Laboratorio di Architetture degli elaboratori n. 16, 20-22/4/04

Strutture dati dinamiche: LISTE.

Una lista è una struttura lineare analoga ad un vettore senza il vantaggio dell'accesso diretto ad un qualsiasi elemento ma con il vantaggio di poter crescere o diminuire in dimensione dinamicamente.

Ogni dato viene memorizzato in un nodo, dove si trova anche un puntatore con l'indirizzo del successivo elemento della lista, oppure il puntatore nullo (valore 0) che indica il termine della lista (la lista vuota). La convenzione utilizzata per memorizzare un nodo di una lista è la seguente:

puntatore all'elemento successivo (32 bit)	Dato (dim a seconda del tipo)

La dimensione del dato ovviamente dipenderà dal tipo di lista (liste di interi a 32 bit=una word, liste di char=un byte, liste di stringhe=un numero variabile di char terminati dal carattere 0, eccetera).

Analogamente a quanto fatto per vettori e matrici solitamente utilizzeremo il termine *indirizzo base* di una lista per denotare l'indirizzo del primo nodo di una lista. Il primo nodo di una lista viene anche detto testa, quindi l'indirizzo della testa di una lista *L* coincide con l'indirizzo base di *L*. Se la lista è vuota il suo indirizzo base sarà 0 e ovviamente non ci saranno nodi in memoria.

Si abuserà spesso la notazione indicando una lista il cui indirizzo del nodo iniziale è contenuto in un registro col nome del registro stesso. Si dirà quindi "la lista A0" invece di "la lista il cui indirizzo del primo nodo è contenuto in A0". Data una lista "A0", il valore del contenuto della testa è 4(A0), mentre il puntatore all'elemento successivo è (A0). Si testino opportunamente i seguenti stralci di codice che operano su liste:

- per caricare il secondo elemento di una lista di byte "A0" in T0:

```
beqz    $a0,error
lw      $a0,($a0)
beqz    $a0,error
lb      $t0,4($a0)
```

- per cancellare il nodo successivo di "A0":

```
beqz    $a0,error
lw      $t0,($a0)
beqz    $t0,error
lw      $t0,($t0)
sw      $t0,($a0)
```

- per inserire "A1" a seguito di "A0":

```
beqz    $a0,error
beqz    $a1,error
lw      $t0,($a0)
sw      $t0,($a1)
sw      $a1,($a0)
```

- per posizionare A0 sul nodo che contiene (il valore di) T0, se esiste:

```
loop: beqz $a0,failure
      lb   $t1,4($a0)
```

```
        beq  $t0,$t1,success
        lw   $a0,($a0)
        b    loop
```

Per **capire** meglio il codice si consiglia di simularlo **manualmente** su una rappresentazione grafica.

Per una **verifica corretta** (col simulatore) si definiscano le liste di test in modo non contiguo, ad esempio per la lista di interi [1,2,3,4]:

```
lista1:  .word nodo2
         .word 1
nodo4:   .word 0
         .word 4
nodo3:   .word nodo4
         .word 3
nodo2:   .word nodo3
         .word 2
```

oppure per la lista di stringhe [“Laboratorio”, “di”, “Architetture”,””]:

```
nodoC:   .word    nodoD
         .asciiz  “Architetture”
nodoB:   .word    nodoC
         .asciiz  “di”
lista2:  .word    nodoB
         .asciiz “Laboratorio”
nodoD:   .word    0
         .asciiz  “”
```

ESERCIZIO 1

Si progetti una subroutine che accetti in ingresso l’indirizzo base di una lista di interi a 8 bit mediante il registro A0 e produca in uscita nella word meno significativa del registro V0 la lunghezza della lista. Con lunghezza di una lista si intende il numero di nodi di una lista. **N.B.** la lunghezza di una lista vuota (cioè di indirizzo base 0 word) è 0 word.

ESERCIZIO 2

Si progetti una subroutine che accetti in ingresso:

1. un numero naturale k a 32 bit nel registro A0
2. l’indirizzo base di una lista L di valori su $[0,k]$ nel registro A1
3. l’indirizzo base di un vettore V di naturali a 32 bit di lunghezza k+1 nel registro A2.

La subroutine deve assegnare ad ogni elemento $V[i]$ il numero di occorrenze del valore i nella lista L. Ad esempio con k=5 e la lista `[1,2,3,1]` dovremo produrre il vettore

0	2	1	1	0	0

(Visto che non è specificato altrimenti, il contenuto iniziale del registro V deve essere assunto indefinito.)