```
TITLE CodiceFiscale: per esame 1/7/2009

comment *
Costruzione della prima parte (11 caratteri) del codice fiscale.
N.B. Anziche' nello stack (come richiesto nel compito), qui i
caratteri calcolati sono posti nel segmento dati e stampati a
video insieme ai dati in input.
*

;-----------------------------------------------------------------
; Definizione costanti
CR EQU 13                       ; carriage return
LF EQU 10                       ; line feed
DOLLAR EQU '$'
max_len equ 32                  ; nome e cognome possono avere al max questa lunghezza

;-----------------------------------------------------------------
;    M  A  C  R  O
;-----------------------------------------------------------------

display macro xxxx                  ; N.B. ogni stringa deve terminare con '$'
        push dx
        push ax
        mov dx,offset xxxx
        mov ah,9
        int 21h
        pop ax
        pop dx
endm

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
separa_consonanti_vocali macro stringa,stringa_len
local ciclostringa,consonante,vocale,continua
        push bx
        push cx
        mov cx,stringa_len
        xor bx,bx
        mov n_consonanti,0
        mov n_vocali,0
ciclostringa:
        mov al,stringa[bx]
        and al,11011111B    ; si rende sempre maiuscolo il carattere
('A'=41h,'a'=61h)
        cmp al,'A'
        je vocale
        cmp al,'E'
        je vocale
        cmp al,'I'
        je vocale
        cmp al,'O'
        je vocale
        cmp al,'U'
        je vocale
consonante:
        mov di,n_consonanti
        mov consbuf[di],al
        inc n_consonanti
        jmp continua
vocale:
        mov di,n_vocali
        mov vocbuf[di],al
        inc n_vocali
continua:
        inc bx
        loop ciclostringa
        pop cx
        pop bx
endm

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
riempi_consonanti_vocali macro offset
local riempi_consonanti,check_addendum_vocali,riempi_vocali,next
        mov cx,n_consonanti
```

```
        xor bx,bx
riempi_consonanti:
        mov al,consbuf[bx]
        mov offset[bx],al
        inc bx
        loop riempi_consonanti
check_addendum_vocali:
        cmp n_consonanti,3
        je next
        mov cx,3
        sub cx,n_consonanti
        xor bx,bx
        mov di,n_consonanti
riempi_vocali:
        mov al,vocbuf[bx]
        mov offset[di],al
        inc bx
        inc di
        loop riempi_vocali
next:
endm

;-----------------------------------------------------------------
;
PILA SEGMENT STACK 'STACK'     ; definizione del segmento di stack
      DB      64 DUP('STACK')  ; lo stack e' riempito con la stringa 'stack'
                               ; per identificarlo meglio in fase di debug
PILA ENDS

;-----------------------------------------------------------------
;
DATI SEGMENT PUBLIC 'DATA'          ; definizione del segmento dati

NOME         db "Rio"
NOME_len     equ $-NOME
             db DOLLAR              ; terminazione: serve per la stampa a video
COGNOME      db "Bo"
COGNOME_len  equ $-COGNOME
             db DOLLAR
DATANASCITA  db "23/10/1920",DOLLAR
SESSO        db 'M',DOLLAR
CODEFISC     db 11 dup ('-'),DOLLAR ; output previsto: "BOXRIO20R23"
CRLF         db CR,LF,DOLLAR
codici_mesi  db "ABCDEHLMPRST"
vocali       db "AEIOU"
consbuf      db max_len dup ('X')   ; buffer consonanti nome/cognome
vocbuf       db max_len dup ('X')   ; buffer vocali nome/cognome
                                    ; ('X'=riempitivo nomi corti)
n_consonanti dw 0                   ; numero di consonanti rilevate nel nome/cognome
n_vocali     dw 0                   ; numero di vocali rilevate nel nome/cognome

DATI ENDS

;=================================================================

CSEG SEGMENT PUBLIC 'CODE'

MAIN proc far
        ASSUME CS:CSEG,DS:DATI,SS:PILA,ES:NOTHING;

        MOV AX,SEG DATI
        MOV DS,AX

elabora_cognome:
        separa_consonanti_vocali COGNOME,COGNOME_len
_riempi_campo_cognome:
        cmp n_consonanti,3
        jge __riempi_solo_consonanti_cognome
        riempi_consonanti_vocali codefisc
        jmp elabora_nome
__riempi_solo_consonanti_cognome:
        mov al,consbuf[0]
        mov codefisc[0],al
```

```
        mov al,consbuf[1]
        mov codefisc[1],al
        mov al,consbuf[2]
        mov codefisc[2],al

elabora_nome:
        separa_consonanti_vocali NOME,NOME_len
_riempi_campo_nome:
        cmp n_consonanti,3
        jg __riempi_solo_consonanti_nome
        riempi_consonanti_vocali codefisc+3
        jmp riempi_datasex
__riempi_solo_consonanti_nome:
        mov al,consbuf[0]
        mov codefisc[3],al
        mov al,consbuf[2]
        mov codefisc[4],al
        mov al,consbuf[3]
        mov codefisc[5],al

riempi_datasex:
_year:
        mov al,datanascita[8]
        mov codefisc[6],al
        mov al,datanascita[9]
        mov codefisc[7],al
_month:
        xor bx,bx
        cmp datanascita[3],'1' ; in datanascita[3] una cifra ('0' o '1')
        jne ___savemonth
        add bl,10              ; "10" <-> ottobre, etc.
___savemonth:
        mov ah,datanascita[4]  ; in datanascita[4] una cifra ('0','1', o '2')
        sub ah,'0'             ; in ah un numero (0, 1 o 2)
        add bl,ah              ; in bx un indice (1...12)
        dec bl                 ; in bx un indice (0...11)
        mov al,codici_mesi[bx]
        mov codefisc[8],al
_day:
        mov al,datanascita[0]  ; in al una cifra ('0','1','2' o '3')
        cmp sesso,'F'
        jne ___saveday
___add40:  add al,4            ; in al una cifra ('4','5','6' o '7')
___saveday:
        mov codefisc[9],al
        mov al,datanascita[1]
        mov codefisc[10],al

stampaavideo:
        display crlf
        display NOME
        display crlf
        display COGNOME
        display crlf
        display DATANASCITA
        display crlf
        display SESSO
        display crlf
        display crlf
        display codefisc
        display crlf

exit:
        MOV AH,4CH              ; ritorno al DOS
        INT 21H

main endp

cseg ends

END MAIN                    ; il programma comincia all'indirizzo di MAIN
```

```
TITLE CodiceControllo: per esame 1/7/2009

comment *
Costruzione del codice di controllo (1 lettera)
associato ai primi 15 caratteri del codice fiscale.
*

;-----------------------------------------------------------------
; Definizione costanti
CR EQU 13                       ; carriage return
LF EQU 10                       ; line feed
DOLLAR EQU '$'
num_caratteri equ 16            ; numero caratteri codice fiscale

;-----------------------------------------------------------------
;    M  A  C  R  O
;-----------------------------------------------------------------

display macro xxxx                  ; N.B. ogni stringa deve terminare con '$'
        push dx
        push ax
        mov dx,offset xxxx
        mov ah,9
        int 21h
        pop ax
        pop dx
endm

;-----------------------------------------------------------------
;
PILA SEGMENT STACK 'STACK'      ; definizione del segmento di stack
      DB      64 DUP('STACK')   ; lo stack e' riempito con la stringa 'stack'
                                ; per identificarlo meglio in fase di debug
PILA ENDS


;-----------------------------------------------------------------
;
DATI SEGMENT PUBLIC 'DATA'  ; definizione del segmento dati

                    ; N.B. per il codice fiscale usare solo lettere maiuscole.
                    ; Codice controllo previsto: 'K'. Questo carattere sostituira'
                    ; in memoria il carattere '$' della stringa codefisc
CODEFISC            db "CLMCRL66C03A662",'$',DOLLAR

CRLF                db CR,LF,DOLLAR
alfabeto            db "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
num_caralfabeto     equ $-alfabeto
valori_cardispari   db 1,0,5,7,9,13,15,17,19,21
                    db 1,0,5,7,9,13,15,17,19,21
                    db 2,4,18,20,11,3,6,8,12
                    db 14,16,10,22,25,24,23
value_even          dw 0
value_odd           dw 0

DATI ENDS


;=================================================================
;
CSEG SEGMENT PUBLIC 'CODE'

MAIN proc far
        ASSUME CS:CSEG,DS:DATI,SS:PILA,ES:NOTHING;

        MOV AX,SEG DATI
        MOV DS,AX

; stampa a video dell'input

display crlf
display codefisc
```

```
; accumulazione dei valori corrispondenti ai caratteri dispari

xor bx,bx
xor dh,dh          ; azzera la parte piu' significativa di dx
mov value_odd,0    ; superfluo (variabile gia' inizializzata in fase di allocazione)
accumula_dispari:
        mov al,codefisc[bx]
        mov cx, num_caralfabeto
        xor si,si
_cerca_carattere:
        cmp al,alfabeto[si]
        je _accumula_valore_dispari
        inc si
        loop _cerca_carattere
_accumula_valore_dispari:
        mov dl,valori_cardispari[si]
        add value_odd,dx
        add bx,2
        cmp bx, num_caratteri-1
        jl accumula_dispari

; accumulazione dei valori corrispondenti ai caratteri pari

xor ah,ah          ; azzera la parte piu' significativa di ax
mov bx,1
mov value_even,0   ; superfluo (variabile gia' inizializzata in fase di allocazione)
        accumula_pari:
        mov al,codefisc[bx]
_check_if_letter:
        cmp al,'A'
        jge __lettera
__cifra:
        sub al,'0'
        jmp _accumula_valore_pari
__lettera:
        sub al,'A'
_accumula_valore_pari:
        add value_even,ax
        add bx,2
        cmp bx, num_caratteri-1
        jl accumula_pari

; somma dei valori e calcolo codice di controllo

mov ax, value_odd
add ax, value_even

mov bl,26
div bl                          ; in ah il resto di AX/BL (in al il quoziente)

add ah,'A'                      ; conversione valore->lettera ascii

mov codefisc[num_caratteri-1],ah

; stampa a video dell'output

display crlf
display codefisc
display crlf


exit:
        MOV AH,4CH              ; ritorno al DOS
        INT 21H

main endp

cseg ends

END MAIN                        ; il programma comincia all'indirizzo di MAIN
```