ESERCIZIO 3 – GRUPPI C: Sperimentazione della ricorsione in IJVM

In una lezione di laboratorio abbiamo visto come realizzare il metodo "fattoriale(n)" ricorsivo in IJVM e come interpretare l'evoluzione dello stack ad ogni INVOKEVIRTUAL e ad ogni IRETURN.

L'esercizio 3 (il terzo e ultimo di tre esercizi validi come esonero dalla prova pratica del corso di Architetture 2) consiste nell'osservazione del comportamento di un programma che include un metodo ricorsivo. Tale programma dovrà essere sottoposto a uno o più test utili per illustrare come si sviluppano le chiamate ricorsive e come vengono allocati e poi eliminati i record di attivazione in momoria..

Nella relazione dovrà essere dettagliatamente documentata l'evoluzione dello stack per una porzione significativa di esecuzione del programma: in particolare si dovranno mostrare delle "fotografie" dello stack (potete ricopiarne il contenuto in tabelle) in occasione delle chiamate al metodo ricorsivo (a diversi livelli di annidamento). Per ogni nuovo record di attivazione presente nella "fotografia" dovrete indicare cosa contiene ogni suo elemento (facendo riferimento allo schema allegato, simile alla figura del libro di testo, ma disegnata in modo che gli indirizzi crescano verso il basso, rappresentazione più simile a ciò che si vede nella finestra "Memory" dell'emulatore).

NOTA: se doveste trovarvi nella necessità di spostarvi nella finestra Memory in modo da visualizzare una diversa zona di memoria basta inserire un nuovo indirizzo (per es. 4000, che è l'indirizzo contenuto nel registro CPP) nella finestrella in basso al centro e premere OK. In questo modo potrete per esempio vedere l'intero contenuto della *constant pool*. Per tornare alla visualizzazione dello stack inserire nella casella in basso l'indirizzo 8000. Analogamente si può spostare la "vista" della finestra Method Area sulla zona di memoria che contiene il codice.

I breakpoint si inseriscono scegliendo "insert breakpoint" dal menu "breakpoint" e digitando l'indirizzo dell'area dei metodi dove volete sia inserito il breakpoint. Se volete inserire più breakpoint ripetete l'operazione più volte. Ogni volta che si ricarica un programma oppure si preme il tasto "Reset" vengono cancellati tutti i breakpoint (che dovranno essere reinseriti se si vuole rieseguire il programma interrompendolo in quegli stessi punti).

Nella pagina seguente è descritto il metodo ricorsivo da testare (già implementato nel programma TorriHanoi.jas allegato che attende in input il numero di dischi iniziale e poi stampa ogni mossa e infine stampa il numero complessivo di mosse):

PER I GRUPPI A CUI E' ASSEGNATO IL TIPO DI ESERCIZIO 3C:

Il metodo ricorsivo fornito calcola gli spostamenti di anelli richiesto dal gioco "torre di Hanoi"

Obiettivo del gioco: Spostare ***n*** dischi, dal piolo 1 al piolo 3 (utilizzando anche il 2, se necessario): si può spostare un solo disco per volta. Non si deve mai mettere un disco grande sopra ad un disco più piccolo.

torri-hanoi(n,p1,p2) // sposta n dischi da p1 a p2

- Se n=1: stampare mossa: p1->p2
- Se n>1:
  - Sposta n-1 dischi da p1 a p3 // torri-hanoi,(n-1,p1,p3)
  - Sposta 1 disco da p1 a p2 // torri-hanoi,(1,p1,p2)
  - Sposta n-1 dischi da p3 a p2 // torri-hanoi,(n-1,p3,p2)

Nota: p3 è l'indice del piolo rimanente, diverso da p1 e p2

Per facilitare l'immissione di valori (in base 10) da passare al metodo e la stampa del risultato (sempre in base10) sono anche inclusi due metodi: LeggiDecimale e StampaDecimale.

Nell'eseguire i test osservate l'evoluzione dello stack a mano a mano che si sviluppano richiami annidati della funzione ricorsiva e i vari record di attivazione si impilano uno sull'altro, osservate in particolare che parametri e variabili locali assumono valori diversi nei record di attivazione impilati, pur essendo originati dalla stessa funzione "astratta". Osservate anche che il valore "PC del chiamante" salvato sullo stack è diverso nel record di attivazione della istanza richiamata direttamente nel main rispetto a quella richiamate all'interno della funzione stessa. Inoltre ragionate sul costo del calcolo (sia in termini di numero di operazioni eseguite sia in termini di spazio occupato sullo stack) se si utilizza la definizione ricorsiva della funzione per calcolarla.