14_03 Primo appello estivo

Scrivere un sottoprogramma per un microcontrollore Atmel della famiglia AVR XMEGA in grado di eseguire l'operazione di "modulo 7" su un numero intero senza segno rappresentato su 8 byte nella memoria estesa del microcontrollore. I byte sono disposti consecutivamente a partire dai meno significativi, a partire dall'indirizzo contenuto in `X`. Il risultato deve essere lasciato nel registro `R16`. Il dato di partenza può essere distrutto.

```
/* Nota: l'operazione di modulo 7 su un numero binario si attua ripetendo ricorsivamente la
somma del dato diviso 8 col suo resto, a cui si toglie 7 se maggiore di 7, fino a quando il
risultato è un numero minore di 7.
*/
sum8:               //somma R16 al numero (X) da 8 byte
   push R16
   push R17
   push R18
   ldi R18,8
   s1:
      ld R17,X
      adc R17,R16
      st X+,R17
      ldi R16,0     //non tocca i flag!
      dec R18
   brne s1
   sbiw XH:XL,8     //ripristina il puntatore
   pop R18
   pop R17
   pop R16
ret

div2:               //divide il numero (X) per 2
   push R17
   push R18
   adiw XH:XL,8     //posiziona il puntatore alla fine del numero
   ldi R18,8
   clc              //azzera il carry per la prima ror
   d1:
      ld R17,X
      ror R17
      st -X,R17     //rimette il byte del numero dimezzato
      dec R18
   brne d1
   pop R18
   pop R17
ret

div8:                //divide il numero (X) per 8
   rcall div2
   rcall div2
```

```
   rcall div2
ret

tst8:               //vede se il numero (X) è nullo
   push R16
   push R17
   push R18
   clr R16          //lascia R16 nullo se tutti i byte sono nulli
   ldi R18,8
   t1:
      ld R17,X+
      tst R17
      breq t2
         inc R16    //se un byte non è nullo rende R16 diverso da 0
      t2:
      dec R18
   brne t1
   sbiw XH:XL,8     //ripristina il puntatore
   tst R16          //lascia il risultato nel flag Z
   pop R18
   pop R17
   pop R16
ret

mod7:                //questa è la subroutine richiesta
   clr R16           //inizializza R16 a 0
   loop:
      rcall sum8     //somma R16
      ld R16,X
      andi R16,0x07  //isola una terna binaria
      cpi R16,7      //esegue mod 7
      brlo m1
         subi R16,7  //sottrae 7 se >= a 7
      m1:
      rcall div8     //divide per 8
      rcall tst8     //se il numero è diverso da 0 ripete
   brne loop
ret
```

```
/* Si può arrivare a una soluzione meno ottimizzata rispetto al tempo di esecuzione ma più
semplice lavorando a livello di byte, osservando che 256 modulo 7 fa 4.
*/
mod7:
   push R17
   push R18
   adiw XH:XL,8        // Si sposta in fondo al numero
   ldi r18,8
   loop:
      ld R16,-X       //carica, a partire dal più significativo
      rcall mod7byte  //esegue modulo 7
      add R16,R17     //somma il residuo dovuto alla parte più significativa, al max 24
      rcall mod7byte  //si riporta ancora tra 0 e 6
      mov R17,R16     //salva per la prossima iterazione
      lsl R17
      lsl R17         //moltiplica per 4, cioè per 256 mod 7
      dec R18
   brne loop
   pop R18
   pop R17
ret

mod7byte:             //esegue r16 mod 7
subi r16,7
brcc mod7byte
subi r16,-7
ret
```