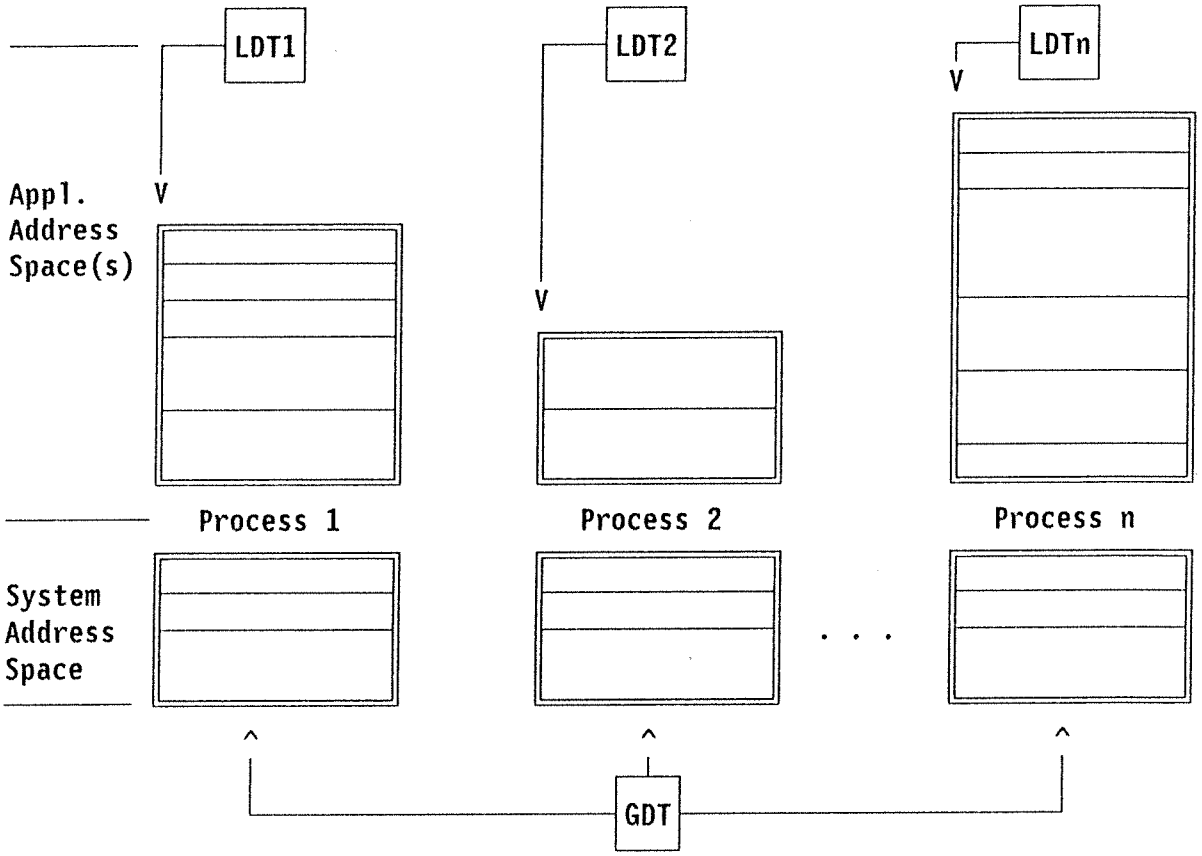


(*) determinato dai parametri RMSIZE e PROTECTONLY nel CONFIG.SYS

OS/2 : System Address Space

- Mantiene uno spazio comune di indirizzi per tutti i suoi processi
- **System Address Space** : selettori definiti nella **GDT** che mappano segmenti accessibili da tutte i processi
- Include il codice del nucleo, gli entry point dell'API, il codice e i dati dei device drivers
- Ogni processo nel sistema ha la stessa vista logica dei segmenti di memoria descritti nella **GDT**
- Tutte le applicazioni hanno uguale accesso alla memoria mappata nella Global Descriptor Table

OS/2 : Application Address Space



- OS/2 fornisce ad ogni processo il suo spazio di indirizzi locali
- Lo spazio di indirizzi di ogni processo viene mappato mediante una **LDT**
- OS/2 alloca dinamicamente segmenti aggiungendo ingressi alla **LDT**
- Ogni processo e' *isolato* dagli altri

OS2 : Loader

- OS/2 e' responsabile per il caricamento delle applicazioni in memoria
- Prima di cedere il controllo ad un processo OS/2 alloca ed inizializza una **LDT**
- Se il programma contiene riferimenti ad una dynamic-link library (**DLL**) il loader di OS/2 legge la libreria in memoria e crea gli ingressi nella LDT per ognuno dei suoi segmenti
- Se i segmenti del processo o della DLL sono gia' presenti in memoria il loader crea un ingresso nella LDT che punta al segmento preesistente
- OS/2 supporta segmenti **load on demand** : il loader crea gli ingressi nella LDT, ma li marca *non presenti* e li carica solo quando il processo cerca di accederli

OS/2 : Segment Motion

- OS/2 gestisce la memoria allocando e deallocando segmenti
- Durante la deallocazione si creano *buchi* nella memoria fisica
- Sfruttando le possibilita' di virtualizzazione della CPU 80286, OS/2 muove i segmenti per creare aree libere
- Il movimento dei segmenti avviene nel momento in cui ad OS/2 giunge una richiesta di memoria che non e' in grado di soddisfare
- L'utente puo' abilitare o disabilitare questo meccanismo con il parametro **MOVE/NOMOVE** dell'istruzione **MEMMAN** nel **CONFIG.SYS**

OS/2 : Segment Swapping

- OS/2 permette alle applicazioni di usare aree di memoria piu' grandi della memoria fisica installata nel sistema (**memory overcommit**)
- Ogni segmento nel sistema possiede un indicatore di **segment-not-present** nella descriptor table
- OS/2 usa questo indicatore per riconoscere se e' necessario caricare il segmento da un dispositivo esterno (disco fisso)
- Quando la memoria fisica libera non e' piu' sufficiente per soddisfare le richieste (anche dopo il segment motion), OS/2 applica un algoritmo di **LRU** (Least Recently Used) per determinare quali segmenti scartare
- Se necessario OS/2 copia (**swap**) i segmenti da o su un dispositivo esterno (il pathname indicato dall'istruzione **SWAPPATH** nel **CONFIG.SYS**)
- Il meccanismo puo' essere abilitato o disabilitato con il parametro **SWAP/NOSWAP** dell'istruzione **MEMMAN** nel **CONFIG.SYS**
- OS/2 non supporta la paginazione implementata nella iAPX 386

OS/2 : Shared Memory

- OS/2 supporta la comunicazione diretta tra processi mediante segmenti di memoria condivisi : **shared-memory segments**
- La Shared Memory e' la forma piu' semplice di **interprocess communications (IPC)**
- Quando un processo condivide un segmento di memoria, OS/2 incrementa un contatore associato al segmento; viceversa quando il segmento viene liberato
- OS/2 implementa due tipi di shared-memory segments : **globali** e **locali**
- **Global shared-memory segments** sono visibili a tutti i processi nel sistema : sono accedibili conoscendo il loro nome, una stringa **ASCIIZ** della stessa forma dei pathname : **\SHAREMEM\MYSEG**
- **Local shared-memory segments** sono una facility che permette a due processi di condividere memoria: e' necessario identificare i processi mediante il loro ID (**PID**)

OS/2 : Memory Management API Calls

API Call	*Descrizione*
DosAllocSeg	Alloca un segmento di memoria
DosReallocSeg	Modifica la dimensione di un segmento
DosFreeSeg	Libera un segmento di memoria
DosAllocShrSeg	Alloca un named shared-memory segment
DosGetShrSeg	Ottiene l'accesso ad un named shared segment
DosGiveSeg	Permette ad un altro processo di accedere un local shared segment
DosAllocHuge	Alloca un segmento di memoria multiplo (>64Kb)
DosGetHugeShift	Ottiene il selector increment per huge segment
DosReallocHuge	Modifica la dimensione di un segmento multiplo (huge)
DosCreateCSAlias	Crea un *alias* eseguibile per un data segment

DosMemAvail Ritorna la dimensione del piu' grande blocco libero di memoria

Nota

- **OS2MEMU PACKAGE** sul disco PCTOOLS
- Indispensabile per determinare la situazione della memoria
- OS/2 sembra non possedere le primitive per fornire tutte le informazioni sulla situazione della memoria fisica e virtuale
- Scritto da R.L.Cook : uno dei progettisti del gestore della memoria
- Sfrutta la struttura dei blocchi di controllo del kernel
- Dipende dalla versione di OS/2

OS/2 : Task Management

OS/2 : Tasking Model

- Tre concetti fondamentali:
  thread, **processo**, **sessione**
- **Thread** : identifica l'unita' di esecuzione conosciuta dal dispatcher di OS/2
- **Processo** : e' un insieme di uno o piu' threads a cui sono associate le risorse del sistema (memoria, files aperti, devices, ...)
- **Sessione** : e' un insieme di uno o piu' processi a cui e' associata una *console virtuale* (tastiera, video e mouse)
- Occorre distinguere tra applicazione e processo : il processo con il quale nasce un'applicazione puo' a sua volta creare processi addizionali che sono parte dell'applicazione

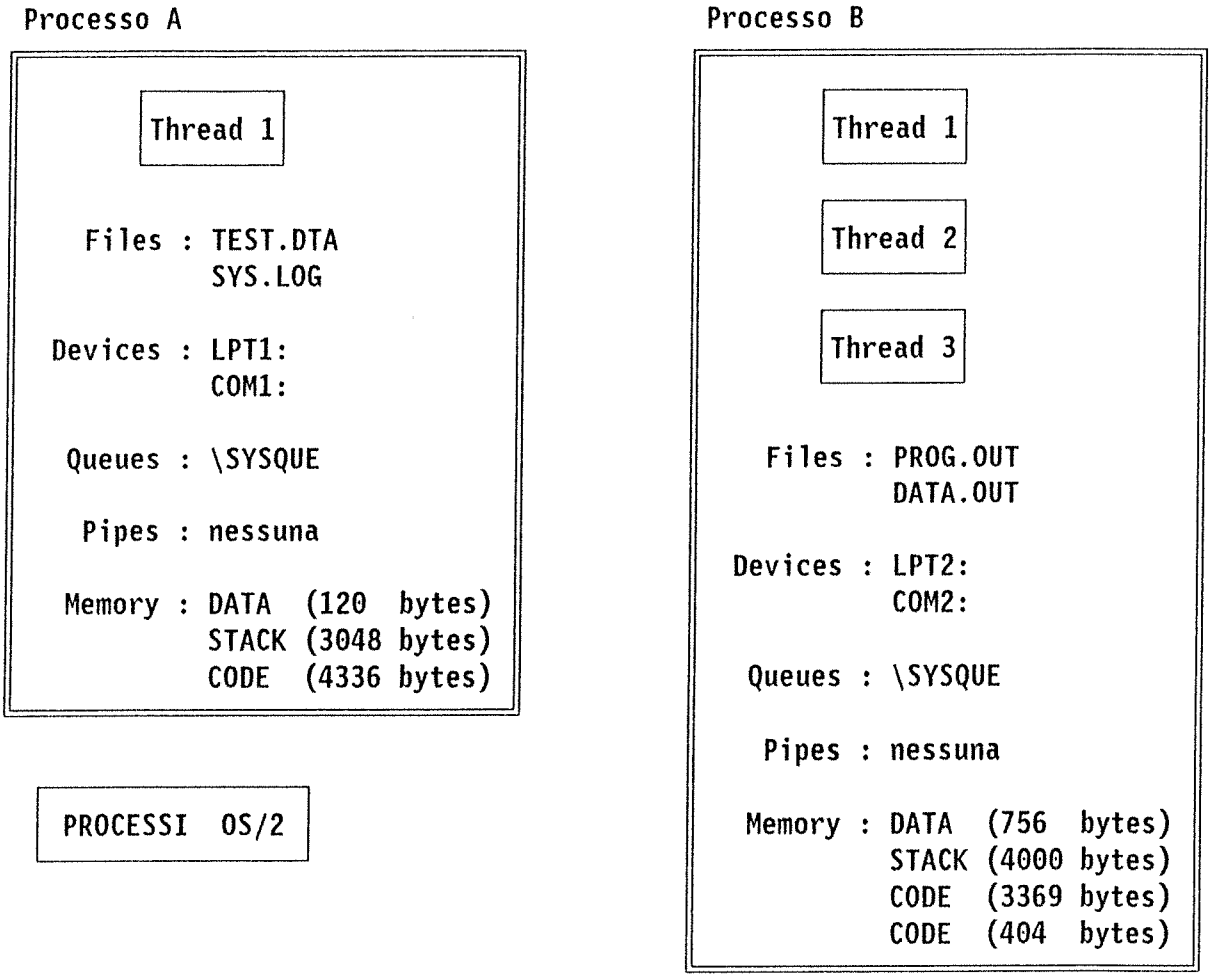
OS/2 : Threads

- L'unita' base di esecuzione di OS/2 e' il **thread**
- Ogni processo nel sistema possiede almeno un thread
- Il thread fornisce il codice eseguibile con un ambiente di esecuzione che consiste di : valori dei registri, stack e modo della CPU
- L'ambiente di esecuzione di un thread viene definito **contesto** del thread
- Ogni applicazione puo' impiegare uno o piu' threads
- OS/2 mantiene attivo un processo finche' almeno uno dei suoi threads e' in esecuzione
- All'interno di un processo i threads condividono tutte le risorse (per esempio hanno accesso agli stessi segmenti di memoria via un'unica **LDT**)
- Possono essere visti come subroutine separate e concorrenti dello stesso programma
- OS/2 usa un meccanismo software per il dispatch dei threads e non le features hardware della iAPX 286 (che pure esistono) vedi [3] pag.98

- I threads non possiedono una struttura gerarchica e sono individuati dal ThreadID, unico all'interno di un processo

OS/2 : Processi

- Le risorse in OS/2 sono raggruppate in unita' logiche chiamate **processi**
- Quando OS/2 attiva un programma per prima cosa crea un processo, quindi carica il programma e da' il controllo ad un thread per mettere in esecuzione il codice
- Ogni processo possiede il suo insieme di risorse, in particolare ha almeno un thread, dei files aperti, devices, pipes, queues e segmenti di memoria
- I processi possiedono una struttura gerarchica : un processo che ne esegue un secondo viene chiamato **padre**, mentre il processo eseguito viene definito **figlio**
- Il padre mantiene il controllo dei suoi figli (per esempio puo' terminarli), mentre i figli ne ereditano alcune risorse
- I figli ereditano : files, pipes, standard devices (stdin e stdout), character devices, environment segment, priorita' e sessione



PROCESSI OS/2

- Sono identificati dal **PID** (Process ID), unico nel sistema
- I processi vengono creati con la primitiva API **DosExecPgm**
- Un processo puo' creare una sezione di codice che verra' eseguita quando termina : **exitlist**

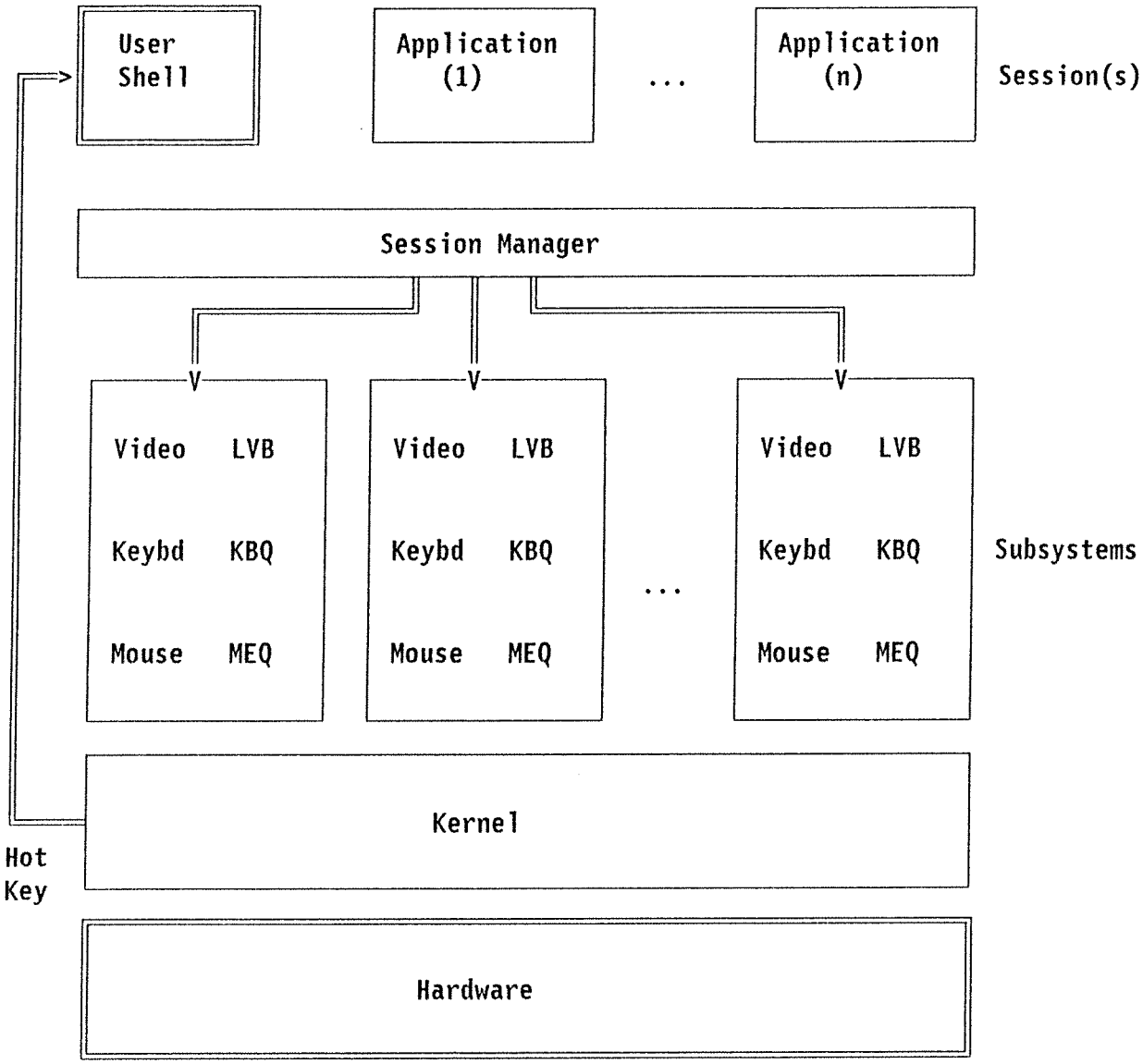
OS/2 : Task Management API calls

Process API Call	*Descrizione*
DosExecPgm	Esegue un programma creando un processo
DosKillProcess	Termina uno o piu' processi
DosCWait	Wait per il termine di un processo figlio

Thread API Call	*Descrizione*
DosCreateThread	Crea un nuovo thread di esecuzione
DosExit	Termina il thread corrente
DosResumeThread	Riprende l'esecuzione di un thread
DosSuspendThread	Sospende l'esecuzione di un thread
DosEnterCritSec	Permette ad un solo thread del processo di girare
DosExitCritSec	Permmette a tutti i threads del processo di girare

OS/2 : Sessioni

- OS/2 e' un sistema *multitasking* che permette l'uso contemporaneo di piu' applicazioni
- Per gestire quale applicazione e' visibile sullo schermo e riceve l'input da tastiera e mouse, OS/2 usa uno speciale componente denominato **SESSION MANAGER**
- Non e' un servizio del nucleo : e' implementato come un **subsystem**
- L'utente seleziona quale applicazione portare in foreground mediante una interfaccia : la **Session Manager Shell**
- La Session Manager Shell e' a sua volta un'applicazione che gira in una propria sessione ed interagisce con l'utente via **Hot Key** (Standard Edition 1.0)
- Il Session Manager si occupa di associare il giusto contesto di subsystems con l'applicazione di **foreground**



* LVB : Logical Video Buffer
* KBQ : Keyboard Queue
* MEQ : Mouse Event Queue

OS/2 : Session Manager Shell

OS/2 Standard Edition 1.0

Aggiornamento | F1=Guida

Selettore dei programmi

Per scegliere un programma usare i tasti <, >, ^ o V, quindi premere Invio.
Per selezionare Aggiornamento premere F10 ed Invio.

Avviare un programma
---

* Introduzione all'OS/2
* Richiesta comandi OS/2

Passare ad un programma in esecuzione
---

* Richiesta comandi DOS
* CP78 Emulator
* Richiesta comandi OS/2

OS/2 : Time Slice

- OS/2 ha un **preemptive time-slicing dispatcher** : tutti i threads presenti nel sistema ricevono una quantita' di tempo fissa (**time slice**) della CPU
- Al termine del time slice OS/2 puo' decidere di togliere il controllo al thread : cio' puo' accadere anche tra due istruzioni di un programma
- I thread devono porre attenzione quando accedono risorse condivise appartenti ad un processo : ad essi puo' essere tolto il controllo mentre aggiornano una struttura di dati
- L'utente puo' configurare il valore minimo e massimo del time-slice usando il parametro **TIMESLICE** nel **CONFIG.SYS**

OS/2 : Priorita'

- Tutti i threads nel sistema competono per il tempo di CPU
- Tuttavia esistono dei thread che devono avere la precedenza sugli altri : ad esempio un thread che controlla una linea di comunicazione dovrebbe avere una priorita' maggiore di uno che controlla una stampante
- OS/2 implementa uno schema a piu' livelli di priorita' con variazione dinamica e *Round-Robin* all'interno dello stesso livello di priorita'
- Cio' significa che tutti i thread hanno associato un livello di priorita' : un thread ad un livello piu' alto, pronto per girare, ottiene **sempre** la CPU **prima** di un thread ad un livello piu' basso
- OS/2 usa **classi di priorita'** per separare thread con caratteristiche differenti. Esistono **quattro** classi di priorita' :
  - **Time Critical**
  - **Foreground**
  - **Regular**
  - **Idle**

OS/2 : Classi di Priorita'

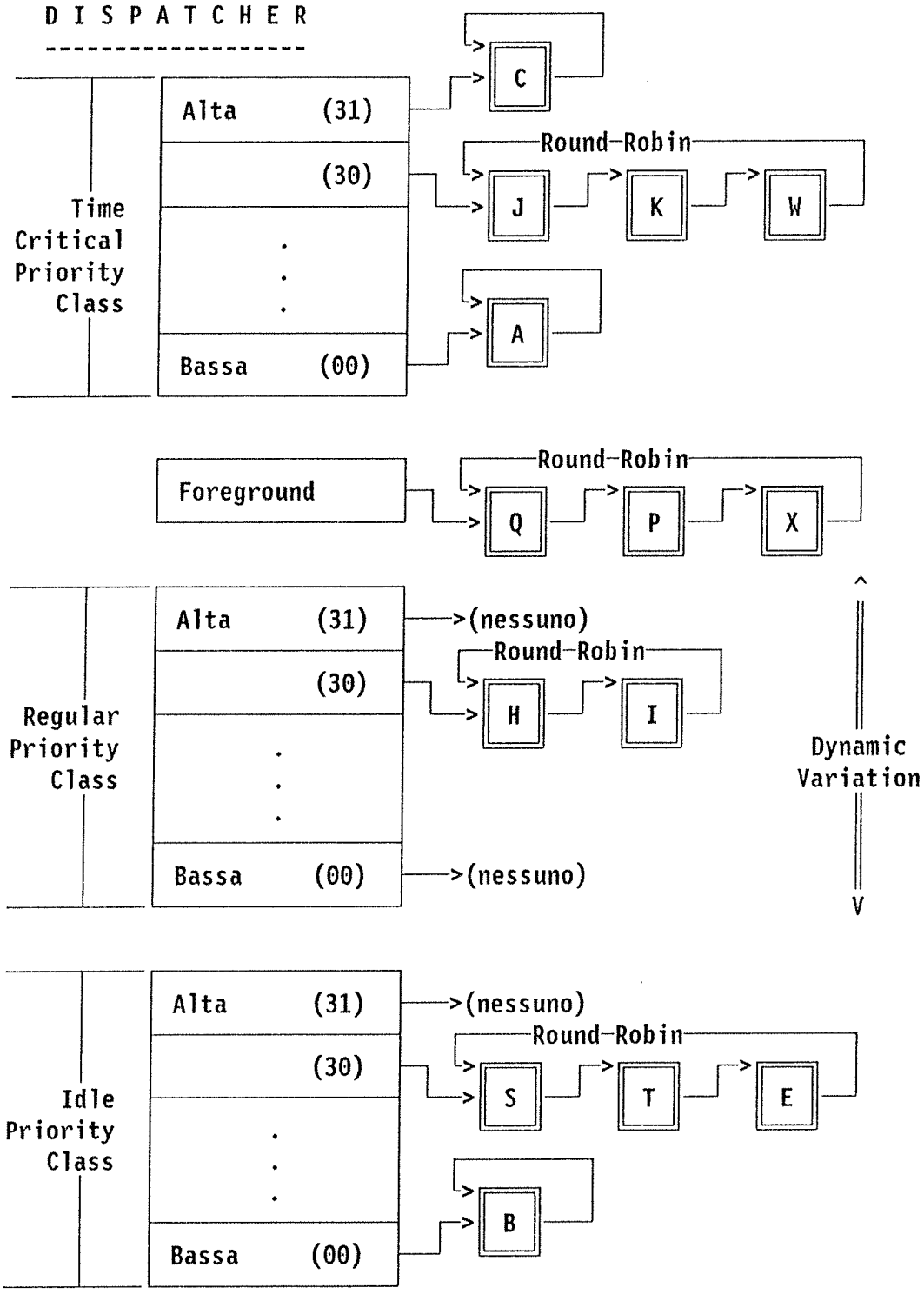
- La classe **Time Critical** ha **32.** livelli di priorita' : thread che richiedono attenzione immediata (ad esempio real time) dovrebbero girare in questa classe

- La classe **Foreground** ha **1** solo livello di priorita' : il processo che possiede lo schermo (*sessione di foreground*) gira sempre in questa classe

- La classe **Regular** ha **32** livelli di priorita' : il sistema varia dinamicamente le priorita' in questa classe in dipendenza dell'uso dei dispositivi di I/O e della CPU
  (vedi le istruzioni **MAXWAIT** e **PRIORITY** nel **CONFIG.SYS**)

- La classe **Idle** ha **32** livelli di priorita' : e' la classe a priorita' piu' bassa

- Un processo puo ottenere e cambiare la classe e il livello di priorita' dei suoi threads mediante le primitive API, **DosGetPrty** e **DosSetPrty**


D I S P A T C H E R
Time Critical Priority Class
Alta (31)
(30)
Bassa (00)
C
Round-Robin
J
K
W
A
Foreground
Round-Robin
Q
P
X
Regular Priority Class
Alta (31)
(30)
Bassa (00)
->(nessuno)
Round-Robin
H
I
->(nessuno)
Dynamic Variation
Idle Priority Class
Alta (31)
(30)
Bassa (00)
->(nessuno)
Round-Robin
S
T
E
B

OS/2 : Semafori

- Poiche' ad ogni thread puo' essere tolto il controllo in qualsiasi punto dell'esecuzione, OS/2 prevede delle funzioni per **serializzare** l'accesso alle risorse condivise
- I **Semafori** sono delle strutture di dati la cui proprieta' e' garantita da OS/2 ad un solo thread alla volta.
- Per esempio se due thread, all'interno dello stesso processo, aggiornano una stessa struttura di dati possono usare un **semaforo RAM** : OS/2 permette ad un thread di ottenere il semaforo e blocca il secondo thread fino a che il primo non rilascia il semaforo
- Esistono **semafori di sistema** e **semafori RAM** : i primi sono utili nella comunicazione tra processi, mentre i secondi vengono usati nella comunicazione tra threads di uno stesso processo
- I semafori RAM sono delle strutture di dati visibili dal programma (**long int**) e possono assumere due stati : **set** e **clear** (*level sensitive*)
- Il passaggio dallo stato **set** allo stato **clear** *risveglia* tutti i thread che sono in attesa sul semaforo

OS/2 : Semaphore API Calls

API Call	Descrizione
DosSemClear	Clear di un semaforo
DosSemSet	Set di un semaforo
DosSemWait	Wait su un semaforo
DosSemSetWait	Wait e Set di un semaforo (*atomica*)
DosSemRequest	Ottiene un semaforo
DosMuxSetWait	Wait su una lista di semafori
DosSemOpen	Apre un semaforo di sistema
DosSemClose	Chiude un semaforo di sistema
DosSemCreate	Crea un semaforo di sistema

OS/2 : Interprocess Communications (IPC)

- OS/2 prevede un ricco insieme di interfacce per permettere la comunicazione tra processi
- OS2 usa i **flags** per segnalare gli eventi ad un processo
- Altri meccanismi a cui abbiamo gia' accennato sono i **semafori** e la **shared memory**
- Le **pipes** sono delle stream non strutturate di dati, simili ad un file sequenziale. Essenzialmente sono delle code FIFO.
- Le **queues** sono liste di elementi che possono essere accedute in modo FIFO, LIFO o per priorita'. I dati non vengono copiati tra la coda e il processo : al processo viene fornito l'indirizzo dell'elemento. La coda puo' essere letta senza distruggerne gli elementi

OS/2 : IPC API Calls

Flags API Call	*Descrizione*
DosSetSigHandler	Registra un Signal Handler
DosFlagProcess	Manda un Flag

Pipes API Call	*Descrizione*
DosMakePipe	Crea una pipe
DosRead	Legge da una pipe
DosWrite	Scrive in una pipe
DosClose	Chiude una pipe

Queues API Call	*Descrizione*
DosCloseQueue	Chiude una queue
DosCreateQueue	Crea una queue
DosOpenQueue	Apre una queue
DosPeekQueue	Esamina un elemento di una queue
DosPurgeQueue	Cancella tutti gli elementi di una queue

DosQueryQueue	Determina il numero di elementi di una queue
DosReadQueue	Legge un elemento da una queue
DosWriteQueue	Scrive un elemento in una queue

OS/2 : Sommario

- Ampia memoria reale e virtuale
  Gestione dinamica della memoria
- Memory overcommit
- Applicazioni Multiple
- Multitasking
- Interprocess Communications
- Isolamento dei processi
- Livelli di Privilegio
- Rispetto delle *regole* : API
- Dynamic linked call/return interface
  Supporto linguaggi ad alto livello
- Compatibilita' del file system con PC DOS
- Ambiente di esecuzione PC DOS (FAMAPI)

BIBLIOGRAFIA

Bibliografia

[1] **iAPX 286 Programmer's Reference Manual** *Intel Corporation, 3065 Bowres Avenue, Santa Clara, CA 95051* 1985

[2] **iAPX 286 Operating System Writer's Guide** *Intel Corporation, 3065 Bowres Avenue, Santa Clara, CA 95051* 1985

[3] *IBM System Journal 27, No.2* 1988

[4] **Operating System/2 Technical Reference, Volume I, 84X1434** *IBM Corporation* Ottobre 1987

[5] **Operating System/2 Technical Reference, Volume II, 84X1440** *IBM Corporation* Ottobre 1987

[6] **Operating System/2 Programmer's Toolkit, 84X1448** *IBM Corporation* Settembre 1987

[7] E. Jacobucci **OS/2 Programmer's Guide** *Osborne McGraw-Hill, Berkeley, CA* 1988

[8] J.I. Krantz, A.M.Mizel, R.L.Williams **OS/2 Features, Functions and Applications** *John Wiley & Sons, Inc., NY* 1988

[9] **OS2BOOKS FORUM** *Disco IBMPC (shadows TOOLS at UITHONE)*