1.1.1 Esercizio: conversione in caratteri maiuscoli

Scrivere un programma che accetta in ingresso una stringa di massimo 256 caratteri *esclusivamente* minuscoli terminata da ritorno carrello, stampa i singoli caratteri mentre vengono digitati, poi va a capo e stampa l'intera stringa a video in maiuscolo.

```
.DATA
messaggio:   .FILL 256,1,0

.TEXT
_main:          NOP
                MOV $254, %CX
                LEA messaggio, %EBX
ciclo:          CALL inchar
                CMP $0x0D, %AL
                JE dopo
                CMP $'a', %AL
                JB ciclo
                CMP $'z', %AL
                JA ciclo
                CALL outchar
                AND $0x5F, %AL            #converte in maiuscolo
                MOV %AL, (%EBX)
                INC %EBX
                DEC %CX
                JNZ ciclo
dopo:           MOVB $0x0A, (%EBX)
                MOVB $0x0D, 1(%EBX)
                CALL newline

                LEA messaggio, %EBX
                CALL outline
                CALL pause      # pausa per vedere cosa c'è sul video
                XOR %EAX, %EAX
                RET

.INCLUDE "C:/amb_GAS/utility"
```

1.1.2 Esercizio – algoritmo di Euclide per il MCD

Scrivere un programma Assembler che si comporta come segue:
1. legge da tastiera due numeri naturali A e B in base 10, sotto l'ipotesi che siano rappresentabili su 16 bit.
2. Se almeno uno dei due e' nullo, termina. Altrimenti,
3. Esegue l'algoritmo di Euclide per il calcolo del loro MCD, (riassunto di seguito), *stampando tutti i risultati intermedi*.
4. ritorna al punto 1.

L'algoritmo di Euclide per il calcolo dell'MCD tra due numeri A e B e':

```
 passo 0: i=0; X(0)=A;  Y(0)=B;
 passo i: stampa i, X(i), Y(i).
          se X(i)=0, allora Y(i)=MCD e l'algoritmo e' terminato.
          altrimenti:
            X(i+1)=max( X(i), Y(i) ) mod  min( X(i), Y(i) )
            Y(i+1)=min ( X(i), Y(i) )
            i=i+1
            ripeti
```

```
Esempio:
15                                      15120
10                                      4389
0) 15 10                                0) 15120 4389
1) 5 10                                 1) 1953 4389
2) 0 5                                  2) 483 1953
                                        3) 21 483
                                        4) 0 21
```

```
.DATA
X:              .WORD 0x0000
Y:              .WORD 0x0000

.TEXT
_main:       NOP

#1. legge da tastiera un numero naturale in base 10.

punto1:         MOV $'X', %AL
                CALL outchar
                MOV $':', %AL
                CALL outchar
                CALL indecimal_short
                MOV %AX, X
                CALL newline
                MOV $'Y', %AL
                CALL outchar
                MOV $':', %AL
                CALL outchar
                CALL indecimal_short
                MOV %AX, Y

#2. Se A=0, termina.
punto2:         CMPW $0, X
                JE fine_prog
                CMPW $0, Y
                JE fine_prog
```

```
#3.  Esegue l'algoritmo di Euclide per il calcolo del loro MCD,
#    in SI c'e' la variabile "i"
punto3:         MOV $0, %SI
ciclo:          MOV %SI, %AX
                CALL outdecimal_short
                MOV $')', %AL
                CALL outchar
                MOV $' ', %AL
                CALL outchar
                MOV X, %AX
                CALL outdecimal_short
                MOV $' ', %AL
                CALL outchar
                MOV Y, %AX
                CALL outdecimal_short
                CALL newline

                CMPW $0, X
punto4:         JE punto1

                MOV X, %AX
                MOV Y, %CX
                CMP %AX, %CX    #scambio AX e CX in modo che AX=max
                JBE dopo
                XCHG %AX, %CX

dopo:           MOV $0, %DX
                DIV %CX
                MOV %DX, X
                MOV %CX, Y

                INC %SI
                JMP ciclo

fine_prog:      XOR %EAX, %EAX
                RET

.INCLUDE "C:/amb_GAS/UTILITY"
```

1.1.3 Esercizio – calcoli con numeri naturali

Si implementi un programma Assembler che si comporta come segue:
1) legge con eco da tastiera due numeri nat. A e B in b10 (assumendo che siano rappresentabili su 16 bit) e un numero naturale N in base dieci (assumendo che sia rappresentabile su 8 bit)
2) se A>=B (maggiore o uguale), ovvero N=0 termina, altrimenti:
3) stampa a video, su una nuova riga la sequenza di N numeri:
B + (B-A), B + 2*(B-A), ... , B + N*(B-A)
eventualmente terminando la sequenza in anticipo qualora il successivo numero da stampare non appartenga all'intervallo di rappresentabilità per numeri naturali su 16 bit
4) ritorna al punto 1).

```
Esempio:
A:0013                      A:0000
B:0025                      B:9000
N:05                        N:12
37 49 61 73 85              18000 27000 36000 45000 54000 63000


.GLOBAL _main

.DATA
A:          .WORD 0x0000
B:          .WORD 0x0000

.TEXT
_main:       NOP

inizio:      CALL newline
             CALL newline
#punto 1
             MOV  $'A', %AL   #Legge A
             CALL outchar
             MOV  $':', %AL
             CALL outchar
             CALL indecimal_short
             MOV  %AX, A
             CALL newline
             MOV  $'B', %AL   #Legge B
             CALL  outchar
             MOV  $':', %AL
             CALL outchar
             CALL indecimal_short
             CALL newline
             MOV  %AX, B
             MOV  $'N', %AL   #Legge N
             CALL outchar
             MOV  $':', %AL
             CALL outchar
             CALL indecimal_tiny
             CALL newline
             MOV  %AL, %CL       # CL contiene il numero di iterazioni

# punto 2
             CMP  $0,%CL
             JE   fine
             MOV  A,%AX
```

```
            MOV  B,%DX
            CMP  %DX,%AX
            JAE  fine

# punto 3
            SUB  %AX,%DX        # DX contiene B-A
            MOV  B, %AX
ciclo:      ADD  %DX, %AX
            JC   inizio

stampa:     CALL outdecimal_short
            PUSH %AX
            MOV  $' ',%AL
            CALL outchar
            POP  %AX
            DEC  %CL
            JNZ  ciclo
# punto 4
            JMP  inizio

fine:       XOR %EAX, %EAX
            RET

.INCLUDE "C:/amb_GAS/utility"
```