Architettura degli Elaboratori

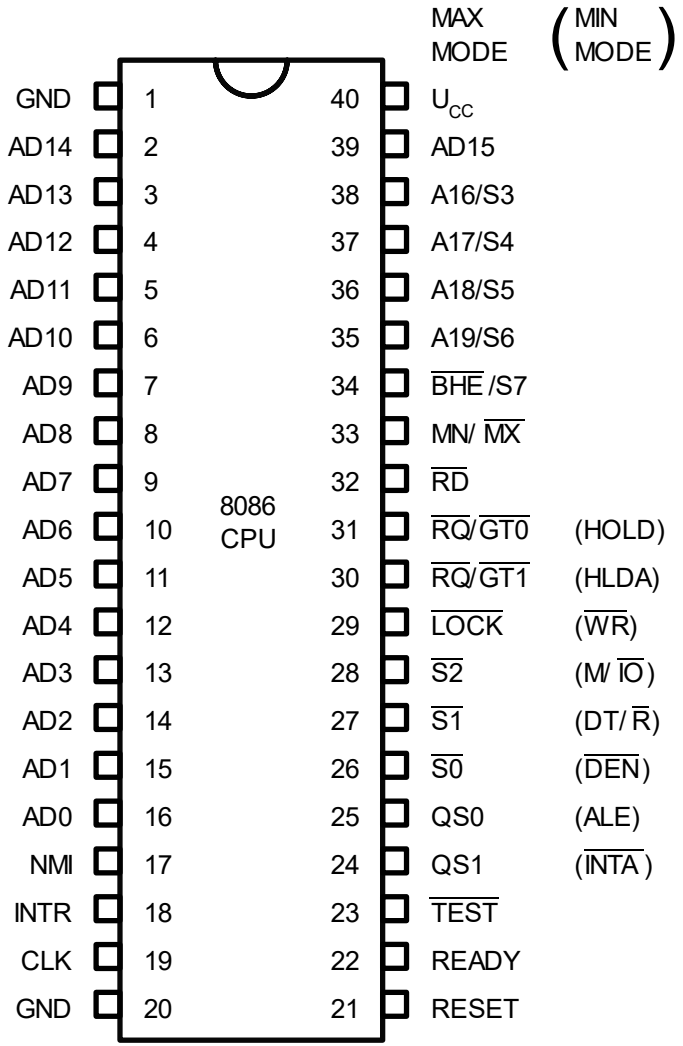
Lezioni 19-20 – Architettura x86 e Linguaggio Assembly

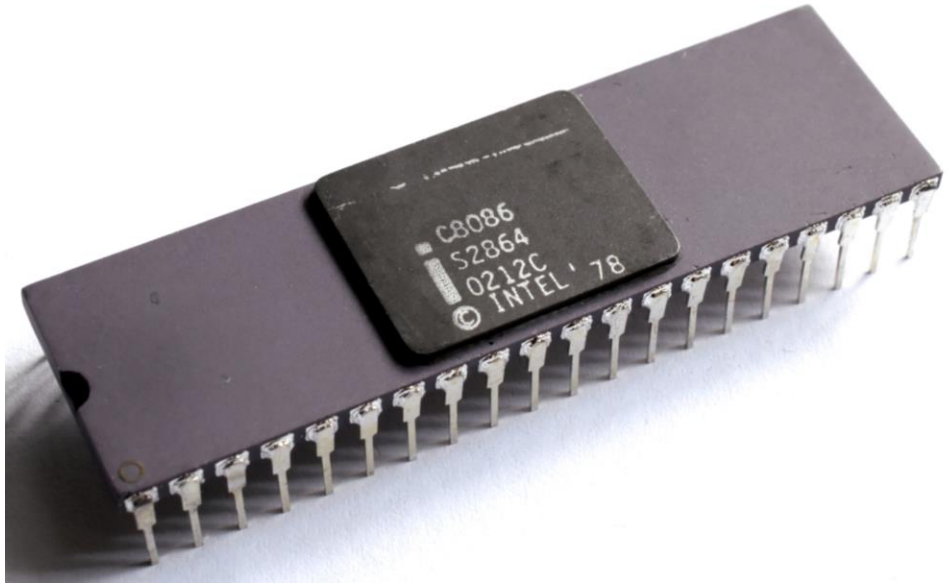
Giuseppe Cota
Dipartimento di Scienze Matematiche Fisiche e Informatiche
Università degli Studi di Parma

Processore 8086/8088

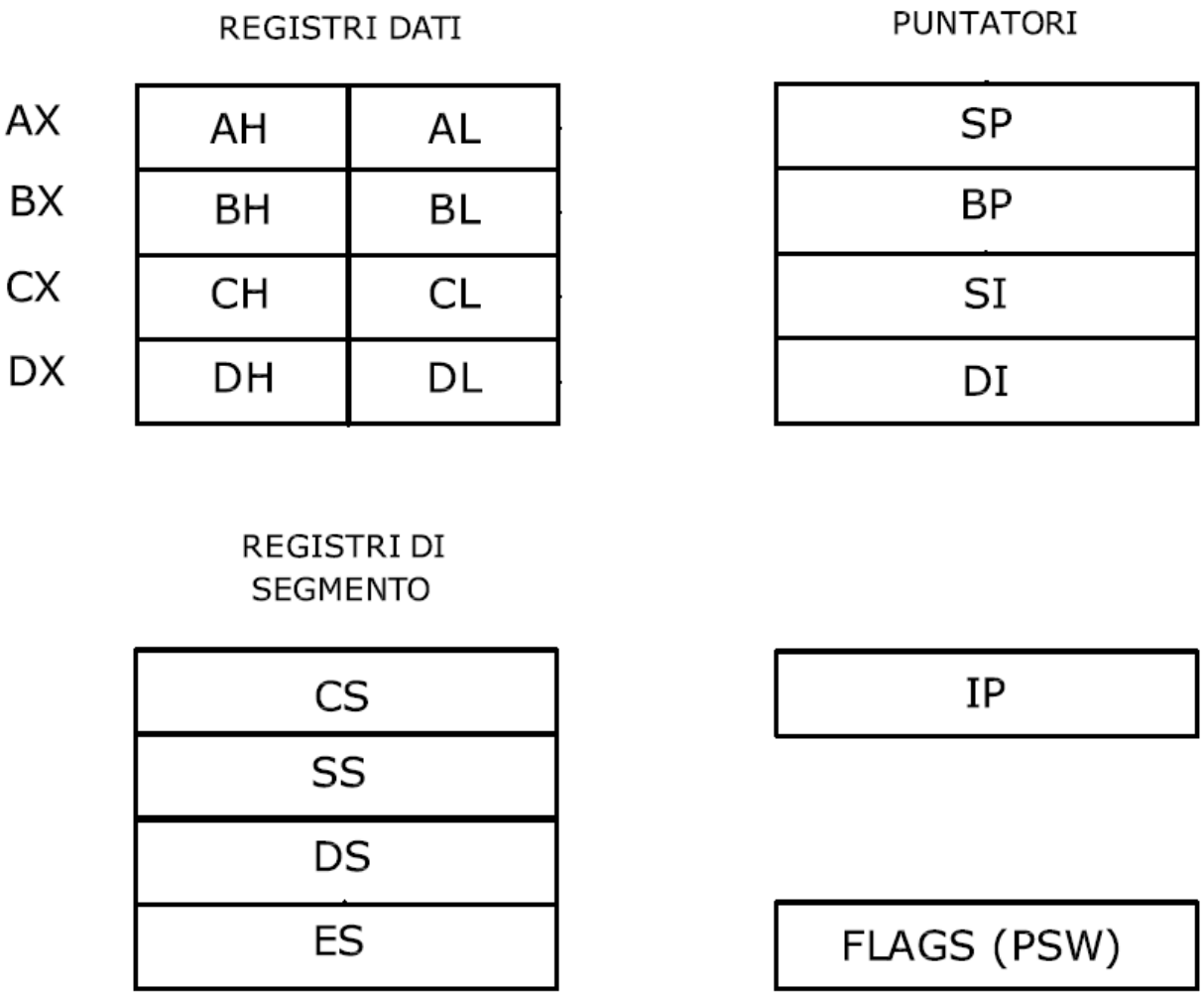
- Microprocessore Intel di terza generazione.
- 8086 rilasciato nel 1978 l'anno dopo viene rilasciato l'8088
- Bus indirizzi: 20 bit
  - Spazio di indirizzamento 2^{20} byte = 1 MiB
    - Da 00000_H a $FFFFF_H$
- Bus dati
  - 8 bit per 8088
  - 16 bit per 8086
- Con l'8086 nasce la famiglia di processori x86.
  - Quasi tutti i processori Intel (anche quelli odierni) mantengono la retrocompatibilità.
- Sebbene non sia più utilizzato, l'architettura e il repertorio delle istruzioni sono alla base dei processori successivi.

8086


MAX MODE
(MIN MODE)
GND 1
AD14 2
AD13 3
AD12 4
AD11 5
AD10 6
AD9 7
AD8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
GND 20
8086 CPU
40 U_{CC}
39 AD15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 $\overline{BHE}$/S7
33 MN/$\overline{MX}$
32 $\overline{RD}$
31 $\overline{RQ}$/$\overline{GT0}$ (HOLD)
30 $\overline{RQ}$/$\overline{GT1}$ (HLDA)
29 $\overline{LOCK}$ ($\overline{WR}$)
28 $\overline{S2}$ (M/$\overline{IO}$)
27 $\overline{S1}$ (DT/$\overline{R}$)
26 $\overline{S0}$ ($\overline{DEN}$)
25 QS0 (ALE)
24 QS1 ($\overline{INTA}$)
23 $\overline{TEST}$
22 READY
21 RESET


C8086
S2864
0212C
INTEL '78

Registri dell'8086



Registri dell'8086

- Tutti i registri sono di 16 bit
- 8 registri di uso generale:
  - 4 per i dati: AX, BX, CX e DX
  - 4 usati come puntatori o registri indice: SP, BP, SI e DI
- 4 registri di segmento: CS, DS, SS e ES
- 1 registro usato come puntatore di istruzione all'interno di un segmento: IP
  - CS insieme ad IP (CS:IP) corrisponde al Program Counter (PC)
- 1 registro di stato chiamato registro FLAGS
  - Solo 9 bit sono utilizzati
  - Corrisponde al Processor Status Word (PSW)

I registri dati

- Utilizzabili come registri a 16 bit
  - vengono identificati con i termini AX, BX, CX e DX.
- Utilizzabili come registri a 8 bit
  - AH, BH, CH e DH identificano i byte più significativi (AH = AHigh)
  - AL, BL, CL e DL identificano i byte meno significativi (AL = Alow)
- Nella maggior parte dei casi possono essere usati indistintamente, ma hanno anche ruoli specifici:
  - **AX** (*accumulator register)*, serve per istruzioni I/O, per stringhe e operazioni matematiche.
  - **BX** (*base register*), può essere usato per il calcolo degli indirizzi.
  - **CX** (*count register*), usato come contatore in certe istruzioni.
  - **DX** (*data register*), è richiesto da alcune istruzioni di I/O e per operazioni matematiche con grandi valori (uso di DX e AX).

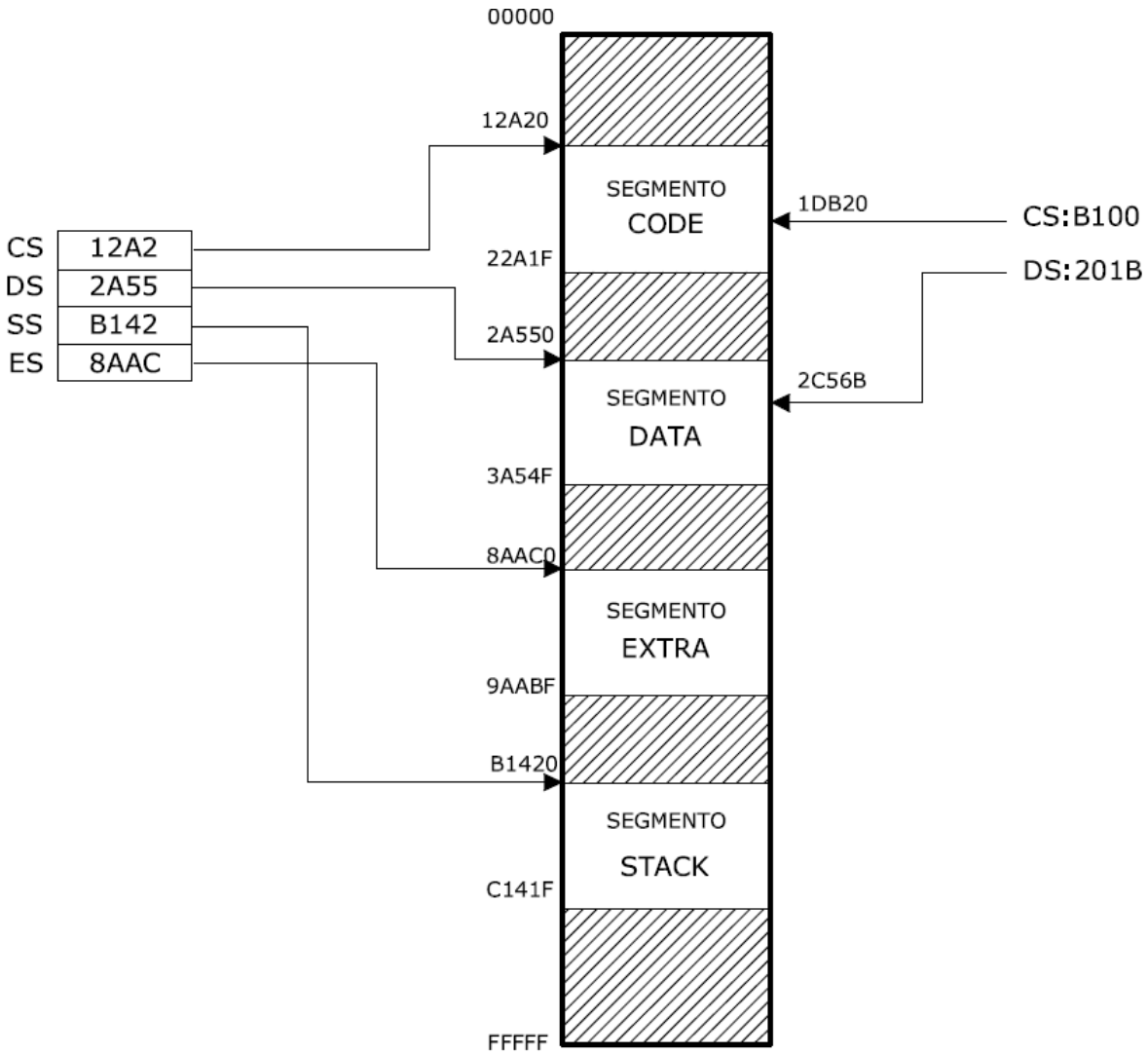
I registri puntatore e indice

- Registri che contengono gli *scostamenti (offset)* all'interno dei segmenti.
- **SP** (*stack pointer*): puntatore alla cima dello stack
- **BP** (*base pointer*): puntatore entro lo stack, ma può essere impiegato come generico registro indice.
- **SI** (*source index*): registro indice di uso generico. Il come deriva dal fatto che certe istruzioni con le stringhe richiedono che la stringa sorgente sia individuata tramite SI.
- **DI** (*destination index*): registro indice di uso generico. Il come deriva dal fatto che certe istruzioni con le stringhe richiedono che la stringa di destinazione sia individuata tramite DI.

Registri di segmento

- Un aspetto caratteristico della CPU 8086 è che lo spazio di memoria è diviso in segmenti.
- Ogni segmento è un'unità logica di memoria indirizzabile separatamente e indipendentemente da altri segmenti.
  - Inizia a un indirizzo di memoria multiplo di 15
  - estensione massima $2^{16} = 64$ KiB
- I registri di segmento tengono traccia della posizione in memoria dei segmenti attualmente in uso.
- **CS** (*code segment*): identifica il segmento di codice corrente.
- **DS** (*data segment*): identifica il segmento dei dati corrente.
- **SS** (*stack segment*): identifica il segmento di stack corrente.
- **ES** (*extra segment*): identifica il segmento extra corrente.
  - Il segmento extra in genere viene utilizzato per memorizzare i dati.

Segmentazione della memoria



Registro Instruction Pointer IP

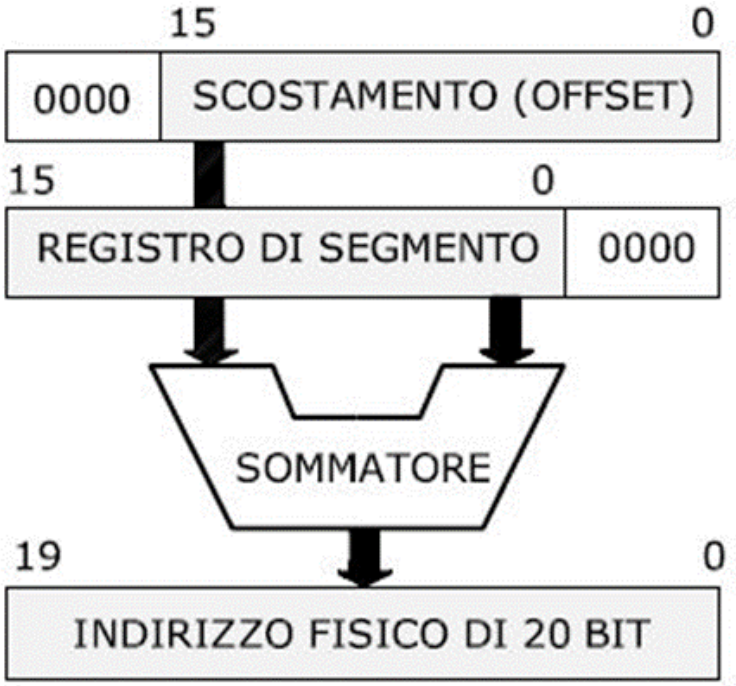
- I codici di istruzione vengono sempre prelevati dal segmento di codice.
- È necessario un registro che contenga l'offset (lo scostamento) della prossima istruzione da eseguire rispetto al segmento di codice corrente.
  - IP contiene l'offset
  - CS contiene l'indirizzo del segmento di codice corrente
- Sono chiamati *indirizzi logici*, gli indirizzi nella forma

Indirizzo segmento SEG : OFFSET **Scostamento**

- L'indirizzo logico di un'istruzione sarà CS:IP

Generazione dell'indirizzo fisico

- Indirizzo logico composto di due parti SEG:OFFSET (ambedue a 16 bit)
- Indirizzo fisico (presentato all'esterno) su 20 bit
  - L'*indirizzo fisico* è ottenuto facendo SEG $*$ 16 + OFFSET



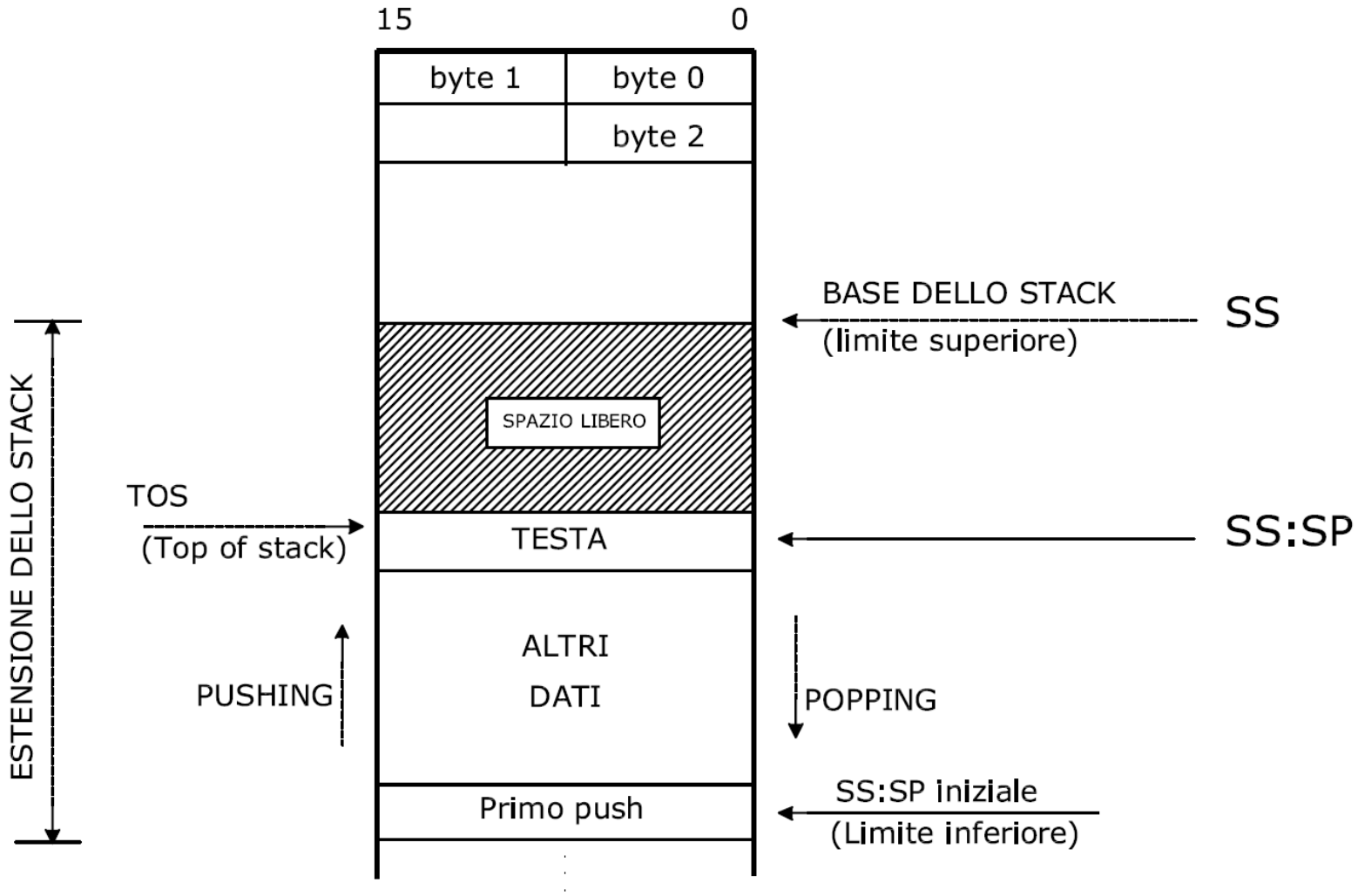
Generazione dell'indirizzo fisico
Esempi

- Uso la notazione esadecimale ($16 = 10_H$)
- $1A_H{:}0H = 1A_H \cdot 10_H + 0_H$
- $1A_H{:}5_H = 1A_H \cdot 10_H + 0_H$

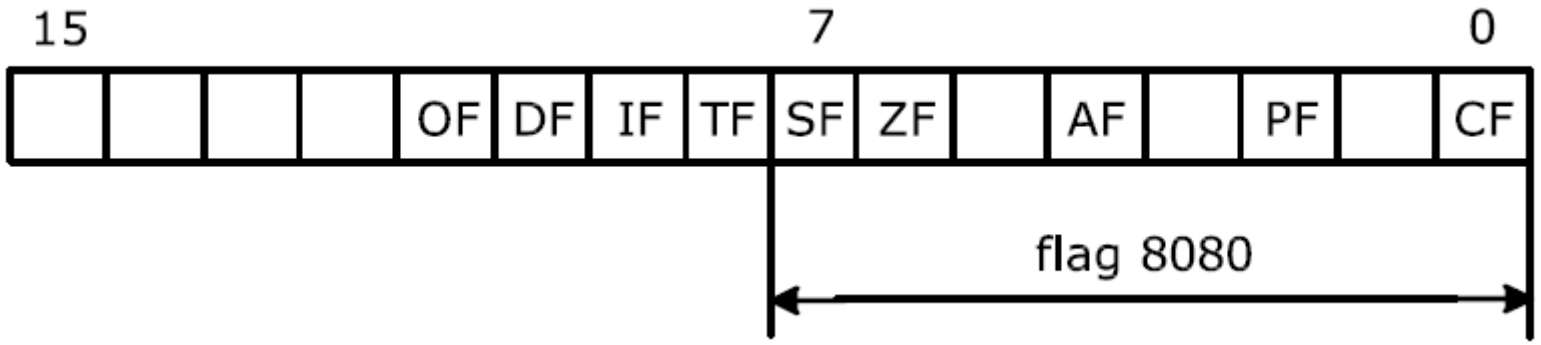
Stack

- Un solo stack è quello corrente:
  - SS contiene l'indirizzo del segmento stack
  - SP contiene l'offset dello stack
  - SS:SP punta alla testa dello stack
- **Lo stack cresce andando dagli indirizzi più alti a quelli più bassi.**
- Operazione di PUSH:
  - SP ← SP – 2
  - Scrittura di 2 byte nel nuova testa dello stack
- Operazione di POP:
  - Lettura di 2 byte dalla testa dello stack
  - SP ← SP + 2

Stack


15
0
byte 1
byte 0
byte 2
BASE DELLO STACK
(limite superiore)
SS
SPAZIO LIBERO
TOS
(Top of stack)
TESTA
SS:SP
ESTENSIONE DELLO STACK
ALTRI
DATI
PUSHING
POPPING
Primo push
SS:SP iniziale
(Limite inferiore)

Registro FLAGS



- Registro di stato che contiene 9 indicatori di 1 bit chiamati *flag*
  - 6 registrano informazioni sullo stato del processore (flag di stato) e 3 controllano le operazioni del processore (flag di controllo)

Registro FLAGS

- Flag di stato
  - CF (Carry Flag): vale 1 se un'istruzione ha generato un riporto.
  - AF (Auxiliary Flag): vale 1 se dopo un'addizione c'è stato un riporto dal nibble meno significativo e se dopo una sottrazione c'è stato un prestito al nibble meno significativo.
  - OF (Overflow Flag): vale 1 se un'istruzione ha generato un overflow.
  - SF (Sign Flag): vale 1 se il risultato di un'istruzione è negativo.
  - ZF (Zero Flag): vale 1 se il risultato di un'istruzione è 0.
  - PF (Parity Flag): vale 1 il risultato di un'istruzione ha un numero pari bit a 1. Considera solo il byte meno significativo.
- Flag di controllo
  - DF (Direction Flag): serve a controllare la direzione di manipolazione delle stringhe da parte di alcune istruzioni a esse riservate.
  - IF (Interrupt Flag): abilita o disabilita le interruzioni esterne.
  - TF (Trap Flag): permette l'esecuzione di un'istruzione per volta (single step); usato per il debugging

Fase di avvio iniziale

- Quando la CPU viene avviata a CS viene assegnato il valore FFFF mentre a IP viene assegnato il valore 0. Conseguentemente la prima istruzione eseguita è quella alla locazione FFFF0.
- Nella parte alta dello spazio degli indirizzi viene usata una memoria di tipo ROM, allo scopo di garantire che quando la CPU viene alimentata entri in scena il firmware di inizializzazione del sistema:
  - il Basic Input/Output System (BIOS).

Indirizzamento registro-registro e uso degli operandi immediati

- Possiamo leggere e scrivere valori da e verso registri.
  Esempio:

```
MOV AH, BL    ; AH ← BL
```

- Possiamo anche scrivere sui registri usando gli operandi immediati, ossia dando direttamente un valore numerico da scrivere in un registro.
  Esempio:

```
MOV AX, 358   ; AX ← 358
```

Indirizzamento delle porte I/O

- Per leggere scrivere da periferiche possiamo usare i comandi IN e OUT
- IN permette di leggere da una porta ammette due operandi
  - Il primo operando è il registro di destinazione e il dato va inserito in AL (8 bit) o AX (16 bit),
  - Il secondo operando è il numero della porta. Può essere un numero tra 0 e 255 oppure il valore contenuto in DX.
  - Esempio

```
IN AL, 7          ; leggi da porta 7 e inserisci in AL
IN AX, DX ; leggi da porta indicata in DX e inserisci in AX
```

- OUT permette di scrivere su una porta e ammette due operandi
  - Primo operando è la porta: un numero tra 0 e 255 oppure il valore contenuto in DX.
  - Secondo operando: AX o AL (dove deve risiedere l'output da inviare alla porta)
  - Esempio:

```
OUT DX, AL ; scrivi nella porta indicata in DX il valore in AL
OUT 4, AX  ; scrivi nella porta 4 il valore in AX
```

Metodi di indirizzamento alla memoria

- Oltre che ai registri e alle porte è possibile accedere alla memoria e questo può essere fatto in **tanti modi**.
- Questi possono essere raggruppati in 3 categorie:
- **Indirizzamento diretto:** si specifica l'indirizzo di memoria tramite un valore numerico detto scostamento.
  Esempio: `MOV AX, VAR ; AX ← M[Offset(VAR)]`
- **Indirizzamento indiretto tramite registro base:** si specifica l'indirizzo di memoria tramite il valore contenuto in uno tra i registri base BX e BP.
  Esempio: `MOV [BX], AX ; M[BX] ← AX`
- **Indirizzamento indiretto tramite registro indice**: si specifica l'indirizzo di memoria tramite il valore contenuto in uno tra i registri di indice SI o DI
  Esempio: `MOV AX, [SI] ; AX ← M[SI]`

Metodi di indirizzamento alla memoria

- È possibile combinare le tre modalità ottenendo tutte le possibili combinazioni.
- Le combinazioni valide possono essere ottenute sfruttando la seguente tabella (ognuno dei tre elementi può non essere presente)

Discostamento (valore diretto)	**+**	**BX**	**+**	**SI**
		BP		**DI**

Esempi:

```
MOV AX, VAR[BX]     ; AX ← M[Offset(VAR)+BX]
MOV [BX][DI], AX    ; M[BX+DI] ← AX
MOV CX, VAR[BX][SI] ; CX ← M[Offset(VAR)+BX+SI]
```

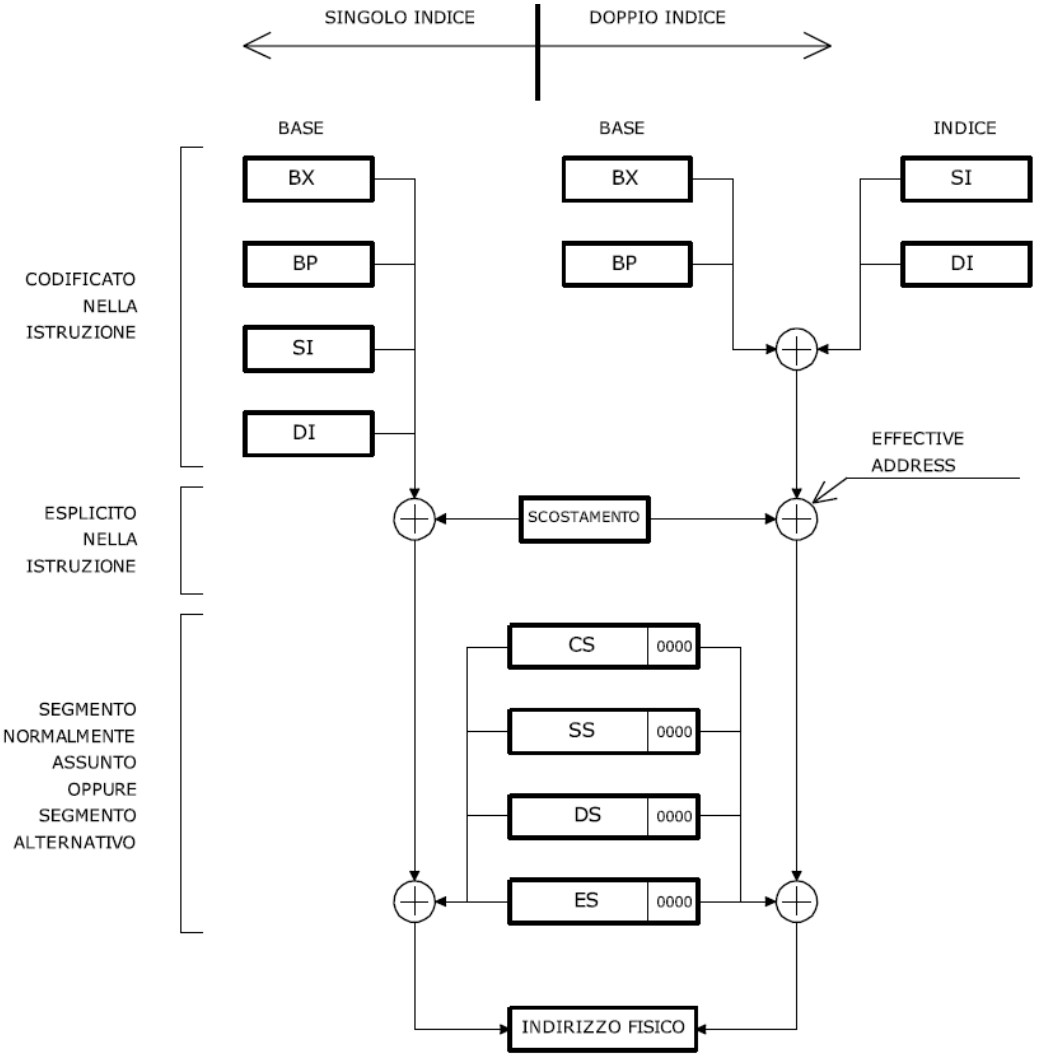
Metodi di indirizzamento alla memoria

- Queste modalità di indirizzamento si riferiscono sempre e solo al calcolo dell'offset (scostamento) finale.
- L'indirizzo logico è sempre costituito dalla coppia SEG:OFFSET
- **Inoltre**, se nella modalità di indirizzamento compare BP, allora il segmento di riferimento sarà SS (segmento stack), altrimenti il riferimento di default è sempre DS (segmento dati).
- Posso tuttavia esplicitare il segmento nell'istruzione.

Esempi

```
MOV AX, VAR[BX]      ; AX ← M[DS:Offset(VAR)+BX]
MOV [BX][DI], AX     ; M[DS:BX+DI] ← AX
MOV CX, VAR[BX][SI] ; CX ← M[DS:Offset(VAR)+BX+SI]
MOV AX, VAR[BP]      ; AX ← M[SS:Offset(VAR)+BP]
MOV BX, ES:VAR       ; BX ← M[ES:Offset(VAR)]
```

Schema metodo di calcolo dell'indirizzo di memoria



Linguaggio Assembly

Domande?

Riferimenti principali

- Appendice C di ***Calcolatori elettronici. Architettura e Organizzazione,*** Giacomo Bucci. McGraw-Hill Education, 2017. http://highered.mheducation.com/sites/dl/free/8838675465/1098336/AppC.pdf (download gratuito)
- Approfondimenti:
  - Appendice G di ***Calcolatori elettronici. Architettura e Organizzazione,*** Giacomo Bucci. McGraw-Hill Education, 2017. http://highered.mheducation.com/sites/dl/free/8838675465/1098336/AppG.pdf (download gratuito)
  - Instruction set completo dell'8086: https://jbwyatt.com/253/emu/8086_instruction_set.html
  - Datasheet e manuali per 8086