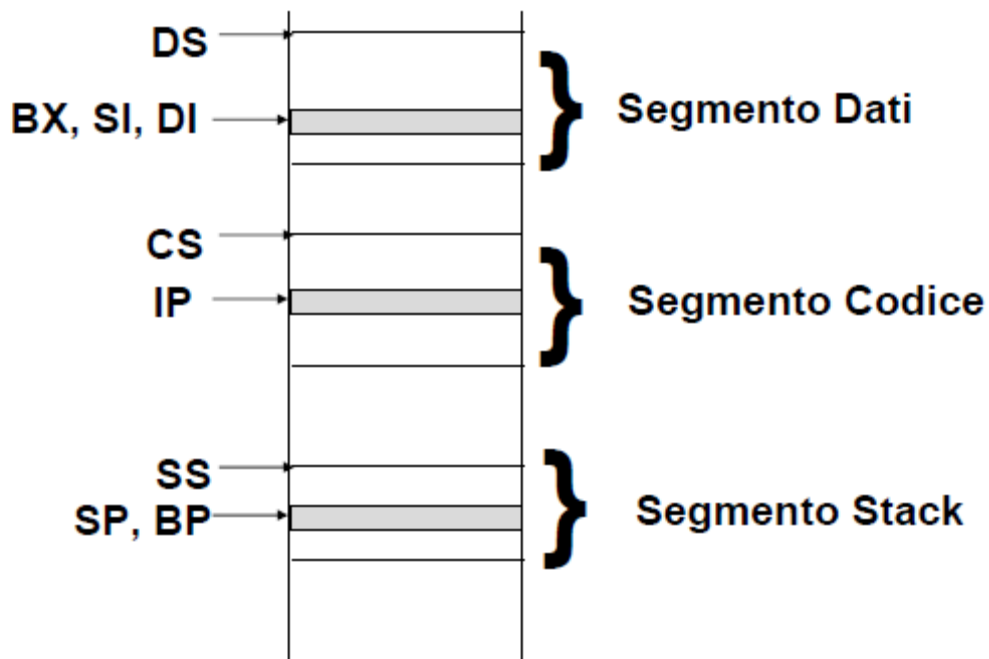


Lezione 24-09-08

Segmentazione della memoria (in RM) (vedi slide da 68 a 72)



Il processore è un modello a memoria segmentata non protetta. La segmentazione è visibile dal programmatore (o dal programma), mentre la paginazione è un meccanismo più nascosto.

La segmentazione è stata introdotta per razionalizzare le protezioni e le articolazioni dei programmi a livello compilatore, per separare i moduli di codice e di dati dai moduli di stack. Lavora sulla memoria logica (cioè la memoria vista dalle istruzioni).

La memoria può essere considerata come divisa in segmenti di 64 KB che iniziano con indirizzi multipli di 16. I gruppi di 16 byte che iniziano con indirizzi multipli di 16 vengono detti paragrafi.

La gestione della segmentazione in RM è diversa da quella in PM.

Calcolo degli indirizzi (vedi slide 73)

Abbiamo visto precedentemente quali sono i registri di segmento e quali sono i registri di indirizzo.

L'indirizzo generico è formato da una parte che indica il segmento e da una parte che indica l'offset, cioè il punto in cui si deve andare a cercare partendo dalla testa del segmento:

Registro di segmento | Registro di offset

Il registro di segmento in modo reale è visibile al programmatore.

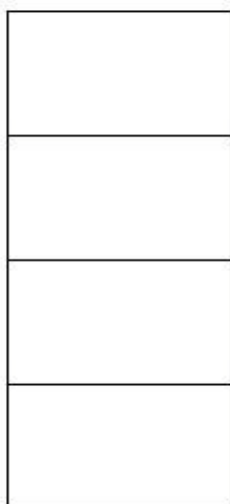
In altre architetture, i registri di segmento non sono visibili all'utente normale che vede soltanto l'offset.

È come se, ad esempio, non si potesse scrivere dentro al registro di segmento le istruzioni MOV: queste sono istruzioni che possono essere eseguite soltanto con dei privilegi (e non con quelli dell'utente tradizionale). Questo fa sì che tali registri vengano manipolati unicamente dal sistema operativo e dal Loader dei processi. Quando si deve caricare in memoria un programma, il sistema operativo, e in particolare la parte che si occupa del caricamento dei programmi, si fa carico della funzione dello stato della memoria e definisce in quale porzione di memoria logica va a finire il programma.

Esistono coppie fisse per segmento e offset (per esempio, il PC (Program Counter) è dato dalla coppia CS:IP; SP (Stack Pointer) è dato da SS:SP; DS:BX indica l'indirizzo di un dato).

Essendo ogni registro di segmento e di offset da 16 bit ciascuno sembra che l'indirizzo sia di 32 bit.

In realtà, l'indirizzo effettivo è di 16 bit perché si utilizza uno schema di memoria a blocchi sovrapponibili in cui lo spazio non sovrapponibile è di 16 byte.



**Memoria a
blocchi non
sovrapponibili**

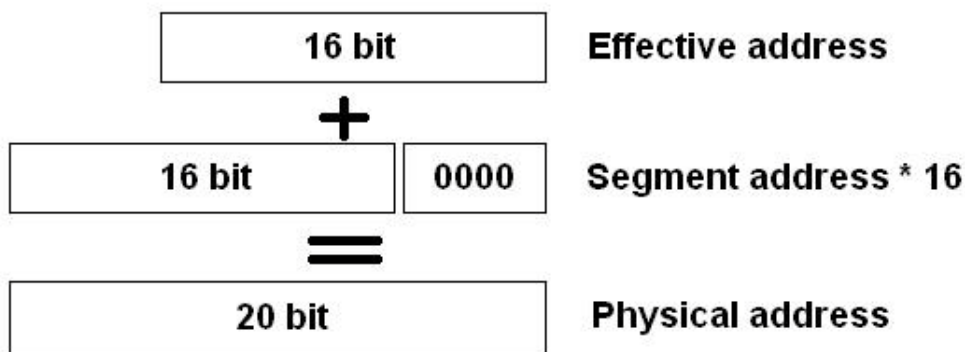


**Memoria a
blocchi
sovrapponibili**

In una memoria a blocchi sovrapponibili sfalsati tra di loro di 16 byte e con una dimensione di 1 MB, l'indirizzo fisico di testa di ciascun segmento sarà di 16 bit di 4 bit posti a 0 per indicare la testa.

<FFFF>_{hex} <0000>_{bin} → ABUS

In generale, ogni volta che l'8086 deve generare un indirizzo fisico da mettere sull'ABUS deve effettuare un'operazione di somma tra il contenuto di un registro BX o di un registro puntatore (che ci dà l'indirizzo di offset) e il contenuto di un registro di segmento. La somma avviene dopo aver "shiftato" di 4 posizioni il registro di segmento:



Partendo dall'indirizzo di testa di un segmento si può trovare una posizione al suo interno sommando all'indirizzo di testa l'offset della posizione interna.

<0102> <0011> (esadecimale) indica, per esempio, che nell'address bus andrà <01031> che è appunto la somma tra il primo shiftato e il secondo.

Consideriamo, per esempio, uno spazio di memoria di un 1 MB: sappiamo che l'ultimo segmento inizia 16 byte prima della fine della memoria e quindi all'indirizzo FFFF0. Se a FFFF0 sommiamo un numero maggiore di 16 si prenderà l'indirizzo ripartendo dalla testa del segmento.

Per concludere, l'ABUS ci indica l'indirizzo di una cella di memoria dato da un indirizzo di segmento e un offset. Questo indirizzo identifica la cella che è unica, ma le combinazioni possono essere molteplici dato che l'indirizzo è costituito da una coppia di valori che possono assumere configurazioni diverse.

Organizzazione della memoria (vedi slide da 74 a 76)

L'organizzazione della memoria è caratterizzata da limitazioni hardware definite dal progettista.

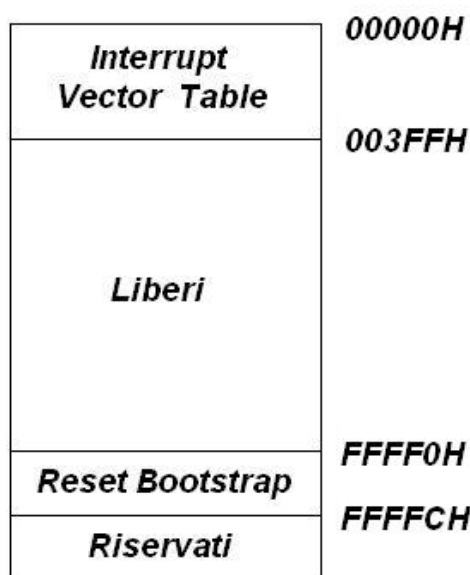
Il PROGRAM COUNTER (PC), o INSTRUCTION POINTER (IP), è un registro della CPU la cui funzione è quella di conservare l'indirizzo in memoria dell'istruzione successiva da eseguire.

Il PC, per esempio all'avvio della macchina, deve essere inizializzato ad un valore cui corrisponde la prima istruzione da eseguire.

La scelta è normalmente quella di definire l'indirizzo di partenza della prima istruzione o in testa o in coda nella memoria.

La scelta della piattaforma x86 (a partire dalla versione 8086 fino ad arrivare ai dual-core e ai quad-core) è di partire con l'istruzione dal fondo della memoria. Per esempio, l'8086 dopo la ricezione del segnale di reset si procura il codice della prima istruzione da eseguire all'indirizzo a 20 bit FFFF0H, cioè 16 byte prima della fine della sua memoria. Nel Pentium, con ABUS a 32 bit, l'indirizzo iniziale sarà FFFFFFF0. A quella locazione deve essere presente una memoria elettronica non volatile, ROM o EPROM. La prima istruzione farà saltare (JMP) ad un piccolo programma di test e di caricamento residente in ROM detto "POST" (Power On Self Test).

Alcune parti di memoria quindi sono dedicate:



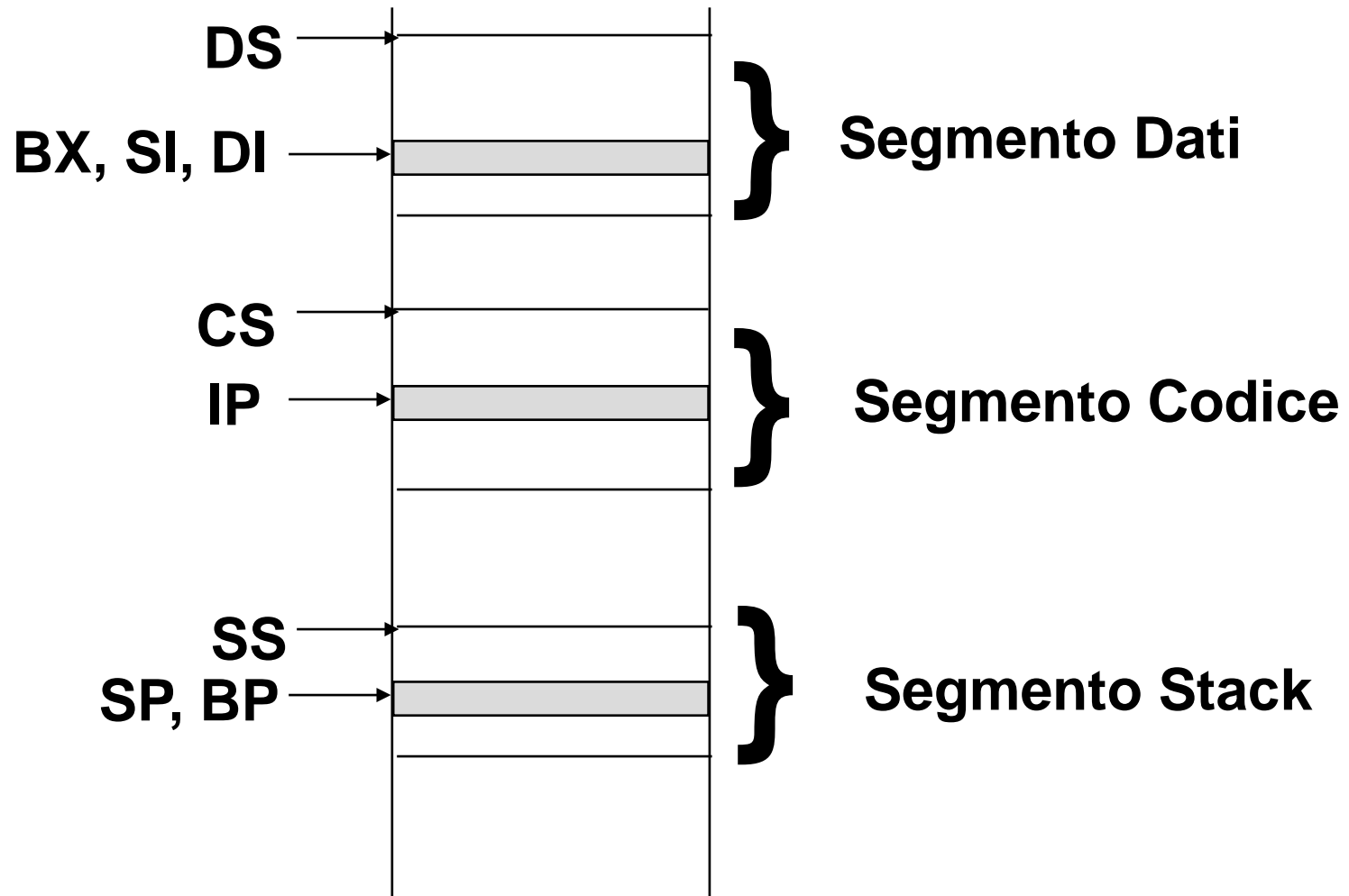
Ad esempio, la memoria dall'indirizzo 00000H allo 003FFH è riservata, in quanto contiene l'Interrupt Vector Table.

Gli indirizzi da FFFF0H a FFFFBH sono utilizzati per contenere un'istruzione di salto alla routine di caricamento del programma di bootstrap.

Le locazioni da FFFFCH a FFFFFFFH sono invece riservate ai dati di versione del BIOS.

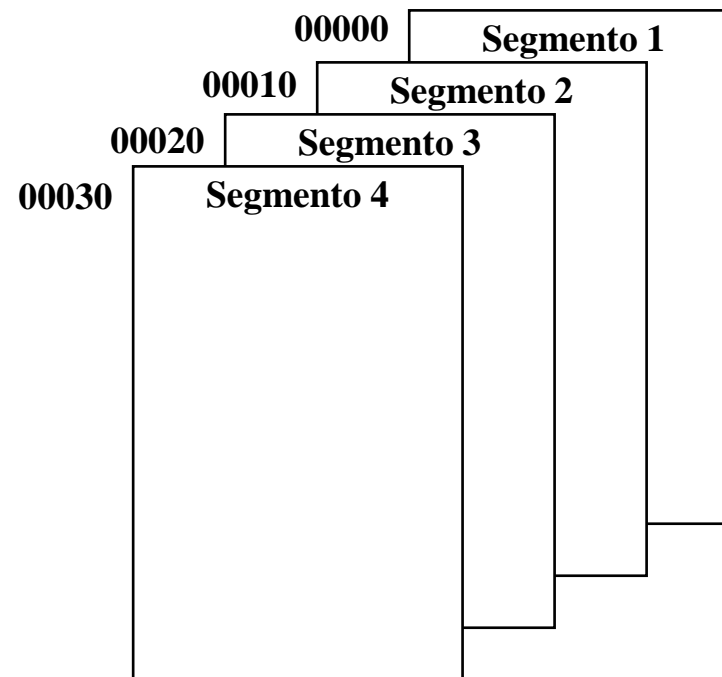
LEZIONE 24 Settembre 2008

Segmentazione della Memoria



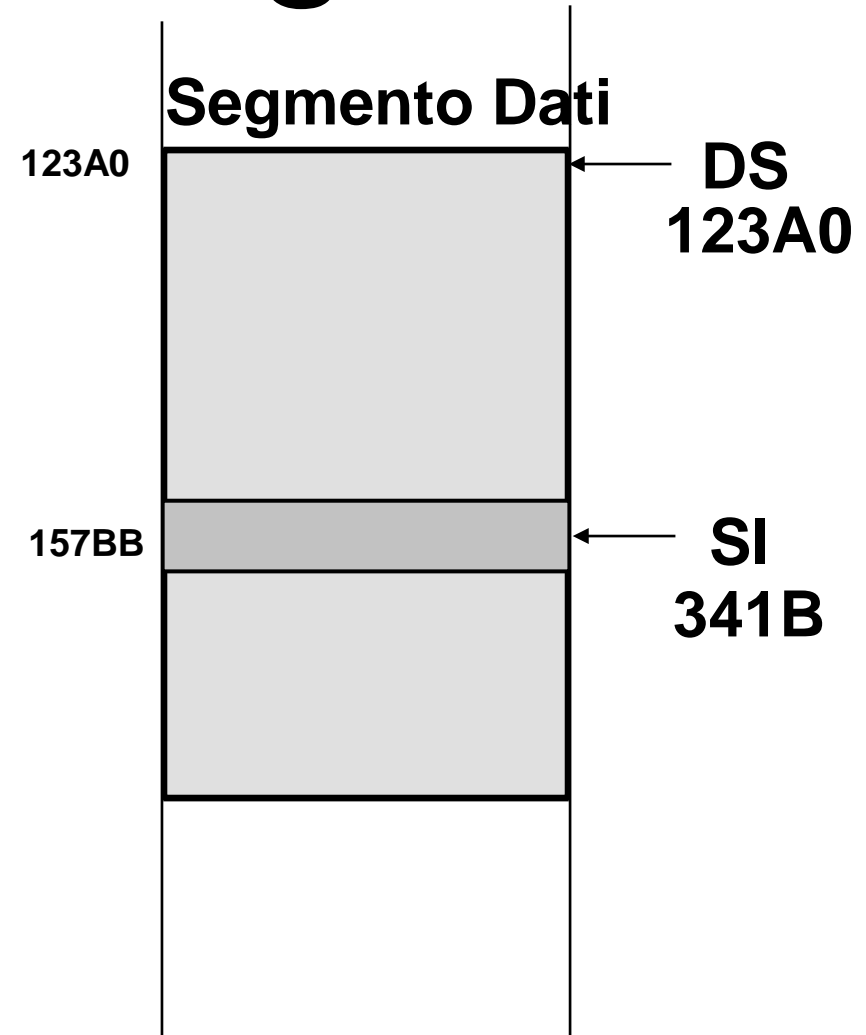
Segmenti

La memoria può essere considerata come organizzata in *segmenti*, ognuno di dimensione pari a 64 Kbyte. Tutti i segmenti cominciano ad indirizzi multipli di 16.



Uso dei Segment Register

Una volta caricato l'indirizzo di testa del segmento in un segment register, tutti gli indirizzi all'interno del segmento sono esprimibili attraverso un registro contenente l'offset.



Paragrafi

I gruppi di 16 byte che iniziano ad indirizzi multipli di 16 si definiscono *paragrafi*. La memoria è quindi organizzata in paragrafi.

Paragrafo 0		00000h
Paragrafo 1		00010h
Paragrafo 2		00020h
Paragrafo 3		00030h
Paragrafo 4		00040h

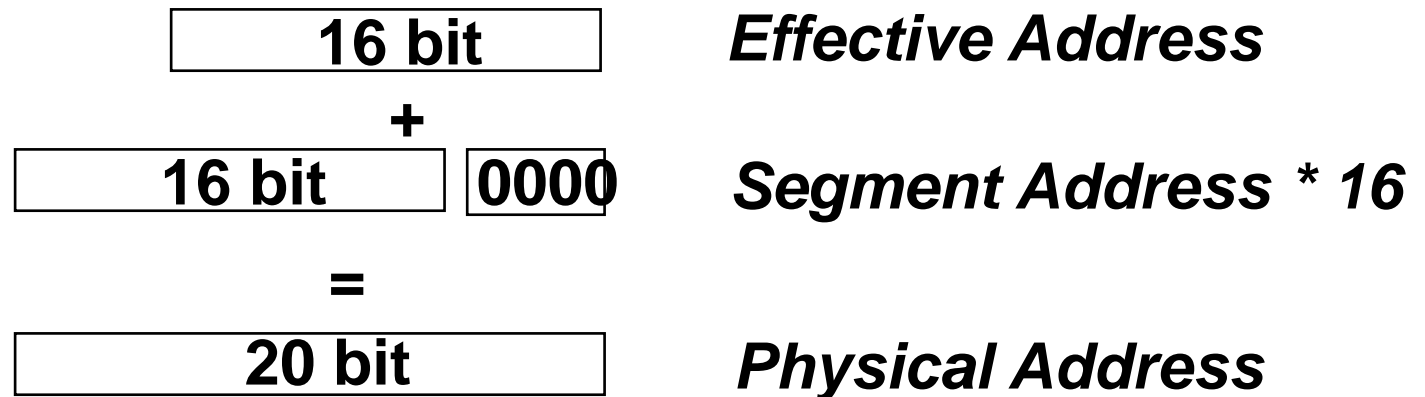
Vantaggi della Segmentazione

- Spazio di indirizzamento pari a 2^{20} , ma indirizzi su 16 bit
- Separazione tra dati, codice e stack
- Possibilità di avere più segmenti dello stesso tipo (dati, codice o stack)
- Possibilità di sovrapposizione tra segmenti, con minimizzazione della memoria sprecata
- Rilocabilità.

Calcolo degli indirizzi

Ogni volta che l'8086 deve generare un indirizzo da mettere sull'A-bus (*physical address*), esso esegue una operazione di somma tra il contenuto di un registro puntatore oppure di BX (*effective address* o *offset*) ed il contenuto di un registro di segmento (*segment address*).

La somma avviene dopo aver moltiplicato per 16 (shift di 4 posizioni) il contenuto del registro di segmento:



Accesso alla Memoria

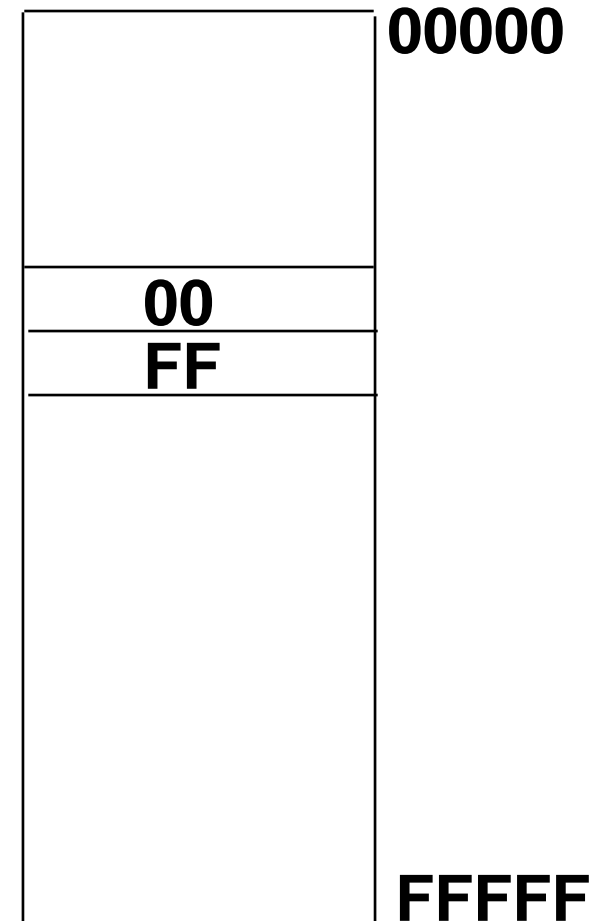
Il modello IA - 16 ha uno spazio di indirizzamento pari a 1 Mbyte.

Il processore è in grado di accedere in un solo passo ad un byte oppure ad una word di memoria, purché questa inizi ad un indirizzo pari. L'accesso a word allineate su indirizzi dispari richiede due cicli di memoria.

Rappresentazione delle word

Si presuppone che i dati memorizzati in una word abbiano il byte meno significativo memorizzato nel byte con indirizzo minore (*little endian*).

Nell'esempio si vede come sia memorizzato il numero FF00.



Parti Riservate della Memoria

Alcune parti della memoria non sono libere, ma *dedicate*, oppure *riservate*.

Ad esempio la memoria dall'indirizzo 00000H allo 003FFH è riservata, in quanto contiene la *Interrupt Vector Table*.

Gli indirizzi da FFFF0H a FFFFBH sono utilizzati per contenere un'istruzione di salto alla routine di caricamento di programma di *bootstrap*.

Le locazioni da FFFFCH a FFFFFH sono invece riservate dati di versione del BIOS.

Interrupt Vector Table	00000 003FF
Liberi	
Reset Bootstrap	FFFF0
Riservati	FFFC