Architettura degli Elaboratori, 2007-08

Esercitazione 3 di verifica

Soluzione: venerdì 2 novembre

Domanda 1

Compilare in assembler D-RISC un programma così definito:

- opera su una lista "linkata" il cui generico elemento contiene gli indirizzi logici base di tre array di interi *A[N]*, *B[N]*, *C[N][N]*, con *N = 1K*, diversi per ogni elemento. Il puntatore ad ogni elemento della lista è una coppia (indicatore di fine lista, indirizzo logico). L'indirizzo della testa della lista, contenente il puntatore al primo elemento, è allocato in un registro generale;
- per tutti gli elementi della lista, viene effettuata la seguente elaborazione:

$$\forall i, j = 0 \,..\, N-1 : C_{i,j} = abs\,(A_i - B_j)$$

da implementare come *procedura* i cui parametri formali sono allocati in registri generali.

Spiegare come sono stati allocati, ed eventualmente inizializzati, i registri generali, e come sono state applicate le regole di compilazione.

Domanda 2

I seguenti quesiti fanno riferimento alla Domanda 1; le spiegazioni devono essere date in modo rigoroso ed usando la corretta terminologia:

a) spiegare quali modi di indirizzamento sono stati utilizzati nella compilazione, distinguendo tra modi primitivi in D-RISC e modi non primitivi, in quest'ultimo caso spiegando come sono stati implementati;

b) descrivere la memoria virtuale e lo spazio di indirizzamento del processo corrispondente al programma, spiegando quali locazioni della memoria virtuale sono inizializzate a tempo di compilazione. Si suppone che il PCB di un processo occupi 128 parole e che le altre informazioni "collegate" a questo processo occupino complessivamente 64K parole;

c) spiegare se la memoria virtuale del processo è tutta allocata staticamente o meno;

d) si supponga che, a causa della dimensione della lista, lo spazio di indirizzamento del processo abbia dimensione maggiore di 4G parole; spiegare se questa condizione è rilevata in fase di compilazione o in fase di esecuzione del processo.

Domanda 3

I seguenti quesiti hanno carattere di generalità; le spiegazioni devono essere date in modo rigoroso ed usando la corretta terminologia:

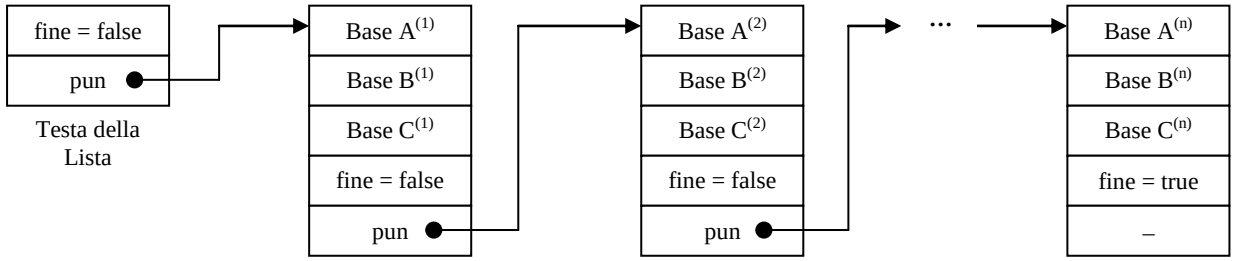
a) si consideri un processo in cui uno stesso registro generale è utilizzato per contenere, in momenti diversi, i valori di più variabili tra loro diverse. Spiegare se la seguente affermazione è vera o falsa o vera sotto certe condizioni: "di conseguenza non è possibile sospendere l'esecuzione del processo in istanti qualsiasi, e successivamente riprenderla in istanti qualsiasi, in quanto non sarebbe possibile determinare quale variabile è allocata in tale registro e quale valore assume";

b) si supponga di disporre di librerie predefinite che consentano di aggiungere nuovi elementi e di eliminare elementi di una lista "linkata". Spiegare se queste librerie devono provvedere all'allocazione dinamica della memoria virtuale del processo, oppure all'allocazione dinamica della memoria principale del calcolatore, oppure ad entrambe.

Soluzione

Domanda 1

La struttura dati lista per questo problema è schematizzabile come nella figura seguente, dove è mostrato il caso di lista non vuota:



$A^{(i)}$, $B^{(i)}$, $C^{(i)}$ significano l'istanza i-esima degli array A, B, C usati dal programma. Il valore di n (≥ 0) è noto a tempo di compilazione, così come i valori di tutti gli array.

Ogni elemento della lista consta di una parte *valore*, di tre parole (indirizzi base di tre array), e di una parte *puntatore* all'elemento successivo, di due parole (indicatore booleano di fine lista e indirizzo dell'elemento successivo).

Il programma sorgente consiste in un ciclo che scandisce tutta la lista, passando da un elemento al successivo mediante puntatori, finché non si incontra la condizione di fine lista. Ogni elemento di lista ($A^{(i)}$, $B^{(i)}$, $C^{(i)}$) viene elaborato chiamando la procedura.

Programma principale

Il programma ha la struttura complessiva di un **do while**,

- preceduto da un fase di inizializzazione che opera sulla testa della lista ricavando il puntatore al primo elemento oppure, se la lista è vuota, provocando subito la terminazione del processo;
- la cui guardia dipende dal valore dell'indicatore di fine lista.

I registri inizializzati a tempo di compilazione sono *Rtesta*, *Rproc* e *RN* [1]. Il compilatore alloca anche i registri temporanei *Rfine*, *Rpun*.

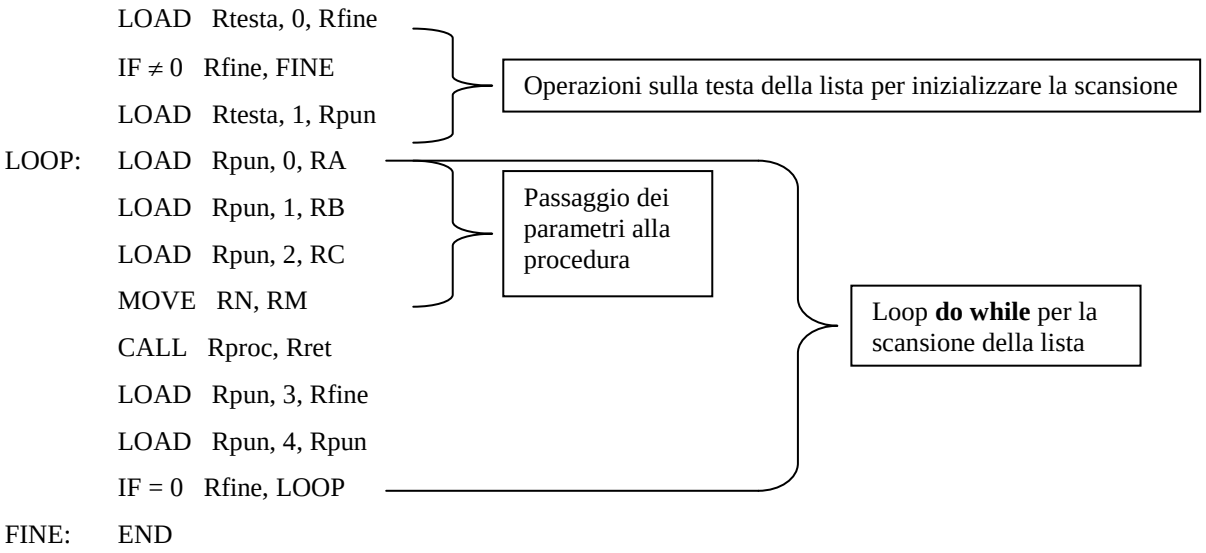
I registri *RA*, *RB*, *RC*, e *RM* sono destinati a contenere i parametri passati alla procedura, rispettivamente: indirizzo base di A, di B, di C, e dimensione degli array.

Questi registri, così come *Rret* per il ritorno da procedura, sono imposti dalla compilazione della procedura se questa è già disponibile (ad esempio, sotto forma di libreria).

Se invece la compilazione del programma principale e della procedura avvengono contemporaneamente, il compilatore sceglie liberamente anche i registri *RA, RB, RC, RM, Rret*.

Applicando le regole di compilazione per composizione, la versione del programma principale compilato è mostrata di seguito:

[1] La dizione "*registro Ri*" è l'usuale abbreviazione per "*registro generale di indirizzo i*".



Si osservi la posizione dell'istruzione *LOAD Rpun, 4, Rpun*, che viene eseguita una volta di più del necessario (senza effetti di sorta sulla correttezza): eseguirla solo a condizione di non uscire dal loop introdurrebbe istruzioni di salto addizionali e/o altre complicazioni nelle struttura del codice compilato, con conseguenze negative anche sulle prestazioni.

Procedura

Lo pseudo-codice sorgente è:

```
for (i = 0; i < M; i++)
  for (j = 0; j < M; j++)
        C[i][j] = if A[i] ≥ B[j] then A[i] – B[j] else B[i] – A[j]
```

L'array bidimensionale C è rappresentato in memoria *per righe*. Questo significa che, per passare da una riga alla successiva, è sufficiente incrementare l'indirizzo base della dimensione della riga stessa:

$$base\_riga\ i = base\_array + i * M$$

L'indice del loop più esterno *(i)* deve essere inizializzato (a zero) a programma con una istruzione esplicita, in quanto si tratta di una procedura che, ogni volta che viene chiamata, deve ri-inizializzare il suo stato. L'indice del loop più interno *(j)* deve, in qualunque programma, essere inizializzato (a zero) a programma con una istruzione esplicita.

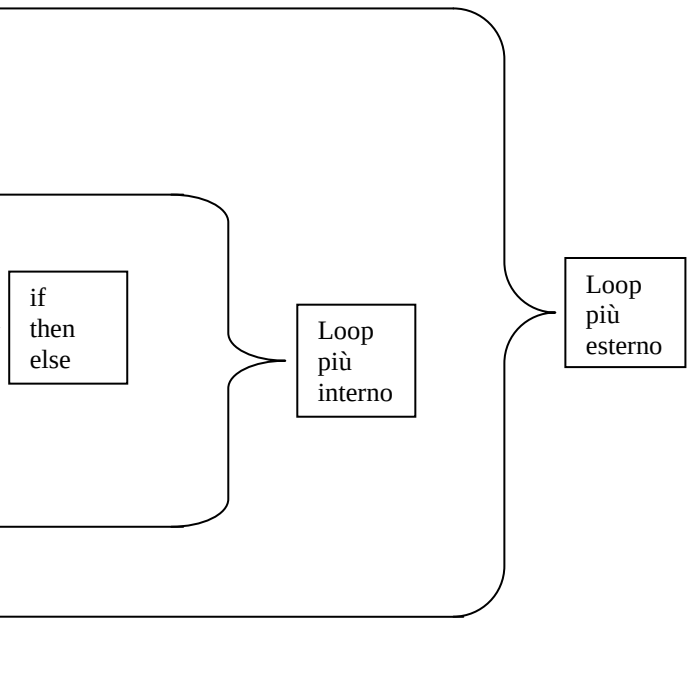
Lo stesso indice *j* serve per il calcolo dell'indirizzo delle componenti di C rispetto alla base di ogni riga.

Entrambi i loop **for** si implementano come **do while**, in quanto è sempre $M > 0$.

Sono usati i registri temporanei *Ra*, *Rb*, *Rc*, *Rt*, *RCriga*; eventualmente, se usati anche dal programma chiamante, all'atto della chiamata ne viene salvato il contenuto in locazioni di comodo del PCB e vengono ripristinati all'atto del ritorno.

Ricordando che i parametri sono passati attraverso i registri *RA, RB, RC, RM*, e che l'indirizzo di ritorno si trova in *Rret*, la compilazione della procedura è la seguente:

```
        CLEAR   Ri
LOOP1:  MUL     Ri, RM, Rt
        ADD     RC, Rt, RCriga
        LOAD    RA, Ri, Ra
        CLEAR   Rj
LOOP2:  LOAD    RB, Rj, Rb
        IF ≥    Ra, Rb, THEN
        SUB     Rb, Ra, Rc
        GOTO    CONT
THEN:   SUB     Ra, Rb, Rc
CONT:   STORE   RCriga, Rj, Rc
        INCR    Rj
        IF <    Rj, RM, LOOP2
        INCR    Ri
        IF <    Ri, RM, LOOP1
        GOTO    Rret
```



Come nel programma principale, esiste una istruzione eseguita una volta in più del necessario (*ADD RC, RN, RC*), per ragioni di maggiore strutturazione del codice e, in ultima analisi, di maggiore efficienza.

Domanda 2

a) Gli indirizzamenti primitivi in D-RISC sono:

- *assoluto* (registri generali),
- *base più indice indiretto via registro* (indirizzi di MV in LOAD, STORE),
- *relativo a IC* (indirizzi di salto).

L'indirizzamento *indiretto via memoria*, usato per i puntatori della lista, non è primitivo ed è emulato mediante una sequenza di istruzioni, la prima delle quali è una LOAD che fornisce in un registro generale l'indirizzo base della struttura contenente la locazione su cui operare, ad esempio:

```
        LOAD    Rtesta, 1, Rpun
        …
        LOAD    Rpun, 3, Rfine
        …
        IF = 0  Rfine, LOOP
```

b) *Memoria virtuale del processo:*

- il PCB del processo occupa le prime 128 locazioni, in modo da conoscere implicitamente il suo indirizzo base (= 0). È inizializzato limitatamente all'immagine di *IC* (= 128), le immagini dei registri generali *Rtesta*, *Rproc* e *RN*, il puntatore alla Tabella di Rilocazione del processo, ed altre eventuali informazioni connesse ai servizi;
- le istruzioni del programma principale occupano le successive 12 locazioni;

- la testa della lista occupa le successive 2 locazioni: inizializzate;
- gli elementi della lista occupano le successive *5n* locazioni: inizializzate;
- gli *n* array A occupano le successive *nK* locazioni : inizializzate;
- gli *n* array B occupano le successive *nK* locazioni : inizializzate;
- gli *n* array C occupano le successive *nM* locazioni: : non inizializzate;
- le istruzioni della procedura occupano le successive 16 locazioni;
- informazioni da "linkare" al processo (istruzioni e dati per il collegamento ai servizi di sistema e la cooperazione del processo con gli altri processi): successive 64K locazioni;
- Tabella di Rilocazione del processo: se la memoria è organizzata a pagine di 1024 parole, detto *L* il numero complessivo di indirizzi logici del processo (*128 + 30 + 5n + 2nK + nM + 64K +* dimensione della stessa Tabella di Rilocazione), le entrate della Tabella di Rilocazione sono *L/1024.* La Tabella non è inizializzata (a rigore è parzialmente inizializzata, nel senso che ogni entrata contiene dei valori che indicano la non significatività degli altri valori presenti nell'entrata stessa).

c) Nelle ipotesi del problema, secondo le quali la lista è di dimensioni fisse, tutta la memoria virtuale del processo è allocata *staticamente*.

(Se - in un caso diverso - la computazione prevedesse creazione / distruzione di elementi di lista, e dei relativi array A, B, C puntati da essi, queste informazioni dovrebbero essere allocate dinamicamente. Si veda il successivo punto b della Domanda 3.)

d) Nelle ipotesi del problema, secondo le quali la lista è di dimensioni fisse, la condizione di superamento della massima capacità dello spazio di indirizzamento (4G parole, in quanto la macchina D-RISC opera con indirizzi logici di 32 bit) è rilevata *a tempo di compilazione*.

(Nel caso, diverso, di allocazione dinamica, la condizione deve essere controllata anche a tempo di esecuzione).

Domanda 3

a) L'affermazione è falsa.

I registri sono allocabili dinamicamente *mediante esplicite istruzioni* che il compilatore inserisce nel codice per sfruttare al meglio i registri stessi.

Supponiamo che, fino ad una certa fase della compilazione, tutti i registri siano stati allocati, e che si presenti la necessità o la convenienza di allocare un'ulteriore variabile in un registro. Il compilatore sceglie uno dei registri "meno utili", sia *Rx*, e, mediante una STORE, ne salva il contenuto in una locazione di comodo (*LOC*) in memoria (virtuale). In una fase successiva della compilazione, se e quando occorresse il vecchio valore di *Rx*, verrà provocata, mediante una LOAD, una copia dalla locazione *LOC* in un registro generale *Ry* (non necessariamente coincidente con *Rx*), il cui contenuto corrente verrà eventualmente salvato in locazioni di comodo in memoria se, in futuro, ce ne fosse ancora bisogno.

Le locazioni di comodo (*LOC*) possono appartenere al PCB del processo, nel quale alcune locazioni (non coincidenti con le immagini di RG e IC) sono riservate a questo scopo, come fossero una sorta di ulteriori "registri lenti".

In fase di commutazione di contesto, tutti i registri generali verranno salvati nell'immagine nel PCB, e le altre locazioni del PCB rimarranno inalterate e disponibili per quando il processo verrà eventualmente ripreso.

b) Si tratta di allocazione dinamica della *memoria virtuale*.

Poiché il processore genera esclusivamente indirizzi logici, ogni informazione che può essere acceduta deve avere un indirizzo logico unico; di conseguenza, volendo aggiungere nuovi elementi alla lista (in generale, volendo aggiungere nuovi oggetti al processo), è necessario stabilire nuovi indirizzi logici con i quali riferirli.

Va da sé che la memoria principale, se (come di regola) è allocata dinamicamente (ad esempio, con paginazione), verrà a maggior ragione allocata dinamicamente per i nuovi elementi della lista, ma questo è indipendente dal fatto che sia stata allocata dinamicamente la memoria virtuale: l'allocazione dinamica della memoria principale si ha anche per processi la cui memoria virtuale sia tutta allocata staticamente.