Laboratorio di Architetture degli elaboratori n. 15, 2-3/5/02

ESERCIZIO

Si progetti una subroutine assembler 68000 che accetti in ingresso:

1) un numero positivo *m* senza segno a 16 bit nella word meno significativa del registro D0,
2) un numero positivo *n* senza segno a 16 bit nella word meno significativa del registro D1,
3) l'indirizzo base di una matrice *M* di $n \times m$ numeri con segno a 8 bit, organizzata per righe, mediante il registro A0,
4) un numero *a* con segno a 8 bit nella word meno significativa del registro D2,
5) un numero *b* con segno a 8 bit nella word meno significativa del registro D3,

La subroutine assegna al byte meno significativo di D0 il valore -1 se le *m* somme degli elementi di colonna di *M* sono tutte comprese nell'intervallo [*a*,*b*], un valore non negativo altrimenti. Ad esempio per la matrice $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 9 & 7 & 5 & 3 & 1 \\ 1 & 3 & 5 & 5 & 5 \end{pmatrix}$ e l'intervallo [11,13], la routine ritorna -1 in quanto le somme 11,12,13,12 e 11 sono comprese nell'intervallo [11,13]. Quindi nel caso

```
matrice  dc.b  1,2,3,4,5
         dc.b  9,7,5,3,1
         dc.b  1,3,5,5,5
```

la subroutine deve rientrare con il byte meno significativo del registro D0 a FF

```
check    movem.l d1/d4-d7,-(sp) salvataggio registri
         move.w  d0,d4          d4 manterrà il valore m
         subq.w  #1,d0          contatore ciclo colonne
         subq.w  #1,d1          valore iniziale ciclo righe
row      move.w  d1,d5          contatore del ciclo righe
         move.w  d0,d6          inizializzo (l'indice) d6 con d0
         clr.b   d7             accumulatore somma colonna
columsum add.b   (a0,d6.w),d7   aggiorno accumulatore
         add.w   d4,d6          sposto d6 sul prossimo elemento
         dbra    d5,columsum    continuo la colonna in esame
         cmp.b   d2,d7          somma è inferiore ad a?
         blt.s   fail           nel caso esce con fallimento
         cmp.b   d3,d7          somma è superiore a b?
         bgt.s   fail           nel caso esce con fallimento
         dbra    d0,row         passo alla prossima riga
success  moveq.l #1,d0          assegno l'output successo
         bra.s   exit           e vado a terminare
fail     moveq.l #0,d0          assegno l'output fallimento
exit     movem.l (sp)+,d1/d4-d7 ripsistino i registri
         rts
```

ESEMPI LISTE

Si testino opportunamente i seguenti stralci di codice che operano su liste. La convenzione utilizzata per memorizzare un nodo di una lista è la seguente:

puntatore all'elemento successivo (Long)	dato

La dimensione del dato ovviamente dipenderà dal tipo di lista (in caso di liste di interi a 16 bit avremo una word, in caso di liste di stringhe ci saranno un numero variabile di char terminati dal carattere 0, eccetera).

Per una verifica corretta si definiscano le liste di test in modo non contiguo, ad esempio per la lista di interi 1,2,3,4:

```
lista1   dc.l  nodo2
         dc.w  1
nodo4    dc.l  0
         dc.w  4
nodo3    dc.l  nodo4
         dc.w  3
nodo2    dc.l  nodo3
         dc.w  2
```

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

```
nodoC    dc.l  nodoD
         dc.b  “Architetture”,0
nodoB    dc.l  nodoC
         dc.b  “di”,0
lista2   dc.l  nodoB
         dc.b  “Laboratorio”,0
nodoD    dc.l  0
         dc.b  0
```

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). Quindi

- per caricare il secondo elemento di una lista di byte “A0” in D0:

```
cmpa.l   #0,a0
beq      error
movea.l  (a0),a0
cmpa.l   #0,a0
beq      error
move.b   4(a0),d0
```

- per cancellare il nodo successivo di “A0”:

```
cmpa.l   #0,a0
beq      error
movea.l  (a0),a1
cmpa.l   #0,a1
beq      error
```

```
movea.l  (a1),a1
move.l   a1,(a0)
```

- per inserire “A1” a seguito di “A0”:

```
cmpa.l   #0,a0
beq      error
cmpa.l   #0,a1
beq      error
movea.l  (a0),(a1)
move.l   a1,(a0)
```

- per posizionare A0 sul nodo che contiene D0, se esiste, nella lista di byte “A0”:

```
loop   cmpa.l   #0,a0
       beq      failure
       cmp.b    4(a0),d0
       beq      success
       movea.l  (a0),a0
       bra      loop
```

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