Traccia di soluzione

2.

```
        .data
Chiedi: .asciiz "Inserire il valore dell'argomento "
Fine:   .asciiz "Programma terminato."
Risp1:  .asciiz "MioFibo( "
Risp2:  .asciiz " )= "

        .text
        .globl main

main:   li $v0, 4                   # Stampa richiesta
        la $a0, Chiedi
        syscall

        li $v0, 5                   # Input n da tastiera
        syscall
        add $s0, $v0, $zero         # salva n nel registro $s0

        bne $t0, $zero, END         # se n=0, salta alla fine

        add $a0, $s0, $zero
        jal MioFibo                 # calcola MioFibo(n)
        add $s1, $v0, $zero         # salva MioFibo(n) in $s1

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

        li $v0, 1                   # stampa argomento
        li $a0, $s0
        syscall

        li $v0, 4
        la $a0, Risp2
        syscall

        li $v0, 1                   # stampa risultato
        li $a0, $s1
        syscall

        j main                      # riesegui il calcolo

END:    li $v0, 4                   # Stampa messaggio Fine
        la $a0, Fine
        syscall

        li $v0, 10                  # termina esecuzione
        syscall
```


1.

```
MioFibo:    addi $sp, $sp, -12          # prologo:
            sw $ra, 0($sp)              # salvo return address,
            sw $a0, 4($sp)              # $a0 e
            sw $s0, 8($sp)              # $s0

            slti $t0, $a0, 2            # if $a0<2, set $t0
            beq $t0, $zero, else
            addi $v0, $zero, 1          # return( 1 )
            j epilogo

      else: subi $a0, $a0, 1            # n → n-1
            jal MioFibo
            addi $t0, $zero, 3          # $t0 ← 3
            mult $v0, $t0               # calcola: 3*MioFibo(n-1)
            mflo $s0                    # e salvalo in $s0
            subi $a0, $a0, 1
            jal MioFibo
            sub $v0, $s0, $v0           # risultato in $v0

  epilogo:  lw $s0, 8($sp)              # ripristino i registri
            lw $a0, 4($sp)
            lw $ra, 0($sp)
            addi $sp, $sp, +12          # ripristino lo stack
            jr $ra                      # ritorno al chiamante
```

3. a) in Assembly MIPS nativo: b) linguaggio macchina (in decimale)

```
0x500:  (inizio:)  add $s0, $a0, $s0    | 0  | 4  | 16 | 16 | 0 | 32 |
                   lw $t0, 0($s0)       | 36 | 16 |  8 |    0       |

        (ciclo:)   lui $s5, 64          | 15 |  0 | 21 |      64    |

                   slti $at, $t0, 101   | 10 |  1 |  8 |     101    |
                   beq $at, $zero, -12  |  4 |  1 |  0 |      -3    |

                   j 0x500              |  2 |     320 (= 0x140)     |
```

4.

Indirizzo: (decimale)	Contenuto byte
1600:	0xFF
01:	0xFF
02:	0xFE
03:	0x0C
04:	0x00
05:	0x00
06:	0x00
07:	0x00
08:	0x00
09:	0x00
10:	0x03
11:	0x00
12:	0xFF
1613:	0xFA

La syscall **sbrk** seguente restituirà l'indirizzo del primo byte successivo a questa area di memoria, e cioè: **1614 (= 0x064E)**