```
;calcola il numero dei numeri positivi e negativi di una tabella di memoria e
;input: r0 = indirizzo prima cella di memoria (che finisce con il valore zero)
;output: r0 = -1 se negativi > positivi
;       r0 = 1 se positivi > negativi
;       r0 = 0 se positivi = negativi

      .orig   x3000
      lea     r0,table
      jsr     pos_neg
blocca  brnzp   blocca
table   .fill   112
      .fill   98
      .fill   15
      .fill   -56
      .fill   0
pos_neg
      st      r1,savr1        ;salvo i registri
      st      r2,savr2
      st      r3,savr3
      and     r2,r2,#0        ;inizializzo r2 che userò per contare i positivi
      and     r3,r3,#0        ;inizializzo r3 che userò per contare i negativi
ciclo   ldr     r1,r0,#0        ;carico primo valore in r1
      brp     cont_pos        ;se positivo vai a conteggio positivi
      brn     cont_neg        ;se negativo vai a conteggio negativi
      brz     fine_tab        ;se nullo fine tabella
cont_pos
      add     r2,r2,#1        ;incremento di 1 r2
      brnzp   prox
cont_neg
      add     r3,r3,#-1       ;decremento di 1 r3
prox    add     r0,r0,#1        ;incremento r0 per leggere prox numero da tabella
      brnzp   ciclo
fine_tab
      and     r1,r1,#0        ;inizilizzo r1 per fare la somma tra positivi e negativi
      add     r1,r2,r3
      brn     magg_neg        ;se risultato negativo sono maggiori i negativi
      brp     magg_pos        ;se risultato positivo sono maggiori i positivi
      brz     uguali
magg_neg
      and     r0,r0,#0
      add     r0,r0,#-1
      brnzp   fine
magg_pos
      and     r0,r0,#0
      add     r0,r0,#1
```

```
        brnzp   fine
uguali  and     r0,r0,#0
fine    ld      r1,savr1
        ld      r2,savr2
        ld      r3,savr3
        ret
savr1   .blkw   1
savr2   .blkw   1
savr3   .blkw   1

        .end
```

```
; Scrivere un sottoprogramma che confronta due numeri in complemento a 2 da 16 bit
; e indica se sono concordi o discordi
; input: prim = primo numero
;        sec = secondo numero
; output: r1 = +1 concordi
;         r1 = -1 discordi

        .orig   x3000
        ld      r0,prim
        ld      r1,sec
        jsr     confronto
blocca  brnzp   blocca
prim    .fill   20
sec     .fill   15

confronto
        add     r0,r0,#0
        brn     negativ         ; se negativo salta, se positivo o nullo scende
        add     r1,r1,#0
        brn     discord         ; se discordi salta, altrimenti sono concordi
concord and     r1,r1,#0        ; azzeriamo r1 perchè dato che sono concordi dobbiamo inserire +1
        add     r1,r1,#1
        ret                     ; concordi - fine
negativ add     r1,r1,#0        ; per vedere che segno ha r1
        brn     concord         ; se anche r1 è negativo sono concordi
discord and     r1,r1,#0        ; azzeriamo r1 perchè dato che sono discordi dobbiamo inserire -1
        add     r1,r1,#-1

        .end
```

```
;dati due numeri in R0 e in R1, dire quale dei 2 è maggiore e se
;R0>R1 mettere in R2 = - 1
;R0<R1 mettere in R2 = + 1
;R0=R1 mettere in R2 = 0

      .ORIG   x3000
      LD      R0,PRIM_N
      LD      R1,SEC_N
      JSR     CONFRONTO
BLOCCA  BRNZP   BLOCCA
PRIM_N  .FILL   10
SEC_N   .FILL   5

CONFRONTO
      ST      R1,SAVR1
      ST      R2,SAVR2
      ST      R3,SAVR3        ;salviamo i registri che modificheremo
      ADD     R0,R0,#0        ;x vedere se è positivo, negativo, nullo
      BRN     NEGAT           ;se salta è negativo
      ADD     R1,R1,#0        ;vediamo se il secondo numero è posit, negat, nullo
      BRN     DISCORD         ;se salta sono discordi
      NOT     R1,R1
      ADD     R1,R1,#1
      ADD     R3,R0,R1        ;visto che sono entrambi positivi o nulli li confrontiamo
      BRP     R1_MIN          ;R0>R1
      BRZ     NULLO           ;R0=R1=0
      BRN     R1_MAGG
NEGAT   ADD     R1,R1,#0        ;qui R0<0 e vediamo che segno ha R1
      BRN     CON_NEG         ;R0 e R1 sono entrambi negativi
DISCORD ADD     R3,R0,R1        ;qui R0>=0 e R1<0
      BRP     R1_MIN          ;R0>R1
      BRN     R1_MAGG         ;R0<R1
      RET
R1_MAGG AND     R2,R2,#0        ;inizializzo R2
      ADD     R2,R2,#1        ;qui R1>R0 e quindi sommo 1 a R2
      RET
R1_MIN  AND     R2,R2,#0
      ADD     R2,R2,#-1       ;qui R0>R1 e quindi sommo -1 a R2
      RET
NULLO   AND     R2,R2,#0        ;qui R0=R1=0
      RET
CON_NEG NOT     R1,R1           ;qui entrambi negativi, facciamo il confronto
      ADD     R1,R1,#1
      ADD     R3,R0,R1
      BRN     R1_MAGG         ;R1>R0
```

```
    BRP    R1_MIN        ;R1<R0
    LD     R1,SAVR1
    LD     R2,SAVR2
    LD     R3,SAVR3
    RET
SAVR1  .BLKW  1
SAVR2  .BLKW  1
SAVR3  .BLKW  1

    .END
```

```
;contare numeri positivi e negativi
;input: r0 = indirizzo prima cella di memoria
;       r1 = indirizzo ultima cella di memoria
;output: r2 = conteggio numeri positivi
;        r3 = conteggio numeri negativi

        .orig   x3000
        lea     r0,table
        lea     r1,endtab
        jsr     neg_pos
blocca  brnzp   blocca
table   .fill   112
        .fill   27
        .fill   -10
        .fill   -2
        .fill   56
endtab  .fill   60
neg_pos
        st      r4,savr4        ;salvo registro
ciclo   ldr     r4,r0,#0        ;carico primo valore in r4
        brzp    positivi        ;se nullo o positivo vai a positivi
;qui negativo
        add     r3,r3,#1        ;incremento di 1 r3 che conta i negativi
        brnzp   confr_celle
positivi
        add     r2,r2,#1        ;incremento di 1 r2 che conta i positivi
confr_celle
        not     r4,r1
        add     r4,r4,#1
        add     r4,r4,r0        ;faccio il confronto tra r1 e r0
        brzp    fine            ;se r1>=r0 fine tabella
        add     r0,r0,#1        ;incremento r0 per leggere il nr. successivo
        brnzp   ciclo           ;salto incondizionato
fine    ld      r4,savr4        ;ripristino i registri
        ret
savr4   .blkw   1
        .end
```

```
;conteggio numeri positivi
;input: r0 = indirizzo prima cella di memoria
;       r1 = indirizzo ultima cella di memoria
;output: r0 = conteggio valori positivi

        .orig   x3000
        lea     r0,table
        lea     r1,endtable
        jsr     cont_pos
blocca  brnzp   blocca
table   .fill   -27
        .fill   -32
        .fill   0
        .fill   56
endtable
        .fill   -12
cont_pos
        st      r2,savr2
        st      r3,savr3
        st      r4,savr4
        and     r3,r3,#0
        ldr     r2,r0,#0
        brnz    ciclo
conteggio
        add     r3,r3,#1
ciclo   not     r4,r1
        add     r4,r4,#1
        add     r4,r4,r0
        brzp    fine
        add     r0,r0,#1
        ldr     r2,r0,#0
        brp     conteggio
        brnz    ciclo
fine    and     r0,r0,#0
        add     r0,r0,r3
        ld      r2,savr2
        ld      r3,savr3
        ld      r4,savr4
        ret
savr2   .blkw   1
savr3   .blkw   1
savr4   .blkw   1

        .end
```

```
;conta numeri doppi

        .orig   x3000
        trap    x23
        jsr     conta_doppie
blocca  brnzp   blocca

conta_doppie
        st      r2,savr2
        st      r3,savr3
        st      r4,savr4
        and     r4,r4,#0
        not     r1,r1
        add     r1,r1,#1
ciclo   add     r2,r0,r1
        brzp    fine
        ldr     r2,r0,#0
        ldr     r3,r0,#1
        not     r3,r3
        add     r3,r3,#1
        add     r3,r2,r3
        brnp    diversi
        add     r4,r4,#1
diversi add     r0,r0,#1
        brnzp   ciclo
fine    add     r0,r4,#0
        not     r1,r1
        add     r1,r1,#1
        ld      r2,savr2
        ld      r3,savr3
        ld      r4,savr4
        ret

savr2   .blkw   1
savr3   .blkw   1
savr4   .blkw   1

        .end
```

```
;contare quante volte due celle consecutive di un array contengono due numeri uguali
;input: r0 = indirizzo prima cella di memoria (che termina con il valore zero)
;output: r1 = conteggio numeri uguali celle consecutive

        .orig   x3000
        lea     r0,table
        jsr     conteggio
blocca  brnzp   blocca
table   .fill   12
        .fill   12
        .fill   -10
        .fill   -10
        .fill   9
        .fill   1
        .fill   1
        .fill   3
        .fill   -15
        .fill   -15
        .fill   0
conteggio
        st      r2,savr2        ;salvo i registri
        st      r3,savr3
        st      r4,savr4
        and     r2,r2,#0        ;inizializzo i registri che utilizzerò
        and     r3,r3,#0
        and     r4,r4,#0
ciclo   ldr     r2,r0,#0        ;carico in r2 il primo valore della tabella
        brz     fine            ;se è zero fine tabella
        not     r3,r3           ;faccio il confronto (anche se la prima volta in r3 non ho nessun valore)
        add     r3,r3,#1
        add     r4,r3,r2        ;metto il risultato della somma in r4
        brnp    continua        ;se il risultato è negativo o positivi leggo prox numero
        add     r1,r1,#1        ;qui il risultato è nullo quindi sono uguali e incremento di 1 r1
continua
        not     r3,r3           ;ripristino il segno di r3 che avevo cambiato per il confronto
        add     r3,r3,#1
        add     r0,r0,#1        ;incremento di uno r0 per leggere il numero successivo
        ldr     r3,r0,#0        ;carico in r3 il valore successivo
        brz     fine            ;se negativo fine tabella
        not     r3,r3           ;faccio il confronto
        add     r3,r3,#1
        add     r4,r3,r2
        brnp    continua2       ;se il risultato è negativo o positivi leggo prox numero
        add     r1,r1,#1        ;qui il risultato è nullo quindi sono uguali e incremento di 1 r1
continua2
```

```
        not     r3,r3           ;ripristino il segno di r3 che avevo cambiato per il confronto
        add     r3,r3,#1
        add     r0,r0,#1        ;incremento di 1 la tabella
        brnzp   ciclo           ;salto incondizionato a ciclo
fine    ld      r2,savr2
        ld      r3,savr3
        ld      r4,savr4
        ret
savr2   .blkw   1
savr3   .blkw   1
savr4   .blkw   1
        .end
```

```
;contare quanti numeri della tabella sono positivi e negativi e mettere la differenza in r2
;input: r0 = indirizzo prima cella di memoria
;       r1 = indirizzo ultima cella di memoria
;output: r2 = differenza conteggio positivi e negativi

        .orig   x3000
        lea     r0,table
        lea     r1,endtab
        jsr     differenza
blocca  brnzp   blocca
table   .fill   12
        .fill   5
        .fill   27
        .fill   3
        .fill   2
endtab  .fill   -9
differenza
        st      r3,savr3        ;ripristino i registri
        st      r4,savr4
        and     r3,r3,#0        ;inizializzo r3 per conteggio positivi
        and     r4,r4,#0        ;inizializzo r4 per conteggio negativi
        ldr     r2,r0,#0        ;carico primo valore
        brp     positivi        ;se positivo vai a positivi
        brn     negativi        ;se negativo vai a negativi
        brz     fine            ;se nullo fine
positivi
        add     r3,r3,#1        ;per i positivi incremento di 1 per ogni positivo
        brnzp   ciclo
negativi
        add     r4,r4,#-1       ;per i negativi decremento di 1 per ogni negativo
ciclo   and     r2,r2,#0        ;inizilizzo r2
        not     r2,r1
        add     r2,r2,#1
        add     r2,r2,r0        ;faccio confronto tra r0 e r1
        brzp    fine            ;se r1=>r0 fine tabella
        add     r0,r0,#1        ;incremento r0 per leggere prox numero in tabella
        and     r2,r2,#0
        ldr     r2,r0,#0        ;carico numero successivo
        brp     positivi        ;se positivo vai a positivi e incremento di 1
        brn     negativi        ;se negativo vai a negativi e decremento di 1
fine    and     r2,r2,#0        ;inizializzo r2
        add     r2,r3,r4        ;sommo r3 e r4 e metto così la differenza in r2
        ld      r3,savr3        ;ripristino i registri
        ld      r4,savr4
        ret
```

```
savr3   .blkw   1
savr4   .blkw   1

        .end
```

```
;scrivere un numero nella posizione corretta
;input: r0 = indirizzo prima cella di memoria
;       r1 = indirizzo ultima cella di memoria
;       r2 = un valore - facciamo 15
;output: r0 = indirizzo prima cella di memoria cn nuova sequenza
;        r1 = indirizzo ultima cella di memoria cn nuova sequenza
;        r2 = lo stesso valore - 15

        .orig   x3402
        lea     r0,table
        lea     r1,endtab
        and     r2,r2,#0
        add     r2,r2,#15
        jsr     ins_ord
blocca  brnzp   blocca
table   .fill   112
        .fill   27
        .fill   0
        .fill   -2
endtab  .fill   -56

ins_ord
        st      r0,savr0
        st      r2,savr2
        st      r3,savr3
ciclo1  ldr     r3,r0,#0        ;carico primo valore r0
        not     r3,r3
        add     r3,r3,#1
        add     r3,r3,r2        ;faccio il confronto per vedere se trovo la posizione di r2
        brp     ins_qui         ;se il risultato è positivo ho trovato la posizione di r2
;qui r2 <= r0
        add     r0,r0,#1
        not     r3,r1
        add     r3,r3,#1
        add     r3,r3,r0        ;faccio il confronto tra r0 e r1
        brnz    ciclo1
;qui r2 da inserire in ultima posizione
        add     r1,r1,#1
        str     r2,r1,#0        ;scrivo il valore di r2
        brnzp   fine
;qui r2 da inserire in array
ins_qui add     r1,r1,#1        ;incremento r1
ciclo2  ldr     r3,r0,#0        ;carico il valore di r0 in r3 per salvarlo dato ke dovrò sovrascriverci
        str     r2,r0,#0        ;scrivo r2 nella giusta posizione in tabella
        add     r2,r3,#0        ;metto in r2 il valore di r3 che poi scriverò in r0 nella giusta posizione
```

```
        add     r0,r0,#1        ;incremento r0
        not     r3,r1
        add     r3,r3,#1
        add     r3,r3,r0        ;faccio il confronto tra r0 e r1
        brnz    ciclo2          ;se negativo o nullo ciclo
fine    ld      r0,savr0
        ld      r2,savr2
        ld      r3,savr3
        ret
savr0   .blkw   1
savr2   .blkw   1
savr3   .blkw   1
        .end
```

```
;calcola valore maggiore in un array
;input: r0 = indirizzo prima cella di memoria
;       r1 = indirizzo ultima cella di memoria
;output: r0 = valore maggiore
;        r1 = indirizzo valore maggiore

        .orig   x3000
        lea     r0,table
        lea     r1,endtab
        jsr     maggiore
blocca  brnzp   blocca
table   .fill   89
        .fill   56
        .fill   -1252
        .fill   190
        .fill   1689
        .fill   -1223
endtab  .fill   -187
maggiore
        st      r2,savr2        ;salvo registri
        st      r3,savr3
        st      r4,savr4
        st      r4,savr5
        ldr     r2,r0,#0        ;metto il primo valore in r2
        brzp    primo_magg      ;se positivo o nullo vai a primo maggiore
        brn     negativi        ;se negativo vai a maggiore
primo_magg
        and     r3,r3,#0        ;inizializzo r3 che userò per inserire il valore massimo corrente
        add     r3,r3,r2        ;metto il valore trovato in r3
        and     r5,r5,#0        ;inizializzo r5 - lo uso per inserire l'indirizzo del massimo corrente
        add     r5,r5,r0        ;metto l'indirizzo del nuovo massimo in r5
negativi
        not     r4,r1
        add     r4,r4,#1
        add     r4,r4,r0        ;faccio il contronto tra r0 e r1
        brzp    fine            ;se r1>r0 è finita la tabella e vai a fine
        add     r0,r0,#1        ;incremento r0
        ldr     r2,r0,#0        ;metto il valore successivo in r2
        brz     negativi        ;se negativo ripeti il ciclo e vai a negativi
;qui confronto perchè entrambi positivi
        not     r2,r2           ;cambio segno a r2 per il confronto
        add     r2,r2,#1
        and     r4,r4,#0
        add     r4,r3,r2        ;sommo r2 con r3
        brzp    negativi        ;se r3 - r2<0 r3 rimane il massimo altrimenti trovato nuovo massimo
```

```
;qui nuovo max
      not    r2,r2          ;riporto r2 positivo
      add    r2,r2,#1
      and    r3,r3,#0       ;inizializzo r3
      add    r3,r3,r2       ;metto il nuovo massimo in r3
      and    r5,r5,#0       ;inizializzo r5
      add    r5,r5,r0       ;metto l'indirizzo del nuovo massimo in r5
      brnzp  negativi       ;ciclo incondizionato salta a negativi
fine  and    r0,r0,#0       ;inizializzo r0
      and    r1,r1,#0       ;inizializzo r1
      add    r0,r0,r3       ;metto il massimo valore in r0
      add    r1,r1,r5       ;metto l'indirizzo del massimo valore in r1
      ld     r2,savr2       ;ripristino i registri
      ld     r3,savr3
      ld     r4,savr4
      ld     r4,savr5
      ret
savr2  .blkw  1
savr3  .blkw  1
savr4  .blkw  1
savr5  .blkw  1

      .end
```

```
; calcolare minimo e massimo tra un array di valori
; input: r0 = indirizzo prima cella array
;        r1 = indirizzo ultima cella array
; output: r0 = valore minimo
; output: r1 = valore maggiore

        .orig   x3000
        lea     r0,table
        lea     r1,endtab
        jsr     magg_min
blocca  brnzp   blocca
table   .fill   -10
        .fill   -34
        .fill   50
        .fill   -20
        .fill   0
        .fill   89
        .fill   63
endtab  .fill   -68
magg_min
        st      r2,savr2
        st      r3,savr3
        st      r4,savr4
        st      r5,savr5
        ldr     r2,r0,#0        ;metto primo valore in r2
        brz     primo_min               ;se nullo o negativo è il primo minore
        brp     primo_magg      ;se positivo è il primo maggiore
primo_min
        and     r3,r3,#0        ;inizializzo r3 che usiamo come cella del minore
        add     r3,r3,r2
        brnzp   ciclo
primo_magg
        and     r4,r4,#0        ;inizializzo r4 che usiamo come cella del maggiore
        add     r4,r4,r2
        brnzp   ciclo           ;somma a r4 il primo maggiore trovato
vecchio_magg
        not     r2,r2
        add     r2,r2,#1
        add     r4,r4,r2
        brnzp   ciclo
vecchio_min
        not     r2,r2
        add     r2,r2,#1
        add     r3,r3,r2
        brnzp   ciclo
```

```
nuovo_magg
        not     r2,r2
        add     r2,r2,#1
        add     r4,r2,#0
        brnzp   ciclo
nuovo_min
        not     r2,r2
        add     r2,r2,#1
        and     r3,r3,#0
        add     r3,r2,#0
ciclo   not     r5,r1
        add     r5,r5,#1
        add     r5,r5,r0        ;confronto r5(=r1) con r0
        brzp    fine            ;se r1>r0 salta a fine e quindi fine array
        add     r0,r0,#1        ;incremento r0 per leggere prox cella
        ldr     r2,r0,#0
        brp     confr_pos       ;se positivo lo confrontiamo con quello gia trovato
        brn     confr_neg       ;se nullo o negativo lo confrontiamo con quello gia trovato
confr_pos
        not     r2,r2
        add     r2,r2,#1
        add     r4,r4,r2        ;r2>r4?
        brp     vecchio_magg
        brn     nuovo_magg
confr_neg
        not     r2,r2
        add     r2,r2,#1
        add     r3,r3,r2        ;r2>r3?
        brn     vecchio_min
        brp     nuovo_min
fine    and     r0,r0,#0
        add     r0,r0,r3
        and     r1,r1,#0
        add     r1,r1,r4
        ld      r2,savr2
        ld      r3,savr3
        ld      r4,savr4
        ld      r5,savr5
        ret
savr2   .blkw   1
savr3   .blkw   1
savr4   .blkw   1
savr5   .blkw   1

        .end
```

```
;riempire la tabella vuota di r1 in modo tale che l'elemento di r1 di posto i sia la somma in complemento
;a due degli elementi di posto i e i+1 della tabellla in ingresso r0 (che termina con il valore zero)
;cioè se r0 contiene i valori 5,-3,3,4,-7,9,-9,0 il sottoprogramma deve riempire la tabella vuota con i valori:
;2,0,7,-3,2,0,9 - cioè di r0:5+(-3) = 2, -3+3 = 0, ecc...
;
;input: r0 = indirizzo prima cella di una tabella di memoria piena
;       r1 = indirizzo prima cella di una tabella di memoria vuota
;output: r1 = valori i e i+1 di r0

        .orig   x3000
        lea     r0,table1
        lea     r1,table2
        jsr     riempi_tabella
blocca  brnzp   blocca
table1  .fill   5
        .fill   -3
        .fill   3
        .fill   4
        .fill   -7
        .fill   9
        .fill   -9
        .fill   0
table2  .fill   0
        .fill   0
        .fill   0
        .fill   0
        .fill   0
        .fill   0
        .fill   0
riempi_tabella
        st      r2,savr2
        st      r3,savr3
        st      r4,savr4        ;salvo i registri
        and     r2,r2,#0
        and     r3,r3,#0
        and     r4,r4,#0        ;inizializzo registri
        ldr     r3,r0,#0        ;primo valore di r0 in r3
        brz     finetabella     ;se nullo fine tabella
        add     r0,r0,#1        ;incremento r1
        ldr     r2,r0,#0        ;secondo valore in r2
        brz     finetabella
ciclo   add     r4,r2,r3        ;sommo i primi due numeri della tabella e li metto in r4
        str     r4,r1,#0        ;scrivo nella prima cella di r1 la somma dei primi due numeri di r0
        add     r1,r1,#1        ;incremento di 1 r1 per posizionarmi sulla cella successiva
        add     r0,r0,#1        ;incremento r0
        ldr     r3,r0,#0        ;carico prox numero in r3
        brz     finetabella
```

```
        add     r4,r2,r3
        str     r4,r1,#0
        add     r1,r1,#1
        add     r0,r0,#1
        ldr     r2,r0,#0
        brnp    ciclo
finetabella
        add     r4,r2,r3
        str     r4,r1,#0
        ld      r2,savr2
        ld      r3,savr3
        ld      r4,savr4
        ret
savr2   .blkw   1
savr3   .blkw   1
savr4   .blkw   1
        .end
```

```
;trova minimo in una tabella di memoria
;input: r0 = indirizzo prima cella di memoria
;       r1 = indirizzo ultima cella di memoria
;output: r0 = valore minimo
;        r1 = indirizzo valore minimo

        .orig   x3000
        lea     r0,table
        lea     r1,endtab
        jsr     valore_min
blocca  brnzp   blocca
table   .fill   380
        .fill   120
        .fill   50
        .fill   0
        .fill   -27
endtab  .fill   -87
valore_min
        st      r2,savr2
        st      r3,savr3
        st      r4,savr4
        st      r5,savr5
        ldr     r2,r0,#0        ;carico primo valore tabella in r2
        brn     primo_min       ;trovato primo minimo
positivi                        ;qui il valore è positivo
        not     r3,r1
        add     r3,r3,#1
        add     r3,r3,r0        ;confronto se r1>=r0
        brzp    fine            ;se r0>=r1 fine tabella

        add     r0,r0,#1        ;incremento r0
        ldr     r2,r0,#0        ;leggo prox numero tabella
        brzp    positivi
primo_min                       ;qua vi è il primo minimo
        and     r4,r4,#0        ;inizializzo r4 che userò per valore minimo
        add     r4,r4,r2        ;metto il primo minimo in r4
        and     r5,r5,#0
        add     r5,r0,#0        ;metto l'indirizzo del primo minimo in r3
ciclo   not     r3,r1
        add     r3,r3,#1
        add     r3,r3,r0
        brzp    fine
        add     r0,r0,#1
        ldr     r2,r0,#0
        brp     ciclo
```

```
                    ;qui è negativo e allora faccio il confronto con il minimo gia trovato
      not    r2,r2
      add    r2,r2,#1
      add    r4,r4,r2
      brp    nuovo_min      ;se r2>r4 trovato nuovo minimo
      brnz   ciclo
nuovo_min
      not    r2,r2          ;ricambio il segno al nuovo minimo per farlo ritornare negativo
      add    r2,r2,#1
      and    r4,r4,#0       ;inizializzo r4 per inserire nuovo minimo
      add    r4,r4,r2       ;sommo il nuovo minimo a r4
      and    r5,r5,#0       ;inizializzo r4 per inserire nuovo inidirizzo valore minimo
      add    r5,r0,#0       ;metto nuovo indirizzo in r4
      brnzp  ciclo
fine  and    r0,r0,#0
      add    r0,r0,r4       ;inserisco il valore minimo in r0
      and    r1,r1,#0
      add    r1,r5,#0       ;metto indirizzo nuovo min in r1
      ld     r2,savr2
      ld     r3,savr3
      ld     r4,savr4
      ld     r5,savr5
      ret
savr2  .blkw  1
savr3  .blkw  1
savr4  .blkw  1
savr5  .blkw  1

      .end
```

```
; Creare un sottoprogramma che dati nei registri R0 e R1 due numeri, mette in
; R2 il valore contenuto in R0 per R1

        .orig   x3001
        ld      r0,primo_n
        ld      r1,sec_n
        jsr     prodotto
blocca  brnzp   blocca
primo_n .fill   6
sec_n   .fill   8
prodotto
        st      r0,savr0
        and     r2,r2,#0        ;inizializzo r2
ciclo   add     r2,r2,r1        ;sommo 0+4 e lo metto in r2
        add     r0,r0,#-1       ;decremento r1
        brnp    ciclo
        ld      r0,savr0        ;ripristino r0
        ret
savr0   .blkw   1

        .end
```

```
;calcolare la somma di una sequenza di numeri che termina con il valore zero
;input: r0 = indirizzo prima cella di memori
;output: r1 = la somma

        .orig   x3000
        lea     r0,table
        jsr     somma
blocca  brnzp   blocca
table   .fill   100
        .fill   73
        .fill   -52
        .fill   -12
        .fill   23
        .fill   87
        .fill   0
somma   st      r2,savr2
        and     r1,r1,#0
ciclo   ldr     r2,r0,#0
        brz     fine
        add     r1,r1,r2
        add     r0,r0,#1
        brnzp   ciclo
fine    ld      r2,savr2
        ret
savr2   .blkw   1

        .end
```

```
;programma di ricerca del massimo valore assoluto in un array di interi con segno
; input RO = indirizzo inizio array
; outpur R0 = massimo valore assoluto

        .orig   x3000
        lea     r0,table        ;carico indirizzo array
        jsr     trova_max
blocca  brnzp   blocca
table   .fill   112
        .fill   -27
        .fill   -1232
        .fill   450
        .fill   15
        .fill   0
trova_max
        st      r1,savr1
        st      r2,savr2
        st      r3,savr3
        and     r2,r2,#0        ;inizializza r2 dove inseriremo il valore max corrente
        ldr     r1,r0,#0        ;carica il primo valore dell'array in r1
        brz     fine            ;se il valore è gia zero il sottoprogramma finisce e va a fine
        brp     primo_max       ;se positivo il valore è il primo valore max assoluto
        not     r1,r1
        add     r1,r1,#1        ;se negativo lo facciamo diventare positivo e continua in primo_max
primo_max
        add     r2,r1,#0        ;mettiamo il primo numero max in r2
ciclo   add     r0,r0,#1        ;incrementiamo la nostra tabella
        ldr     r1,r0,#0        ;carichiamo il valore di r1 in r0
        brz     fine
        brn     confronto       ;se il secondo numero è negativo possiamo confrontare
        not     r1,r1
        add     r1,r1,#1        ;se positivo gli cambiamo il segno x confrontare
confronto
        add     r3,r2,r1        ;somma i due valori
        brzp    ciclo           ;se il numero è negativo r2-r1<0 vuol dire che è stato trovato un nuovo max
                                ;altrimento se r2-r1>0 significa che r2 è ancora il valore max
        not     r2,r1
        add     r2,r2,#1        ;mettiamo il nuovo max in r2 con il segno positivo
        brnzp   ciclo
fine    add     r0,r2,#0        ;infine carico il valore di r2 con il max assoluto in r0
        ld      r1,savr1
        ld      r2,savr2
        ld      r3,savr3
        ret
savr1   .blkw   1
```

```
savr2   .blkw   1
savr3   .blkw   1

        .end
```

```
;trova il valore minimo dell'array e mettilo in r0
;input r0 = inizio array
;output r0 = valore minimo

        .orig   x3000
        lea     r0,table
        jsr     trova_min
blocca  brnzp   blocca
table   .fill   1560
        .fill   470
        .fill   -927
        .fill   482
        .fill   3765
        .fill   0
trova_min
        st      r1,savr1
        st      r2,savr2
        st      r3,savr3
        and     r2,r2,#0        ;inizializzo r2
        ldr     r1,r0,x0        ;incremento r0
        brp     positivi        ;se il primo numero è positivo salta ai positivi
        brz     fine            ;se il primo numero è nullo salta a fine
        brn     primo_min       ;se il primo numero è negativo salta a primo_min
positivi
        add     r0,r0,x1        ;qui r0>0 ora incremento r0
        ldr     r1,r0,#0        ;leggo il secondo numero
        brp     positivi        ;se il secondo numero r0 è positivo salta ancora ai positivi
        brz     fine            ;se il secondo numero r0 è nullo salta a fine
        brn     primo_min       ;se il secondo numero r0 è negativo è il primo numero negativo
primo_min
        add     r2,r1,#0        ;metto primo minimo in r2
ciclo   add     r0,r0,x1        ;incremento di nuovo r0
        ldr     r1,r0,#0        ;scrivo prox numero in r1
        brz     fine            ;se nullo fine
        brp     positivi        ;se positivo salta a positivi
        brn     confronto       ;se negativo confronta i due negativi
confronto
        not     r1,r1           ;cambio segno al secondo negativo
        add     r1,r1,#1        ;sommo 1
        add     r3,r2,r1        ;metto il risultato in r3
        brz     fine            ;se il risultato è nullo salta a fine
        brn     ciclo           ;se il risultato è negativo r0>r1 quindi è ancora lo stesso minimo e salta a ciclo
        brp     nuovo_min       ;se il risultato è positivo salta a nuovo minimo
nuovo_min
        not     r1,r1           ;poichè il secondo minimo è positivo per via del confronto lo facciamo ritornare negativo
        add     r1,r1,#1
        add     r2,r1,#0        ;salviamo il nuovo minimo in r2
        brnzp   ciclo
fine    add     r0,r2,#0
```

```
        ld      r1,savr1
        ld      r2,savr2
        ld      r3,savr3
        ret
savr1   .blkw   1
savr2   .blkw   1
savr3   .blkw   1

        .end
```

```
;trovare e inserire la posizione di un numero in una tabella di memoria
;input: r0 = indirizzo prima cella di memoria
;       r1 = 15
;output r0 = indirizzo prima cella di memoria cn nuova sequenza
;       r1 = 15


        .orig   x3000
        lea     r0,table
        add     r1,r1,#15
        jsr     trova_pos
blocca  brnzp   blocca
table   .fill   1100
        .fill   850
        .fill   125
        .fill   34
        .fill   11
        .fill   0
trova_pos
        st      r0,savr0
        st      r2,savr2
        st      r3,savr3
        and     r2,r2,#0
        and     r3,r3,#0
ciclo   ldr     r2,r0,#0        ;carico primo numero
        brz     fine
        not     r3,r2
        add     r3,r3,#1
        add     r3,r3,r1        ;confronto con r1
        brp     trovato
        add     r0,r0,#1
        brnzp   ciclo
trovato add     r3,r2,#0
        str     r1,r0,#0
        add     r1,r3,#0
        add     r0,r0,#1
        ldr     r2,r0,#0
        brnp    trovato
fine    str     r1,r0,#0
        str     r2,r0,#1
        ld      r0,savr0
        ld      r2,savr2
        ld      r3,savr3
        ret
savr0   .blkw   1
```

```
savr2   .blkw   1
savr3   .blkw   1
        .end
```

```
;calcola positivi

        .orig   x3000
        lea     r0,table
        lea     r1,endtab
        jsr     positivi
blocca  brnzp   blocca
table   .fill   380
        .fill   120
        .fill   0
        .fill   50
        .fill   -27
endtab  .fill   -87
positivi
        st      r2,savr2
        st      r3,savr3
        st      r4,savr4
        and     r3,r3,#0
ciclo   ldr     r2,r0,#0
        brnz    noincr
        add     r3,r3,r2
noincr  add     r0,r0,#1
        not     r4,r1
        add     r4,r4,#1
        add     r4,r0,r4
        brnz    ciclo
        and     r0,r0,#0
        add     r0,r0,r3
        ld      r2,savr2
        ld      r3,savr3
        ld      r4,savr4
        ret
savr2   .blkw   1
savr3   .blkw   1
savr4   .blkw   1

        .end
```

```
;programma di inversione dell'ordinamento di un array (da decrescente a crescente)
;
; input RO = indirizzo inizio array
;       R1 = indirizzo fine array (deve essere R1>=R0)
;
; output nessuno

        .ORIG   x3000
        LEA     R0,PR_CELL
        LEA     R1,SC_CELL
        JSR     CAMB_ORD
BLOCCA  BRNZP   BLOCCA
PR_CELL .FILL   112
        .FILL   27
        .FILL   15
        .FILL   0
        .FILL   -2
        .FILL   -56
SC_CELL .FILL   -56

CAMB_ORD
        ST      R2,SAVR2        ;salva registri utilizzati
        ST      R3,SAVR3
CICLO   LDR     R2,R0,#0        ;carica contenuto di r0 in r2
        LDR     R3,R1,#0        ;carica contenuto di r1 in r3
        STR     R2,R1,#0        ;inverte l'ordine mettendo r1 in r2
        STR     R3,R0,#0        ;inverte l'ordine mettendo r0 in r3
        ADD     R0,R0,#1        ;incrementa r0
        ADD     R1,R1,-1        ;decrementa r1
        NOT     R2,R1           ;confronta i due numeri se R1>=R0
        ADD     R2,R2,#1
        ADD     R2,R0,R1
        BRN     CICLO
        LD      R2,SAVR2
        LD      R3,SAVR3
        RET

SAVR2   .BLKW   1
SAVR3   .BLKW   1

        .END
```