Soluzione della seconda parte del compito di Calcolatori del 15/01/2020

Si tratta di determinare le lunghezze dell'antiperiodo e del periodo (chiamate rispettivamente h e k) dello sviluppo binario di una frazione p/q, con $0 < p < q$ interi. Usando l'algoritmo dato con i numeri p=17, q=28, si trova h=2 e k=3. Infatti, la sequenza ausiliaria $r(t)$, con t=1,2,… (max $t = q$) assume i seguenti valori:

$r(1)$ = 17
$r(2)$ = 6 = 17*2-**1***28
$r(3)$ = 12 = 6*2-**0***28
$r(4)$ = 24 = 12*2-**0***28
$r(5)$ = 20 = 24*2-**1***28
$r(6)$ = 12 = 20*2-**1***28
…

(Nota: I valori $b(t)$ in grassetto sono quelli dello sviluppo binario di 17/28, che andavano calcolati nella prima parte del compito ma che non servono per la soluzione della seconda parte.) Il primo valore ripetuto della sequenza è $12=r(3)=r(6)$, dopodiché la sequenza prosegue periodicamente con $r(7)$=24, r(8)=20 etc. Il periodo è lungo k = 3, avendosi $r(t+k) = r(t)$ per ogni $t \geq t'(12)$, dove abbiamo denotato con $t'(r)$ il valore di t nel quale viene calcolato per la prima volta il valore di r. Detto $t''(r)$ il valore di t corrispondente alla prima ripetizione di r, si ha dunque che $k = t''(r) – t'(r)$ per tutti gli r del periodo, e in particolare per il primo r del periodo, ossia 12:

$$k = t''(12) – t'(12) = 6 – 3 = 3$$

La lunghezza dell'antiperiodo sarà il valore di t che precede $t'(12)$=3, ossia

$$h = t'(12) – 1 = 3 – 1 = 2$$

Nel seguito vengono riportati due programmi in assembly 8086 che risolvono il problema.

Programma 1

Questo primo programma si basa sul riempimento di due vettori di memoria:

- `rsequence`, che contiene i successivi valori di r(t) calcolati al crescere di t;
- `rvisited`, vettore di flags che tiene traccia (flag=1) di quali valori di *r(t)* siano stati già calcolati, e quali no (flag=0). Questo vettore è inizializzato con 0. Il flag *t*-simo verrà scritto nella posizione *r(t)*-sima del vettore.

Alla prima ripetizione di un *r(t'')*, rilevata trovando un valore 1 nella posizione *r(t')* del vettore `rvisited`, i valori di *h* e *k* vengono ricavati scorrendo all'indietro il vettore `rsequence` alla ricerca della posizione *t'(r)* in cui si trova il valore ripetuto.
NOTA: una versione molto meno efficiente di questo programma (che avrebbe l'unico vantaggio di non richiedere l'utilizzo di un vettore di flags) potrebbe, ad ogni *t*, ricercare il valore di *r(t)* appena calcolato nei primi *t*-1 valori del vettore `rsequence`.

```
;---------------------------------------DATI-------------------------------------------------------

p equ 17                        ; costanti < 256, che possono essere usate come variabili sia di tipo byte che word
q equ 28                        ; (vedi seguito: mov bl,p (byte) e cmp bx,q (word))

rsequence db q dup (?)          ; vettore (di interi positivi) che contiene gli r(t) via via calcolati (N.B. max t = q)
rvisited  db q dup (0)          ; vettore di flags che tiene traccia di tutti gli r(t) già calcolati (1: calcolato, 0: non calcolato)


;---------------------------------------CODICE-----------------------------------------------------

initialization:
      mov si, offset rsequence      ; rsequence è riempito incrementandone via via il puntatore (AM = ind. di registro, con registro si)
      mov cl,1                      ; contatore di iterazioni (= numero di r(t) calcolati) eseguite
      mov bl,p                      ; p è il primo valore della sequenza r(t)
      mov [si],bl                   ; p viene posto nel primo elemento del vettore di memoria rsequence
      xor bh,bh                     ; poiché r(t) sta sempre in un byte, si ha r(t) = bl ma anche r(t) = bx (se si pone bh a 0)
      mov rvisited[bx],1            ; si avanza in questo vettore con AM = base & indice. N.B.: l'indice e' il valore corrente di r(t)

flippercycle:
      shl bx,1                      ; r = 2 r
      cmp bx,q                      ; if r ≥ q then r = r-q
      jl termination_check
      sub bx,q
```

```
termination_check:
      mov dh, rvisited[bx]          ; se si trova un flag=0, allora si va ad update, altrimenti c'è una ripetizione e si salta a backtrack
      cmp dh,0
      jne backtrack

update:
      inc cl
      inc si
      mov rvisited[bx],1            ; si scrive il flag 1 nella posizione r(t) di rvisited
      mov [si],bl                   ; si scrive r(t) nel vettore rsequence
      jmp flippercycle

backtrack:                          ; se siamo qui allora cl=h+k, bl = primo valore di r(t) ripetuto e si = offset rsequence + h+k – 1

       mov ch,cl
countdown:
      mov dl,[si]
      cmp bl,dl                     ; trovata la prima istanza di r(t'') nel vettore rsequence? In tal caso è ch = t'(r) = h+1
      je terminate
      dec si
      dec ch
      jnz countdown

terminate:
      sub ch,1
      mov al,ch                     ; qui in al c'è h = ch - 1
      sub cl,al
      mov ah,cl                     ; qui in ah c'è k = cl - h
```

Programma 2

Versione più breve ed efficiente della precedente, che non usa il vettore `rsequence`, e pone in `rvisited` (qui ribattezzato `rvisited_t`) l'indice *t* relativo a *r(t)* anziché un semplice flag binario. Ciò arricchisce l'informazione contenuta nel vettore, e consente di calcolare *h* e *k* alla prima ripetizione del valore *r(t')*, senza dover eseguire la procedura di backtracking usata prima, ma direttamente confrontando l'indice attuale *t''(r)* con l'indice *t'(r)* trovato in memoria.

```
;---------------------------------------DATI-------------------------------------------------------
p equ 17
q equ 28

rvisited_t   db q dup (0)      ; vettore di indici che tiene traccia di tutti gli r(t) calcolati (t: calcolato, 0: non calcolato)

;---------------------------------------CODICE-----------------------------------------------------
initialization:
      mov cl,1
      mov bl,p
      xor bh,bh
      mov rvisited_t[bx],1

flippercycle:
      shl bx,1
      cmp bx,q
      jl termination_check
      sub bx,q

termination_check:
      mov dh, rvisited_t[bx]  ; se si trova un valore 0, allora si va ad update, altrimenti c'è una ripetizione e si salta a terminate
      cmp dh,0
      jne terminate

update:
      inc cl
      mov rvisited_t[bx],cl   ; si scrive il valore t nella posizione r(t) di rvisited_t
      jmp flippercycle

terminate:                    ; se siamo qui allora cl=h+k e dh=h+1
      sub dh,1
      mov al,dh               ; qui in al c'è  h = dh - 1
      sub cl,al
      mov ah,cl               ; qui in ah c'è  k = cl - h
```