

Architettura 8086/8088

M. Rebaudengo - M. Sonza Reorda

Politecnico di Torino
Dip. di Automatica e Informatica

1

M. Rebaudengo - M. Sonza Reorda

Caratteristiche Generali dell'8086

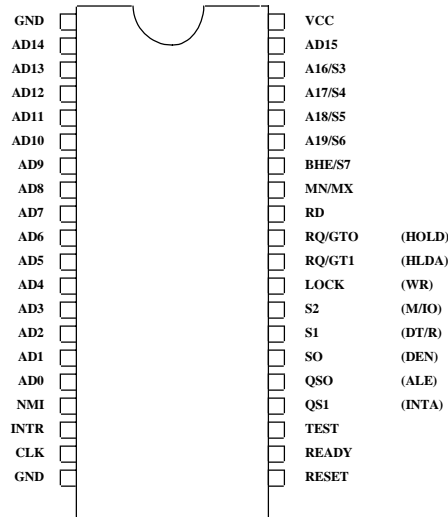
- Usa la tecnologia HMOS.
- Il chip è composto da circa 29,000 transistor ed ha 40 pin.
- A seconda dei modelli può lavorare ad una frequenza di 5 MHz (8086), 8 MHz (8086-2) oppure 10 Mhz (8086-1).
- Richiede un'unica tensione di alimentazione (+5V).
- Spazio di indirizzamento pari a $2^{20}=1\text{Mbyte}$.
- 16 tra i 20 pin di indirizzo fungono anche da pin di dato.
- È orientato alla multiprogrammazione ed al multiprocessing.

2

M. Rebaudengo - M. Sonza Reorda

Piedinatura dell'8086

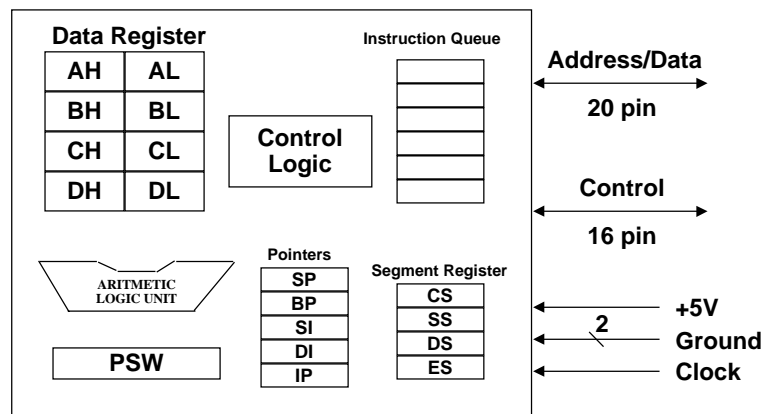
Modo Max (Modo Min)



3

M. Rebaudengo - M. Sonza Reorda

Modello Architeturale



4

M. Rebaudengo - M. Sonza Reorda

Registri

Possono essere suddivisi in 3 gruppi:

- registri di dato
- registri puntatore
- registri di segmento.

5

M. Rebaudengo - M. Sonza Reorda

Registri di Dato

Sono **AX** (*Accumulator Register*), **BX** (*Base Register*), **CX** (*Count Register*) e **DX** (*Data Register*).

Sono utilizzati per memorizzare operandi e risultato delle operazioni.

Possono essere utilizzati come registri da 16 bit oppure come coppie di registri da 8 bit.

BX può anche essere utilizzato nel calcolo di indirizzi.

CX viene anche utilizzato come contatore da talune istruzioni.

DX contiene l'indirizzo di I/O in alcune istruzioni di I/O.

6

M. Rebaudengo - M. Sonza Reorda

Registri Puntatore

Sono IP, SP, BP, SI e DI.

IP (*Instruction Pointer*) contiene il puntatore alla prima istruzione da eseguire. IP non può comparire esplicitamente come operando di una istruzione.

SP (*stack pointer*) contiene il puntatore alla testa dello stack.

BP (*Base Pointer*) viene utilizzato come base per fare accesso all'interno dello stack.

SI (*Source Index*) e **DI** (*Destination Index*) vengono utilizzati come registri indice.

7

M. Rebaudengo - M. Sonza Reorda

Registri di Segmento

Sono CS, DS, ES e SS.

Vengono utilizzati per costruire gli indirizzi fisici con i quali fare accesso in memoria.

Contengono i puntatori all'inizio dei segmenti di codice, di dato, di dato supplementare e di stack, rispettivamente.

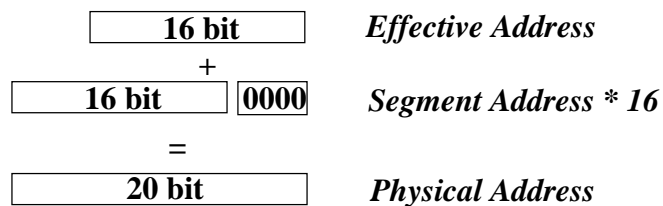
8

M. Rebaudengo - M. Sonza Reorda

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:

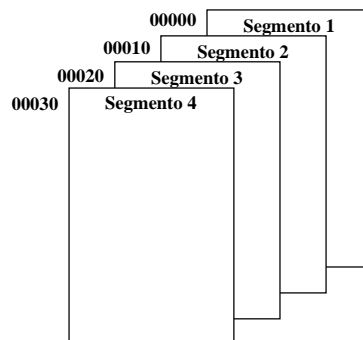


9

M. Rebaudengo - M. Sonza Reorda

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.

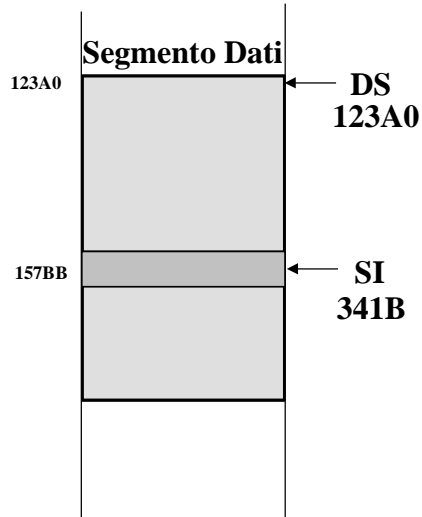


10

M. Rebaudengo - M. Sonza Reorda

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.



11

M. Rebaudengo - M. Sonza Reorda

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

12

M. Rebaudengo - M. Sonza Reorda

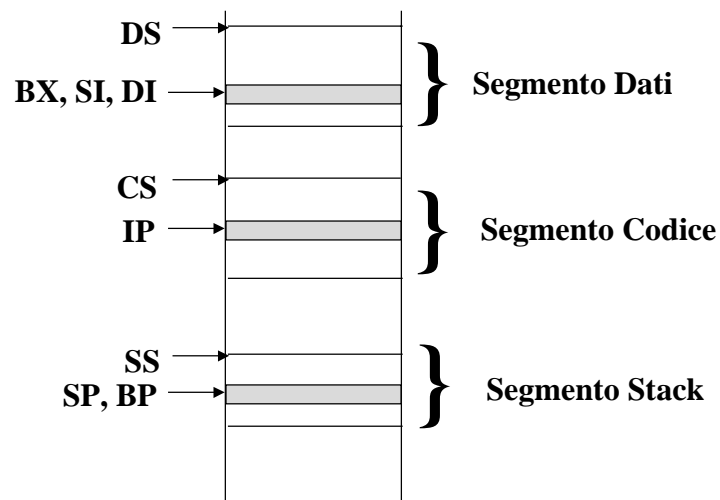
Vantaggi della Segmentazione

- 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.

13

M. Rebaudengo - M. Sonza Reorda

Segmentazione della Memoria



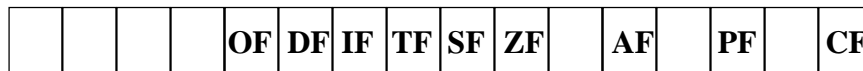
14

M. Rebaudengo - M. Sonza Reorda

Process Status Word (PSW)

È composta da 16 bit, ma solo 9 di questi sono usati. Ogni bit corrisponde ad un flag. I flag si dividono in:

- *flag di condizione*
- *flag di controllo.*



15

M. Rebaudengo - M. Sonza Reorda

Flag di Condizione

Vengono automaticamente scritti al termine di varie operazioni:

- **SF (Sign Flag):** coincide con il MSB del risultato dopo una operazione aritmetica
- **ZF (Zero Flag):** vale 0 se il risultato è nullo, 1 altrimenti
- **PF (Parity Flag):** vale 1 se il numero di 1 negli 8 bit meno significativi del risultato è pari, 0 altrimenti
- **CF (Carry Flag):** dopo le operazioni aritmetiche vale 1 se c'è stato *riporto* (somma) o *prestito* (sottrazione); altre istruzioni ne fanno un uso particolare
- **AF (Auxiliary Carry Flag):** usato nell'aritmetica BCD; vale 1 se c'è stato *riporto* (somma) o *prestito* (sottrazione) dal bit 3
- **OF (Overflow Flag):** vale 1 se l'ultima istruzione ha prodotto un *overflow*.

16

M. Rebaudengo - M. Sonza Reorda

Flag di Controllo

Possono venire scritti e manipolati da apposite istruzioni, e servono a regolare il funzionamento di talune funzioni del processore:

- **DF (*Direction Flag*):** utilizzato dalle istruzioni per la manipolazione delle stringhe; se vale 0 le stringhe vengono manipolate partendo dai caratteri all'indirizzo minore, se vale 1 a partire dall'indirizzo maggiore
- **IF (*Interrupt Flag*):** se vale 1, i segnali di Interrupt mascherabili vengono percepiti dalla CPU, altrimenti questi vengono ignorati
- **TF (*Trap Flag*):** se vale 1, viene eseguita una *trap* al termine di ogni istruzione.

17

M. Rebaudengo - M. Sonza Reorda

Accesso alla Memoria

L'8086/8088 ha uno spazio di indirizzamento pari a 1 Mbyte.

L'8086 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.

L'8088 richiede invece un ciclo per ogni byte cui si fa accesso in memoria.

18

M. Rebaudengo - M. Sonza Reorda

Accesso alla memoria

(segue)

Fisicamente la memoria è organizzata come due banche da 512Kbyte:

- il banco dispari ($D_{15}-D_8$)
- il banco pari (D_7-D_0)

I due banche sono indirizzate in parallelo dalle linee di indirizzo $A_{19}-A_1$.

I dati con indirizzo pari sono trasferiti sui pin D_7-D_0 mentre i dati ad indirizzi dispari sono trasferiti sui pin $D_{15}-D_8$.

19

M. Rebaudengo - M. Sonza Reorda

Accesso alla memoria

(segue)

Il processore fornisce due segnali di *enable* (BHE e A_0) per gestire l'accesso alla memoria.

<i>BHE</i>	A_0	<i>Operazioni in memoria</i>
0	0	Word Intera
0	1	Byte alto (dall'indirizzo dispari)
1	0	Byte basso (dall'indirizzo pari)
1	1	Nessuna

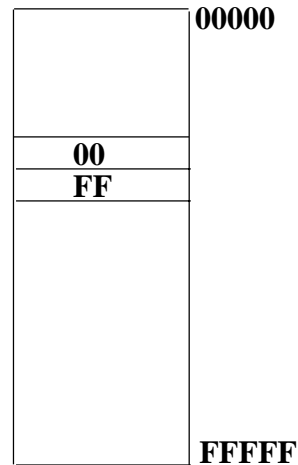
20

M. Rebaudengo - M. Sonza Reorda

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.



21

M. Rebaudengo - M. Sonza Reorda

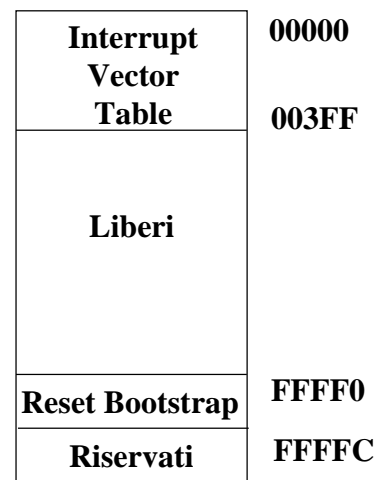
Parti riservate della Memoria

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 per usi futuri.



22

M. Rebaudengo - M. Sonza Reorda

Pipeline

Un meccanismo di pipeline *ideale* è costituito da una serie di stadi che eseguono un compito specifico in un tempo di esecuzione *costante ed uguale* per tutti gli stadi.

Una catena di montaggio di una fabbrica automobilistica funziona con il meccanismo di pipeline. L'assemblaggio di una automobile è caratterizzata da diversi stadi da eseguire in un ordine fisso e caratterizzato da un determinato tempo di esecuzione (ad es. creazione scocca, montaggio del motore, sistemazione degli interni e verniciatura).

I vari stadi produttivi sono posti in cascata e lavorano in modo che l'uscita di uno stadio coincida con l'ingresso dello stadio seguente.

23

M. Rebaudengo - M. Sonza Reorda

Pipeline

(segue)

A regime, ogni stadio è caratterizzato da una coda di prodotti in attesa di essere processati.

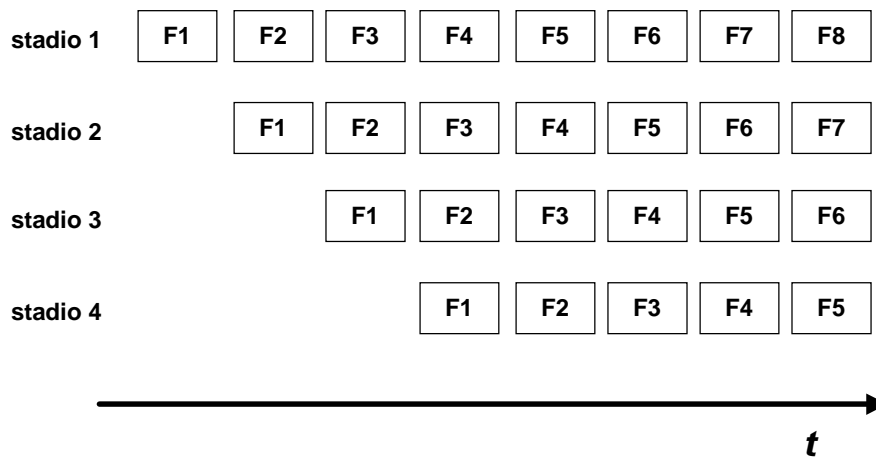
Se i tempi di esecuzione delle varie fasi sono uguali il sistema funziona in maniera ideale: ogni stadio è sempre attivo e la sua coda è sempre vuota; al termine di un'operazione il prodotto passa allo stadio a valle e l'unità produttiva è pronta a processare il prodotto che gli arriva dallo stadio a monte.

In una pipeline ideale il sistema globale riduce il tempo medio di fabbricazione di un prodotto di un fattore pari al numero di stadi della pipeline.

24

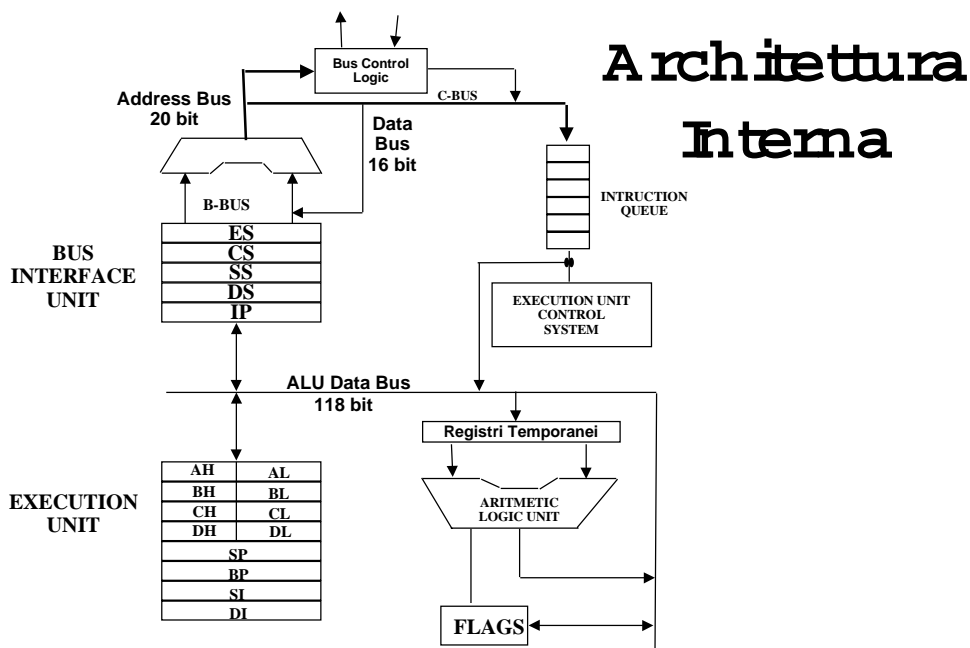
M. Rebaudengo - M. Sonza Reorda

Pipeline ideale



25

M. Rebaudengo - M. Sonza Reorda



26

M. Rebaudengo - M. Sonza Reorda

Execution Unit (EU)

Provvede alla *decodifica* ed alla *esecuzione* delle istruzioni.

Riceve byte per byte le istruzioni dalla BIU, le decodifica, genera gli indirizzi degli operandi (se necessario), li passa alla BIU; una volta ricevuti tutti gli operandi esegue l'istruzione, testa ed aggiorna i flag.

27

M. Rebaudengo - M. Sonza Reorda

Bus Interface Unit (BIU)

Gestisce tutte le operazioni da e per l'esterno:

- Fetch delle istruzioni
- Lettura e scrittura operandi e risultati di istruzioni
- Generazione degli indirizzi
- Accodamento delle istruzioni
- La BIU lavora in parallelo con la EU.

28

M. Rebaudengo - M. Sonza Reorda

Coda delle Istruzioni

La BIU gestisce una struttura FIFO di dimensioni pari a 6 byte (4 nell'8088) in cui vengono accumulate le istruzioni che si prevede dovranno essere eseguite.

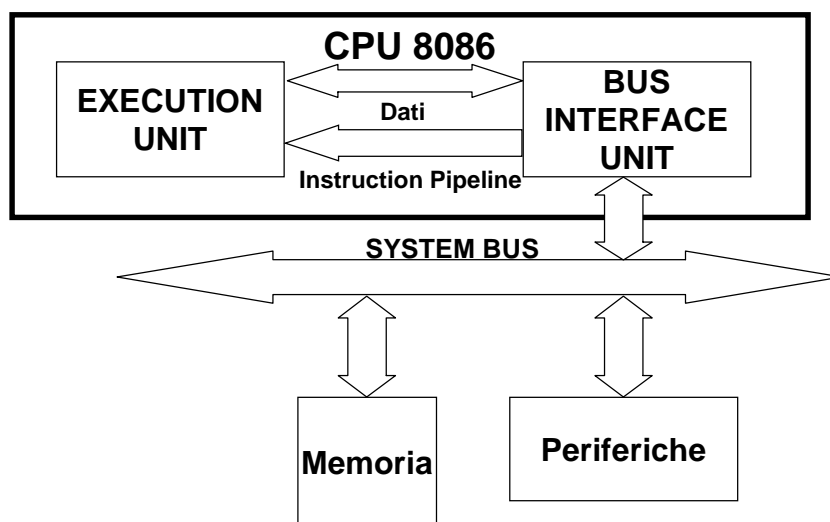
La *Coda delle Istruzioni* viene caricata dalla BIU ogni qual volta vi è una word libera, ed il bus è libero; in tal caso viene letta dalla memoria la word successiva nel Code Segment.

Se viene eseguita una istruzione di salto, la *Coda delle Istruzioni* viene azzerata, e la EU deve attendere il tempo necessario per il fetch di una nuova istruzione prima di poter lavorare.

29

M. Rebaudengo - M. Sonza Reorda

Architettura Interna



30

M. Rebaudengo - M. Sonza Reorda

Pipeline nell'8086

Nel processore 8086 il meccanismo di esecuzione di un'istruzione è organizzato come una pipeline a due stadi:

- la fase di *fetch* (eseguita dalla BIU)
- la fase di *esecuzione* dell'istruzione (eseguita dalla EU).

La pipeline dell'8086 non ha un comportamento ideale perché:

- i tempi di esecuzione dei due stadi non sono fissi, ma funzione del tipo di istruzione;
- non tutte le istruzioni in uscita dalla BIU vengono processate dall'EU.

Coda di istruzioni

Per ottimizzare il processamento della istruzioni da parte della EU, la BIU esegue l'operazione di *prefetch*, ossia carica preventivamente word di istruzioni dalla memoria.

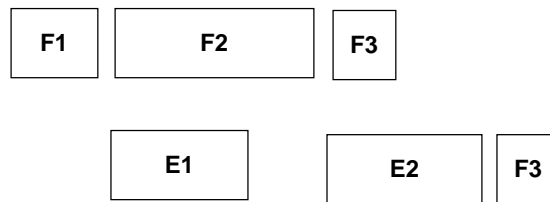
Tali word vengono memorizzate nella coda di istruzioni.

Pipeline nell'8086

(segue)



processamento sequenziale



processamento in pipeline

33

M. Rebaudengo - M. Sonza Reorda

Gestione della memoria

Durante il *prefetch* delle istruzioni, ogniqualvolta vi siano almeno due byte liberi nella coda delle istruzioni, la BIU preleva dalla memoria una parola allineata all'indirizzo pari.

L'unica eccezione si ha nel caso di salto ad indirizzo dispari.

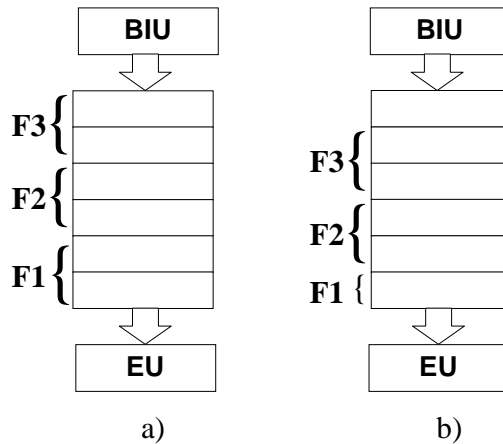
Quando ciò accade, la BIU carica nella coda un solo byte e si allinea all'indirizzo pari successivo; da qui riprende a prelevare word ad indirizzi pari.

34

M. Rebaudengo - M. Sonza Reorda

Gestione della memoria

(segue)



35

M. Rebaudengo - M. Sonza Reorda

Lo Stack

L'8086/8088 prevede alcune strutture e meccanismi hardware per la gestione di uno *stack*.

Lo stack corrisponde al segmento di memoria la cui testa è puntata da SS. Il *top* dello stack (locazione riempita per ultima) è puntato da SP.

Lo stack cresce dalle locazioni di memoria con indirizzo maggiore verso quelle ad indirizzo minore.

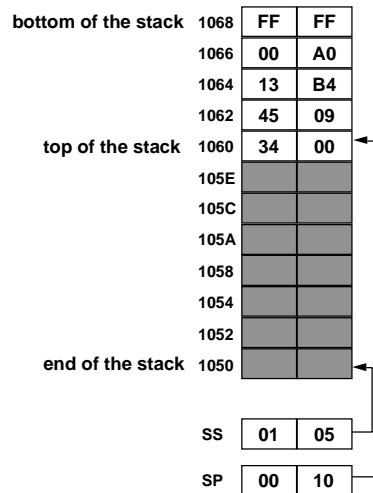
Ogni operazione di PUSH decrementa di 2 unità SP e scrive una word nella locazione da questo puntata.

Ogni operazione di POP estrae una word dalla locazione puntata da SP, e successivamente incrementa SP di 2 unità.

36

M. Rebaudengo - M. Sonza Reorda

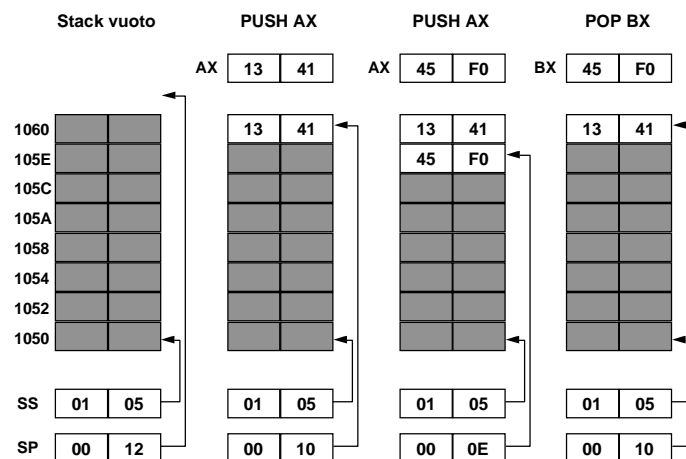
Esem pio di stack



37

M. Rebaudengo - M. Sonza Reorda

Operazioni sul stack



38

M. Rebaudengo - M. Sonza Reorda

Uso dello Stack

Oltre che a seguito delle istruzioni che esplicitamente lo manipolano (PUSH e POP), lo stack viene modificato in altri due casi:

- all'atto della chiamata (CALL) o del ritorno (RET) da una procedura: nel primo caso viene automaticamente eseguita la PUSH dell'IP, nel secondo caso l'analoga POP;
- all'atto dell'attivazione di un interrupt, e del ritorno dalla corrispondente routine di servizio (IRET); nel primo caso vengono eseguite automaticamente le PUSH del *program counter* (registri CS e IP) e della PSW, nel secondo caso vengono eseguite le corrispondenti POP.

I/O

L'accesso alle periferiche avviene spesso attraverso speciali locazioni associate ad un certo indirizzo.

L'accesso a tali locazioni può avvenire nell'8086/8088 sia in modo *memory mapped* sia in *isolated I/O*. Nel primo caso l'accesso alla periferica avviene attraverso una normale istruzione, nel secondo attraverso speciali istruzioni di I/O.

Lo spazio di indirizzamento dell'I/O è pari al più a 64K.

8088

L'8088 differisce dall'8086 unicamente in quanto il suo bus dati esterno ha ampiezza pari ad 8 anzichè 16 bit.

Dal punto di vista interno le uniche differenze si hanno nella BIU (interfaccia verso il bus e coda delle istruzioni).

L'8088 fu realizzato per 2 ragioni principali:

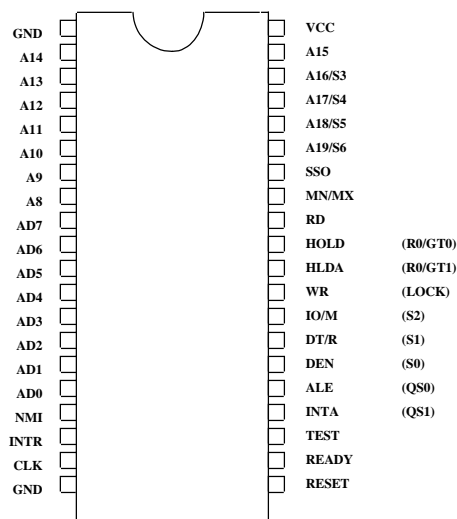
- per permettere di riutilizzare, almeno parzialmente, l'HW realizzato per sistemi 8080/8085
- per rendere utilizzabili le periferiche a 8 bit prodotte per l'8080/8085.

41

M. Rebaudengo - M. Sonza Reorda

Piedinatura 8088

Modo Min (Modo Max)



42

M. Rebaudengo - M. Sonza Reorda