Il Livello Microarchitettura (ii)

A cura di:

Luca Breveglieri Giuseppe Pozzi

DEI, PoliMI, Milano
luca.breveglieri,giuseppe.pozzi@polimi.it

- versione dell'11 aprile 2003 -

Approccio a livelli strutturati

Livello 5	**Linguaggi applicativi**
Livello 4	**Linguaggio assemblatore**
Livello 3	**Sistema operativo**
Livello 2	**Instruction Set**
Livello 1	**Microarchitettura**
Livello 0	**Logica digitale**

Argomenti

- Programmazione di rete e di sistema;
- Linguaggi assemblatori;
- Sistema operativo;
- Instruction set (linguaggio macchina);
- Microarchitettura (ISA);
- Livello logico digitale.

Osservazioni

- I linguaggi assemblatori utilizzano un insieme di istruzioni (instruction set) fortemente dipendente dal tipo di microarchitettura adottato.
- I linguaggi assemblatori vedono direttamente i registri del processore ed il modello di memoria.

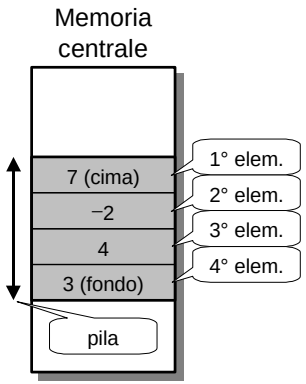
La pila del processore IJVM

Cos'è la pila (I)

- La pila (stack) è una struttura dati di memoria
- La pila è formata da una successione di parole di memoria contigue
- La pila ha un fondo (stack base)
- La pila ha una cima (stack top)
- In IJVM, si può pensare che ogni elemento (parola) contenuto nella pila sia un numero intero

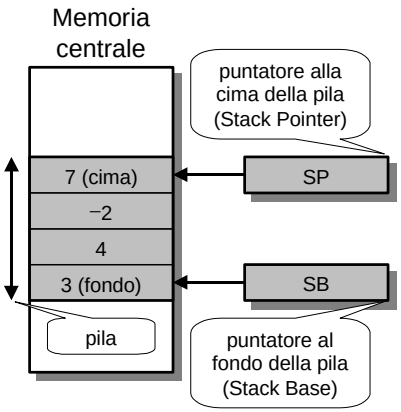
Cos'è la pila (II)

- La figura a lato mostra una pila di 4 parole (elementi)
- La cima contiene il numero 7
- Il fondo contiene il numero 3
- La pila sta nella memoria centrale



Cos'è la pila (III)

- Si usa la pila tramite due registri puntatori
- SP (Stack Pointer), punta alla cima della pila
- SB (Stack Base), punta al fondo della pila



Più in profondità

- Il registro SP (Stack Pointer) contiene l'indirizzo di memoria della cima della pila, che è il primo elemento della pila
- Il registro SB (Stack Base) contiene l'indirizzo di memoria del fondo della pila, che è l'ultimo elemento della pila

Come funziona la pila

- Alla pila si possono:
  - aggiungere elementi sulla cima
  - togliere elementi dalla cima (se la pila non è già completamente vuota)
- Aggiungere un elemento in cima: PUSH
- Togliere un elemento dalla cima: POP
- Naturalmente, si deve anche potere sapere se la pila è vuota oppure se contiene elementi

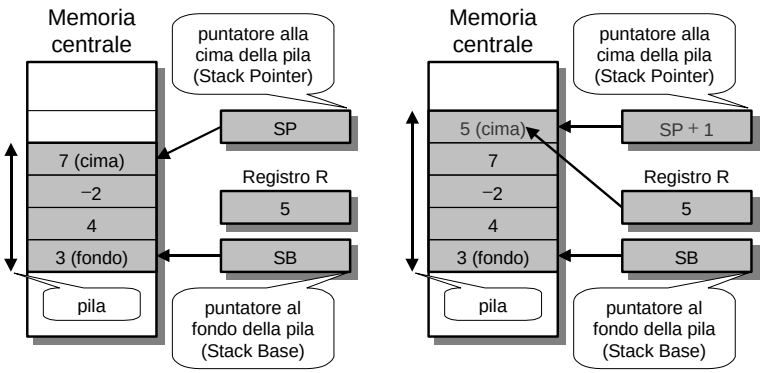
L'operazione PUSH

- Si supponga di volere copiare in pila il contenuto del registro R
  - PUSH R
- Si incrementa di 1 il registro puntatore SP (Stack Pointer)
- Si scrive il valore di R nella cella di memoria di cima della pila, che è la cella di memoria puntata da SP

Esempio: PUSH R

Animazione

Fine



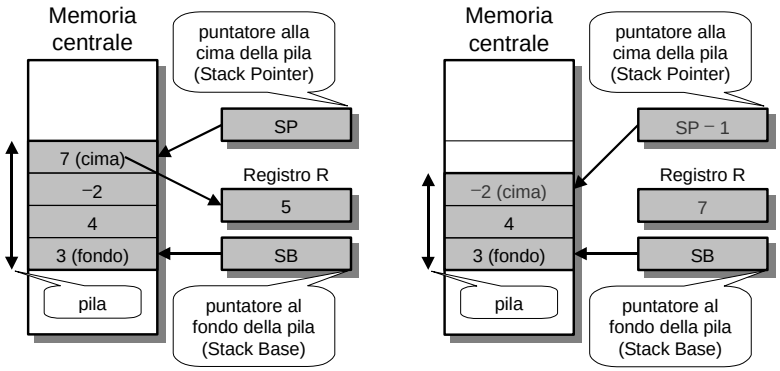
L'operazione POP

- Si supponga di volere togliere il primo elemento della pila, spostandolo nel registro R
  - POP R
- Si copia il primo elemento della pila, che si trova nella cella di memoria puntata dal registro SP, nel registro R
- Si decrementa di uno il registro puntatore SP (Stack Pointer)

Esempio: POP R

Animazione

Fine



Riassumendo

- Indicando con M(SP) la cella di memoria correntemente puntata dal registro SP, si ha:
  - Operazione: PUSH R
    - SP + 1 → SP // incrementa SP di uno
    - R → M(SP) // copia R in M(SP)
  - Operazione: POP R
    - M(SP) → R // copia M(SP) in R
    - SP – 1 → SP // decrementa SP di uno

Alcuni dettagli (I)

- Sulla pila si può impilare (PUSH) anche una costante:
  - PUSH 5 (impila la costante 5)
- Non si può invece spilare (POP) una costante, non avrebbe senso!
  - POP 5 (è insensato)

Alcuni dettagli (II)

- Prima di iniziare a usare la pila, occorre fissare il fondo della pila, inizializzando il registro SB (Stack Base)
- Se si verifica la condizione:

  SP = SB - 1

  la pila è vuota!
- A pila vuota è vietato spilare (POP)

Alcuni dettagli (III)

- La pila mostrata in precedenza cresce verso l'alto
- Si può "ribaltare" la pila, facendola crescere verso il basso

Operazione: PUSH R	Operazione: POP R
$SP - 1 \rightarrow SP$	$M(SP) \rightarrow R$
$R \rightarrow M(SP)$	$SP + 1 \rightarrow SP$

La pila di IJVM

- I linguaggi di programmazione, come il C, Java e altri ancora, permettono:
  - di avere procedure e funzioni dotate di variabili locali
  - di avere procedure e funzioni ricorsive
- La pila serve per gestire l'uso corretto delle variabili locali e il funzionamento della ricorsione

Un caso semplice (I)

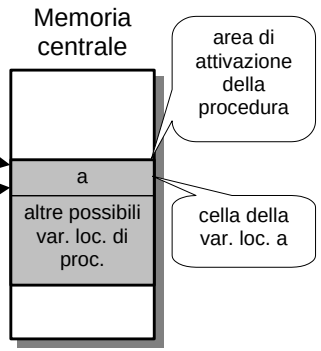
- Si potrebbe supporre di assegnare a ogni procedura (o funzione) un'area diversa di memoria centrale, in cui la procedura tiene le proprie variabili locali
- Problema: se la procedura è ricorsiva, la seconda chiamata della procedura opererebbe sulla stesse variabili locali della prima (che è un errore)

Animazione

Fine

Un caso semplice (II)

```
/* proc. ricorsiva */
void proc (int b) {
  int a;
  a = b + 1;
  if (b < 10) proc (a);
  printf ("%d", a);
} /* proc. */
```



Variabili locali

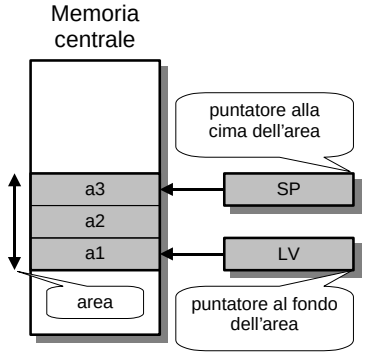
- È dunque impossibile allocare le variabili locali di procedure e funzioni in celle di memoria di indirizzo assoluto (cioè di indirizzo fissato durante la compilazione)
- Occorre utilizzare una strategia diversa: la pila di memoria (o stack)

Variabili locali in pila

- Ogni procedura (o funzione) possiede le proprie variabili locali
- Le variabili locali della procedura vengono allocate in successione in un'area di memoria chiamata "area di attivazione" (o frame) della procedura
- Le aree di attivazione delle procedure vengono impilate (operazione PUSH) oppure spilate (operazione POP)

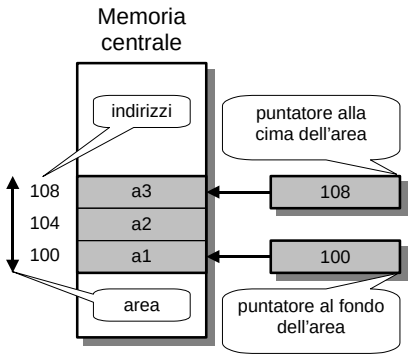
Struttura dell'area di attivazione

- L'area contiene le var. loc. della proc.: a1, a2 e a3
- Il registro SP punta alla cima dell'area (la var. a3)
- Il registo LV punta al fondo dell'area (la var. a1)



Esempio di area di attivazione

- Le var. loc. occupano ciascuna 4 byte
- Gli indirizzi di memoria si riferiscono ai byte

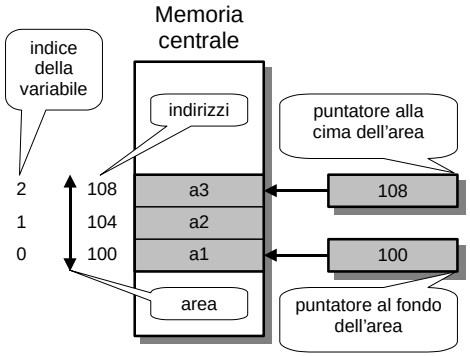


Spiazzamento nell'area di attivazione

- Ogni variabile locale ha uno spiazzamento (offset) all'interno dell'area di attivazione
- Lo spiazzamento è calcolato come distanza in byte a partire dal fondo dell'area di attivazione
- La variabile locale al fondo dell'area di attivazione ha spiazzamento 0

Esempio di spiazzamento

- La var. loc. a1 ha spiazzamento 0
- La var. loc. a2 ha spiazzamento 4
- La var. loc. a3 ha spiazzamento 8



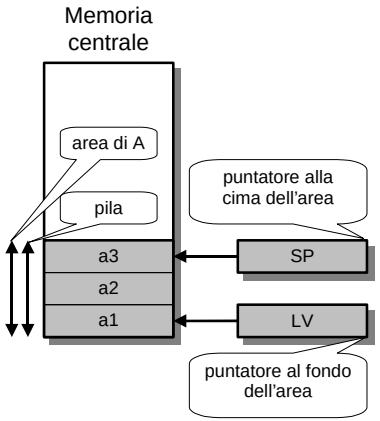
Per comodità, le variabili si possono indicare tramite un indice logico (numvar), partendo da zero; è solo un numero progressivo utile per indicare la variabile

Impilamento delle aree (I)

- La procedura A ha un'area di attivazione contenente 3 variabili locali: a1, a2 e a3
- La procedura B ha un'area di attivazione contenente 4 variabili locali: b1, b2, b3 e b4
- La procedura A, a un certo punto della sua esecuzione, chiama la procedura B

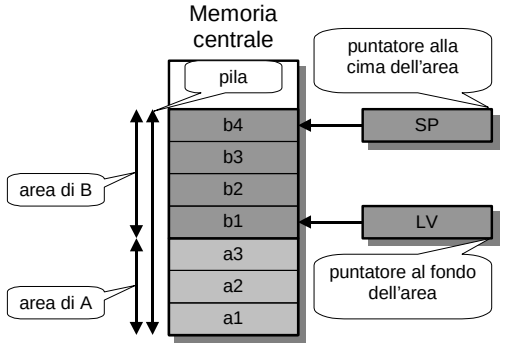
Impilamento delle aree (II)

- Subito dopo l'entrata in esecuzione di A, la pila contiene l'area di attivazione di A
- Il registro LV punta al fondo dell'area di attivazione di A



Impilamento delle aree (III)

- Quando B viene chiamata da A, sopra l'area di A si impila l'area di B
- Il registro LV punta al fondo dell'area di B



Impilamento delle aree (IV)

- Quando si impila l'area di attivazione della procedura B, il registro LV viene incrementato in modo da farlo puntare al fondo dell'area di attivazione di B, e non più di A
- In ogni istante, le var. loc. dell'area di attivazione corrispondente alla procedura correntemente in esecuzione sono indicate dai rispettivi spiazzamenti rispetto al registro LV

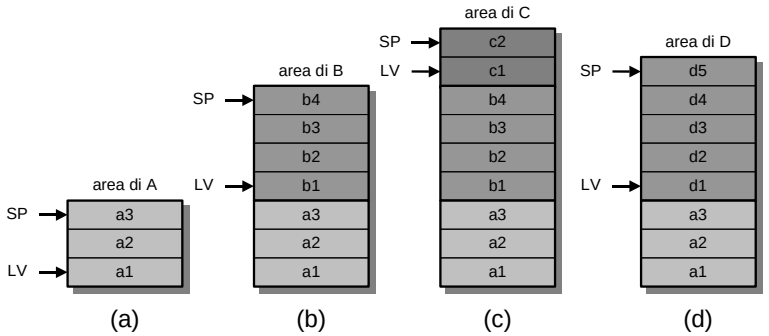
Chiamate multiple (I)

- Si consideri la seguente sequenza di chiamate a procedura:
  - La procedura A entra in esecuzione (caso a)
  - A chiama la procedura B (caso b)
  - B chiama la procedura C (caso c)
  - C termina e anche B termina, e A chiama la procedura D (caso d)

Chiamate multiple (II)

Animazione

Fine



C e B terminano ed entra in esecuzione D

Aree di attivazione e procedure ricorsive

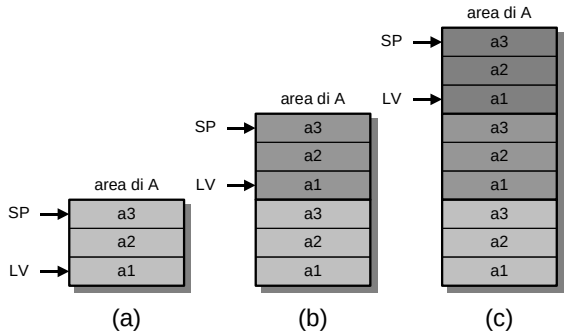
- Il meccanismo delle aree di attivazione è anche in grado di tenere separate le variabili locali di chiamate ricorsive della stessa procedura
- A ogni chiamata ricorsiva, si impila una nuova area di attivazione, identica come struttura a quella precedente

Impilamento di aree ricorsive

Animazione

Fine

La procedura A richiama se stessa 3 volte



A chiama ricorsivamente sé stessa (3^{a} volta)

Calcolo delle espressioni

- La pila ha anche un ruolo fondamentale nel calcolo delle espressioni logico-matematiche, contenenti cioè operazioni logiche oppure aritmetico-algebriche
- Il calcolo di un'espressione può essere sempre ricondotto a una successione di operazioni PUSH e POP, intercalate da operazioni logiche o matematiche

Esempio

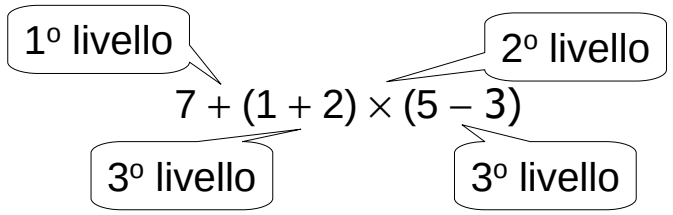
Calcolare: $7 + (1 + 2) \times (5 - 3)$

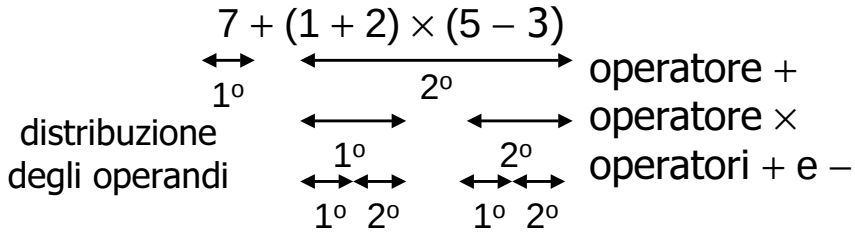
Occorre prima riscrivere l'espressione in forma postfissa (postfix)

- Algoritmo postfix: si parte dagli operatori di 1º livello (gli ultimi da calcolare), e
  - (1) si sposta l'operatore a destra del 2º operando
  - (2) si cancellano le eventuali parentesi
  - (3) si passa agli operatori di livello immediatamente superiore, tornando a (1)
- Quando non ci sono più operatori, termina

Analisi dell'espressione

Animazione Fine





Trasformazione postfix

Animazione Fine

$7 + (1 + 2) \times (5 - 3)$	ESPRESSIONE ORIG.
$7 + (1 + 2) \times (5 - 3)$	- partenza
$7\ (1 + 2) \times (5 - 3)\ +$	- spostato l'operatore +
$7\ (1 + 2)\ (5 - 3) \times +$	- spostato l'operatore ×
$7\ 1 + 2\ 5 - 3 \times +$	- cancellate le parentesi degli operandi dell'operatore ×
$7\ 1\ 2 + 5\ 3 - \times +$	- spostati gli operatori + e –; fine
$7\ 1\ 2 + 5\ 3 - \times +$	FORMA POSTFIX

Calcolo tramite la pila

Animazione Fine

$7\ 1\ 2 + 5\ 3 - \times +$

$7\ 1\ 2 + 5\ 3 - \times +$ FINE: pila vuota, risultato 13

Simulazione del procedimento di calcolo

Sequenza di operazioni PUSH/POP

Si usano i registri temporanei R1, R2 e R3

- PUSH 7
- PUSH 1
- PUSH 2
- POP R1
- POP R2
- R3 = R2 + R1
- PUSH R3
- PUSH 5
- PUSH 3
- POP R1
- POP R2
- R3 = R2 – R1
- PUSH R3
- POP R1
- POP R2
- R3 = R2 × R1
- PUSH R3
- POP R1
- POP R2
- R3 = R2 + R1
- PUSH R3

In pila resta un solo elemento, il risultato

Trasformazione

- Si introducano le istruzioni seguenti:
  - IADD: togli i due numeri in cima alla pila, addizionali e scrivi la somma in cima alla pila
  - ISUB: togli i due numeri in cima alla pila, sottraili (si supponga che il primo numero tolto sia il sottraendo, il secondo il minuendo) e scrivi la differenza in cima alla pila
  - IMUL: togli i due numeri in cima alla pila, moltiplicali e scrivi il prodotto in cima alla pila
- Sono istruzioni del processore IJVM (tranne IMUL, ma che si potrebbe introdurre)

Il programma di calcolo in linguaggio macchina IJVM

- PUSH 7 // scrivi 7 sulla cima della pila
- PUSH 1 // scrivi 1 sulla cima della pila
- PUSH 2 // scrivi 2 sulla cima della pila
- IADD // addiziona i 2 numeri in cima alla pila
- PUSH 5 // scrivi 5 sulla cima della pila
- PUSH 3 // scrivi 3 sulla cima della pila
- ISUB // sottrai i 2 numeri in cima alla pila
- IMUL // moltiplica i 2 numeri in cima alla pila
- IADD // addiziona i 2 numeri in cima alla pila

Simulazione del programma

Animazione

Fine

7 1 2 + 5 3 – × +

13

FINE: nella pila resta il risultato del calcolo

Simulazione dell'esecuzione del programma IJVM per il calcolo dell'espressione

Considerazioni

- Il calcolo delle espressioni tramite la pila si generalizza facilmente anche a:
  - espressioni contenenti variabili
  - espressioni contenenti operatori booleani
  - espressioni contenenti operatori relazionali
  - espressioni contenenti operatori algebrici
  - espressioni con numeri reali, complessi, …
- Il compilatore analizza l'espressione, ne calcola la forma postfix e genera il programma di calcolo dell'espressione