Corso di Architettura degli Elaboratori
Modulo di Assembly

MODELLI DI MEMORIA e CALL

Bruno Iafelice

University of Bologna

iafelice at cs(dot)unibo(dot)it

Argomenti

- Modelli di memoria per 8088
- Chiamata di subroutine e Protocollo chiamante/ chiamato
- Chiamata di subroutine di sistema

MODELLI DI MEMORIA

Gli assemblatori per l'8088 supportano in generale 6 modelli di memoria con complessità crescente:

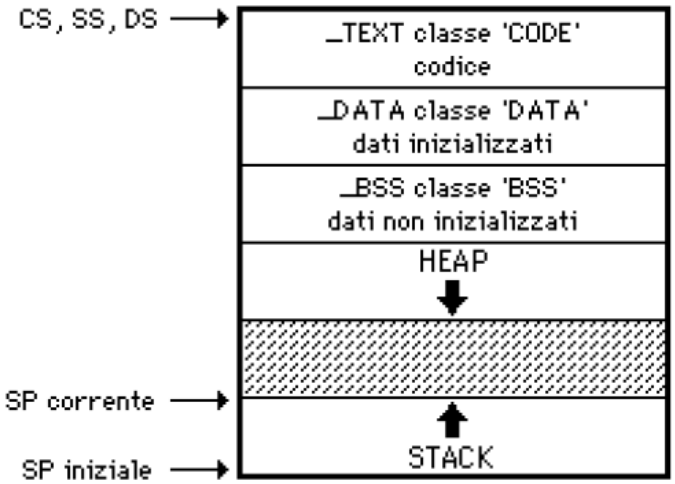
- **tiny**: 1 segmento per codice, dati, stack e heap
- **small**: 1 segmento per codice; 1 segmento per dati, stack e heap
- **compact**: 1 segmento per codice; 1 seg. Dati; 1 seg. Stack; più segmenti per heap
- **medium**
- **large**
- **Huge**

] più segmenti per codice, Più segmenti per dati

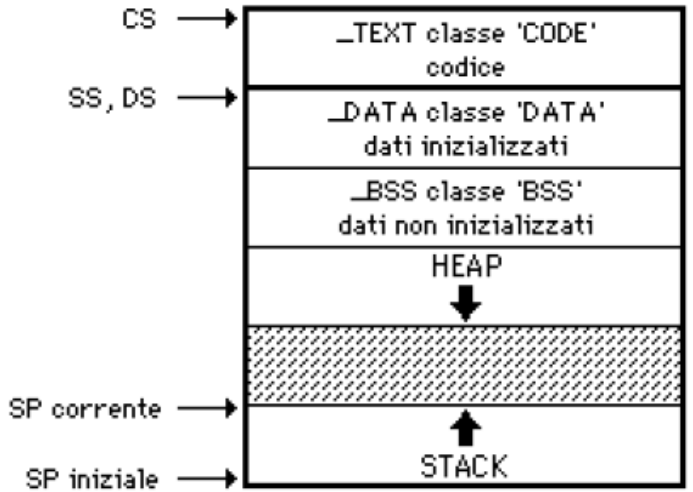
Il programmatore sceglie il modello in base alle dimensioni dei Dati e del Codice.

L'assemblatore allegato al testo di Tanenbaum si avvale del SOLO modello SMALL: 1 segmento codice , 1 dati e stack (SS=DS=ES).

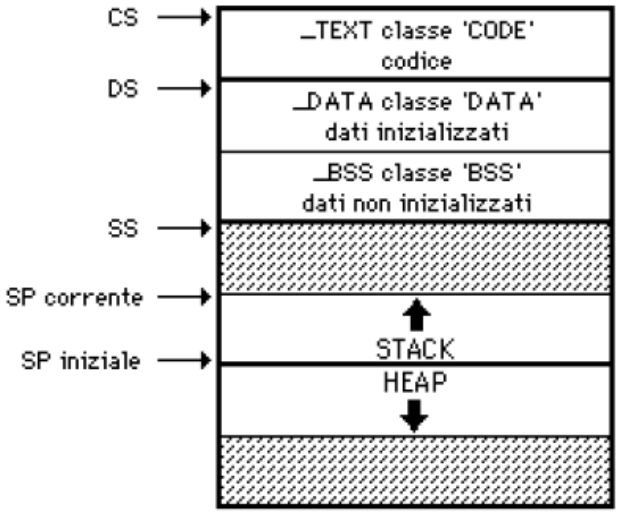
Modello di memoria TINY



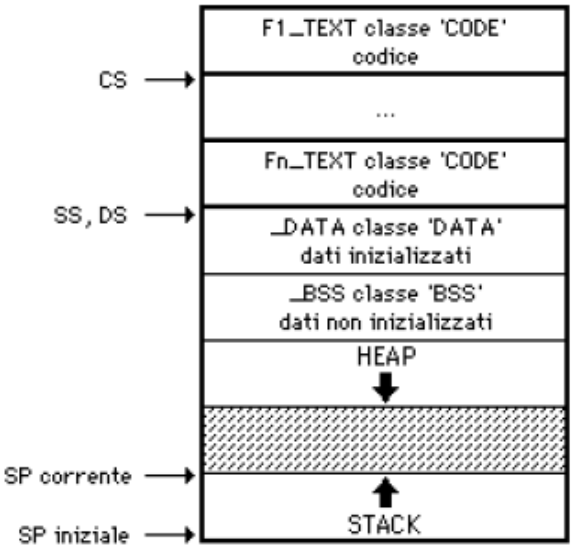
Modello di memoria SMALL



Modello di memoria COMPACT

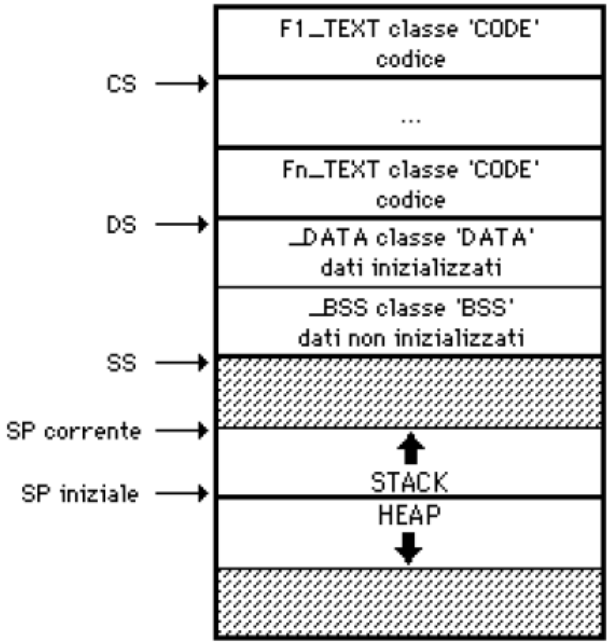


Modello di memoria MEDIUM

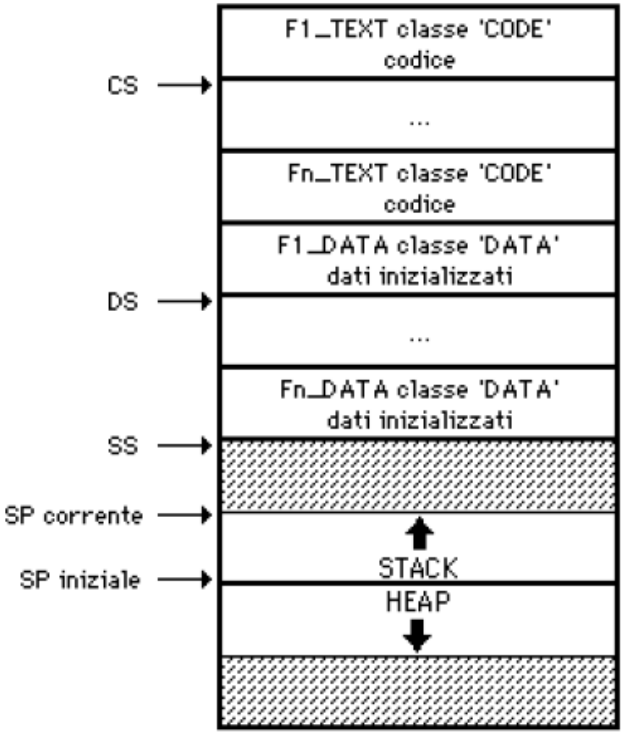


Cortesia Antonio Corradi

Modello di memoria LARGE



Modello di memoria HUGE



Riepilogo

Modello di memoria	Riferimenti al codice	Riferimenti ai dati	Dimensione applicazione
Tiny	near	near	codice + dati ≤ 64k
Small	near	near	codice ≤ 64k + dati ≤ 64k
Compact	near	far	codice ≤ 64k + dati ≤ 1M
Medium	far	near	codice ≤ 1M + dati ≤ 64k
Large	far	far	codice + dati ≤ 1M
Huge	far	far	codice + dati ≤ 1M

Cortesia Antonio Corradi

SOTTOPROCEDURE: Trasferimento del controllo

`CALL proceduraChiamata`

- Trasferisce il controllo dal programma chiamante alla procedura (subroutine) chiamata

`RET`

- Restituisce il controllo dalla procedura chiamata (subroutine) al programma chiamante

CALL

- 8088, chiamate a sub routine:
  - Chiamate **ravvicinate (NEAR)**
  - Chiamate **a distanza (FAR)**

- **Il nostro Assemblatore usa solo le chiamate <u>Ravvicinate</u>**

- Destinazione: etichetta o indirizzo effettivo

- Cosa fa:
  - Salva il punto corrente di esecuzione:
    - valore corrente di IP per chiamate vicine
    - valore corrente di CS e poi di IP per chiamate lontane (cioè PC)
  - Passa il controllo alla procedura chiamata

RET

Cosa fa:

- Recupera il punto di esecuzione del chiamante:
  - valore corrente di IP per chiamate vicine
  - valore corrente di IP e poi di CS per chiamate lontane (cioè PC) ...LIFO!!!
- Ritorna il controllo alla procedura chiamante

Se la procedura chiamata è **NEAR**:

- CALL esegue **push IP**
- RET esegue **pop IP**

Se la procedura chiamata è **FAR**:

- CALL esegue **push CS** e **push IP**
- RET esegue **pop IP** e **pop CS**

Passaggio delle variabili:

- attraverso lo Stack
- impilamento in ordine inverso: LIFO !!!

Annidamento:

- chiamata di subroutine una “dentro” l’altra
- limitato dalle dimensioni dello Stack

Protocollo di programmazione

Per il programma **chiamante**

- Salvare sullo stack le variabili: LIFO!!!
- Invocazione di CALL (passaggio del controllo alla subroutine)

Per la **subroutine**

- Salvare sullo stack l'indirizzo corrente del BP (punto corrente dello stack)
- Copiare in BP il valore corrente di SP
- Recupero variabili dallo stack: LIFO!!!
- Esecuzione
- Salvataggio risultati sullo stack: LIFO!!!
- Recupero BP dallo stack (ripristino)
- Invocazione RET e ritorno del controllo al chiamante

Protocollo ***chiamante/chiamato***

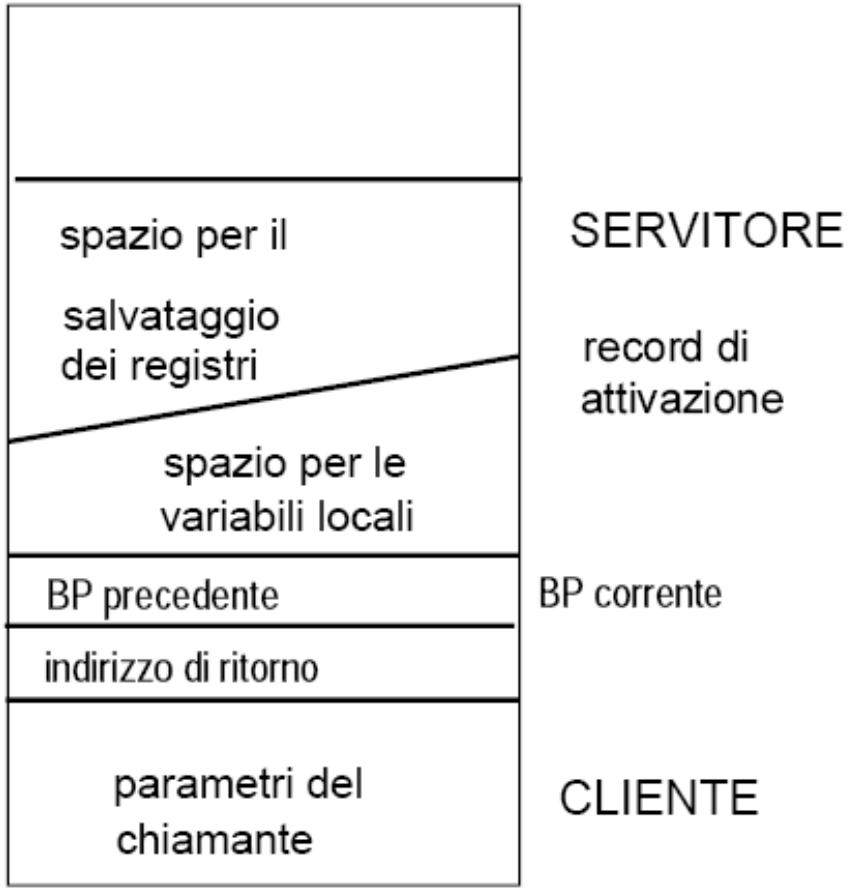
Il **programma cliente** (chiamante) deve eseguire:

```
        PUSH par1
        PUSH par2
        CALL procedura
        POP uscita ; parametro di uscita sullo stack
```

La **procedura chiamata** (**servitore**)

```
procedura PROC far
        PUSH BP
        MOV BP,SP          ;  un nuovo frame
        SUB SP,10          ; spazio per le variabili locali
              ; usate come [BP - i] con i = 2, 4, 6,...
        PUSH AX            ; salvataggio registri cliente
        PUSH BX            ; che il chiamato utilizza
        MOV AX, [BP+6] ; recupero dei parametri
        MOV BX, [BP+8] ; il primo parametro
   < fase di elaborazione ed esecuzione  >
        MOV [BP+8], parametrouscita
        ; il parametro di uscita ricopre l'ingresso
        POP BX ;  ripristino dei valori nei registri
        POP AX ;
        MOV SP,BP ; recupero variabili locali
        POP BP       ; si ristabilisce il vecchio frame
        RET 2        ; eliminazione parametro ingresso
procedura ENDP
```

MEMORIA **BASSA**



MEMORIA **ALTA**

Call FAR

- indirizzo di ritorno BP+2
- variabile2 BP+6 (BP+4 per Call NEAR)
- variabile1 BP+8 (BP+6 per Call NEAR)

Salvataggio dei registri

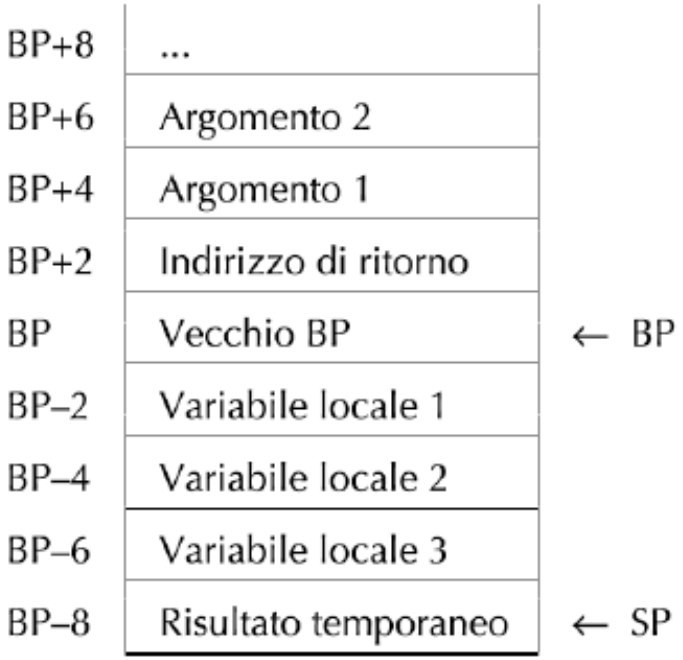
- Se la subroutine usa dei registri è bene che salvi i vecchi valori sullo stack e li ripristini all'uscita
- Di solito si salvano i registri AX BX CX DX ma non SI DI

- ....vedi esempio di prima

Recupero degli operandi (e scrittura dei risultati)

- BP funge da “punto di ancoraggio” o riferimento sullo stack per la sottoprocedura

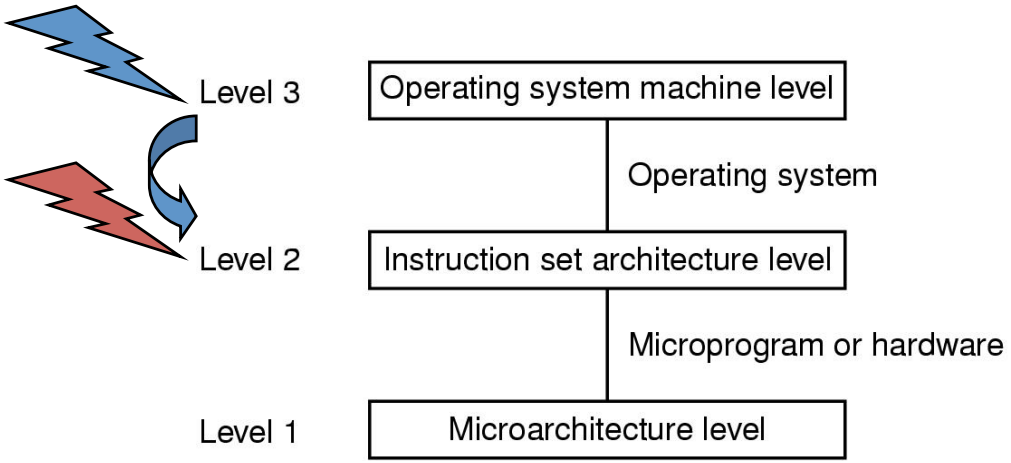
Call NEAR




```
...
PUSH Argomento2
PUSH Argomento1
CALL subRoutine
...
```

SOTTOPROCEDURE DI SISTEMA: Livello macchina del sistema operativo

- Il livello macchina del sistema operativo (OSM) contiene tutte le istruzioni disponibili ai programmatori, pressoché tutte le istruzioni del livello ISA, e le nuove istruzioni aggiunte dal sistema operativo (chiamate di sistema, *system call*).
- Una *system call* invoca un predefinito servizio del sistema operativo (es. lettura/scrittura da/in un file).
- Il livello OSM è interpretato, quindi "dietro" la chiamata di sistema c'è un'*utility* che esegue il servizio a livello ISA.
- Le chiamate fatte dal programma a livello ISA "non passano" per il sistema operativo.



Chiamate a subroutine di sistema (System call)

- Le chiamate di sistema consentono di utilizzare le procedure fornite dal sistema operativo. Per es. per Accesso ai file
- Le routine di sistema possono essere attivate con la sequenza di chiamata standard a sotto-procedura usando l'istruzione `INT` al posto della `CALL`

- Passaggio argomento attraverso lo Stack
- Risultato in:
  - Fino a 16bit → AX
  - Long (32bit) → DX:AX
- INT non modifica il valore degli altri registri (a parte AX e DX)
- Salvare i vecchi valori di AX e DX sullo Stack prima dell'invocazione di INT e recuperarli quando ritorna il controllo
- Il chiamante deve rimuovere gli argomenti dallo Stack dopo il ritorno del controllo

INT (IRET)

INT tipo-di-interrupt
IRET

Un interrupt è simile a una chiamata a procedura

Mentre una chiamata a procedura può essere NEAR o FAR e diretta o indiretta,
l'interrupt esegue sempre una chiamata FAR indiretta, **prendendo l'indirizzo della routine (di servizio) nel vettore degli interrupt**

Mentre una chiamata a procedura salva nello stack l'indirizzo di ritorno,
l'interrupt salva nello stack, oltre all'indirizzo di ritorno, anche il registro flag

INT effettua le seguenti azioni:

1. **push del registro flag nello stack** e **azzeramento dei flag TF e IF** al fine di disabilitare il single-step e il riconoscimento di altri interrupt mascherabili

2. **lettura dell'indirizzo** (di 32 bit) **della routine di servizio dell'interrupt** dalla tabella degli interrupt (la posizione in tabella è data dal tipo di interrupt)

3. **push dei registri CS, IP nello stack** e **caricamento in CS, IP del nuovo indirizzo** (come per CALL di tipo FAR)

IRET deve essere utilizzata in ogni routine di servizio degli interrupt al fine di restituire correttamente il controllo al programma che era stato interrotto (via software, o via hardware)

IRET esegue la **pop di tre word dallo stack** e **le carica**, rispettivamente, **in IP, in CS e nel registro flag**

System call DOS

DOS (Disk O. S.)

DOS funzionalità logiche di sistema
- gestori degli errori;
- primitive sui file (open, close)
- operazioni sui dispositivi
- operazioni sul sistema:

tempo, data, interrupt vector, memoria, direttori;

interrupt inferiori **(<8)** situazioni anomale
interrupt hardware da **8** a **0F**h (eventi asincroni)

Funzioni DOS:
- interrupt da 20h a 3Fh
- interrupt 21h (DOS Function Call)
- funzionalità:
  - gestione video (caratteri) e tastiera
  - gestione file (indirizzi logici)
  - gestione memoria
  - funzioni che operano su date e tempi
  - trasferimento controllo ad altri programmi
  - terminazione programma

Esempi

INT 21H; function call del DOS
selezionate in base al valore contenuto in **AH**

```
; keybinput   AH <- 1
; char = Keybinput (); il carattere in AL
; input e echo del carattere  e check <ctrl><break>
MOV AH, 1
INT   21H

; keyboutput   AH <- 2
; char = Keybioutput ();
; output del carattere in DL
MOV AH, 2
MOV DL, 'a'
INT   21H
```

System call as88

- L'assemblatore allegato al testo di Tanenbaum mette a disposizione una serie di sottoprocedure per mascherare le system call e slegarle dal sistema operativo sottostante
- Si invocano con l'istruzione **SYS**

No.	Nome	Argomenti	Valore restituito	Descrizione
5	_OPEN	*name, 0/1/2	descrittore di file	Apre un file
8	_CREAT	*name, *mode	descrittore di file	Crea un file
3	_READ	fd, buf, nbytes	# di byte	Legge n byte nel buffer buf
4	_WRITE	fd, buf, nbytes	# di byte	Scrive n byte nel buffer buf
6	_CLOSE	fd	0 se ha successo	Chiude il file con descrittore fd
19	_LSEEK	fd, offset(long), 0/1/2	posizione (long)	Sposta il puntatore del file
1	_EXIT	status		Interrompe un processo
117	_GETCHAR		carattere letto	Legge un carattere da standard input
122	_PUTCHAR	char	byte scritto	Scrive un carattere sullo standard output
127	_PRINTF	*format, arg		Stampa formattata sullo standard output
121	_SPRINTF	buf, *format, arg		Stampa formattata nel buffer buf
125	_SSCANF	buf, *format, arg		Legge gli argomenti dal buffer buf