Esercizio di programmazione

sino a 12 punti – è possibile consultare solamente il foglio consegnato con l'instruction set MIPS - tempo: 60 minuti

La notazione polacca inversa permette di scrivere formule matematiche senza utilizzare parentesi e regole di precedenza degli operatori. Quando si incontra un operatore all'interno dell'espressione matematica, questo si applica ai due operandi precedenti. Ad esempio, l'espressione 18 + 25 * (10 – 7) – 13 rappresentata con la notazione polacca inversa diventa 18 25 10 7 - * + 13 –

Si vuole realizzare un programma in linguaggio MIPS per calcolare il valore di un'espressione in notazione polacca inversa. L'espressione è memorizzata in un array di word, in cui gli elementi sono così codificati:

- se il primo bit è 0, l'elemento dell'array è un numero positivo nell'intervallo $(0, 2^{31} – 1)$
- se il primo bit è 1, l'elemento dell'array è un operatore (+, -, *, /). Una possibile codifica è la seguente:
  somma= -1; sottrazione = -2; moltiplicazione = -3; divisione = -4

Si supponga che la sintassi dell'espressione in notazione polacca inversa sia corretta e che nel calcolo del risultato non vi sia overflow. Si scriva una procedura `calcolaPolaccaInversa` che riceva in input come primo parametro l'indirizzo dell'array contenente l'espressione e come secondo parametro la lunghezza dell'array. La procedura restituisce il valore dell'espressione.

La procedura scandisce ogni elemento dell'array. Se è un operando, ne fa il push nello stack. Se è un operatore, preleva i due elementi in cima allo stack e chiama la procedura `eseguiOperazione` per ottenere il valore dell'operazione; questo valore è poi inserito in cima allo stack. La procedura `eseguiOperazione` riceve in input l'operatore, il primo operando, il secondo operando; restituisce il valore dell'operazione.

Tutte le procedure devono essere conformi allo standard e alle specifiche per quanto riguarda il passaggio dei parametri in input, del valore di ritorno e dei registri da preservare.

Di seguito un esempio di programma chiamante e della procedura `eseguiOperazione`:

```
      .data
espressione:      .word 18, 25, 10, 7, -2, -3, -1, 13, -2
tabella:          .word somma, sottrazione, moltiplicazione, divisione
      .text
      .globl main
      .ent main
main: subu $sp, $sp, 4
      sw $ra, ($sp)
      la $a0, espressione
      li $a1, 9
      jal calcolaPolaccaInversa

      lw $ra, ($sp)
      addu $sp, $sp, 4
      jr $ra
      .end main

eseguiOperazione:
      subu $t0, $zero, $a0
      subu $t0, $t0, 1
      sll $t0, $t0, 2
      lw $t1, tabella($t0)
      jr $t1
somma: addu $v0, $a1, $a2
      b fine
sottrazione: subu $v0, $a1, $a2
      b fine
moltiplicazione: mulou $v0, $a1, $a2
      b fine
divisione: divu $v0, $a1, $a2
      b fine
fine: jr $ra
```

Soluzione proposta

```
calcolaPolaccaInversa:
             subu $sp, $sp, 4
             sw $ra, ($sp)

             move $t5, $a0
             move $t6, $a1

ciclo:       beqz $t6, esci      # per ipotesi la lunghezza dell'array è >0
             lw $a0, ($t5)
             lb $t9, ($t5)
             bgez $a0, push

             lw $a1, 4($sp)
             lw $a2,  ($sp)
             addu $sp, 8
             jal eseguiOperazione
             subu $sp, 4
             sw $v0, ($sp)
             b  next

push:        subu $sp, $sp, 4
             sw $a0, ($sp)

next:        addu $t5, 4
             subu $t6, 1
             b ciclo

esci:        addu $sp, 4

             lw $ra, ($sp)
             addu $sp, $sp, 4
             jr $ra
```