Traccia di soluzione

2.

```
                .data
   ChiediEsp:   .asciiz "Inserire il valore dell'esponente "
   NonValido:   .asciiz "Esponente non valido!\n"
   Risp1:       .asciiz "Due elevato a "
   Risp2:       .asciiz "risulta: "

                .text
                .globl main

   main:        li $v0, 4                       # Stampa richiesta
                la $a0, ChiediEsp
                syscall
                li $v0, 5                       # Input n da tastiera
                syscall
                add $s0, $v0, $zero             # salva n nel registro $s0

                slti $t0, $s0, 32
                beq  $t0, $zero, Error          # se NOT(n<32), errore
                slti $t0, $s0, 0
                bne  $t0, $zero, Error          # se n<0, errore

                add $a0, $s0, $zero
                jal PowerOfTwo                  # calcola 2^n
                add $s1, $v0, $zero             # salva 2^n in $s1

                li $v0, 4                       # Stampa finale:
                la $a0, Risp1
                syscall

                li $v0, 1                       # stampa n (in $s0)
                li $a0, $s0
                syscall

                li $v0, 4
                la $a0, Risp2
                syscall

                li $v0, 1                       # stampa 2^n (in $s1)
                li $a0, $s1
                syscall

                li $v0, 10                      # termina esecuzione
                syscall

Error:          li $v0, 4
                la $a0, NonValido
                syscall
                j main
```

1.

```
PowerOfTwo:   addi $sp, $sp, -8          # prologo:
              sw $ra, 0($sp)             # salvo return address
              sw $t0, 4($sp)             # e $t0

              slti $t0, $a0, 1           # if $a0<1, set $t0
              beq $t0, $zero, else
              addi $v0, $zero, 1
              j epilogo

      else:   subi $a0, $a0, 1           # n → n-1
              jal PowerOfTwo
              addi $t0, $zero, 2         # $t0 ← 2
              mult $v0, $t0              # calcola: 2*PowerOfTwo(n-1)
              mflo $v0

   epilogo:   lw $t0, 4($sp)
              lw $ra, 0($sp)
              addi $sp, $sp, +8          # ripristino lo stack
              jr $ra                     # ritorno al chiamante
```

3. a) in Assembly MIPS nativo:

```
lui $a0, 256
addi $at, $zero, 12
div $t1, $at
mflo $s1
slti $at, $s1, -6
beq $at, $zero, -24
```

4.

Indirizzo:	Contenuto byte
0x300:	0x**00**
0x301:	0x**00**
2:	0x**04**
3:	0x**00**
4:	0x**FF**
5:	0x**FF**
6:	0x**FF**
7:	0x**F8**
8:	0x**18**
9:	0x**E8**
A:	0x**00**
0x30B:	0x**10**