Nome: _____ Cognome: _____

Matricola: _____ Anno di immatricolazione _____

Prova di laboratorio di Architettura degli elaboratori

Istruzioni: *Ogni esercizio é descritto tramite un programma in linguaggio C. Se la realizzazione di un I/O non é richiesta, le variabili possono essere inizializzate istruzioni assembler o con direttive (nel caso degli array). Si raccomanda di seguire le convenzioni del linguaggio assembler specie per ció che riguarda le funzioni. Si raccomanda di utilizzare i commenti per indicare la corrispondenza fra variabili del C e registri. I commenti sono anche utili per descrivere cosa volete fare nel codice.*

1. [1.0] Programma che calcola il massimo fra 3 interi. L'esercizio deve essere risolto senza utilizzare macro (`bge ....`).

```
main()
{
  int a, b, c;
  int x; /* massimo */

  a=4; b=10; c=8;
  /* acquisisce a,b,c (non importa farlo) */
  x=c;
  if ((a>b) && (a>c))
       x=a;
  else
       if (b>c)
          x=b;
  /* stampa x (non importa farlo) */
}
```

Soluzione

```
# a, b, c, x mappd to $s0, $s1, $s2, $s3

.text
addi $s0, $zero, 4
addi $s1, $zero, 10
addi $s2, $zero, 8

addi $s3, $s2, 0

slt $t0, $s1, $s0 # b<a
slt $t1, $s2, $s0 # c<a

and $t2, $t0, $t1 # &&

bne $t2, $zero, label0
```

Nome: _____ Cognome: _____

```
slt $t3, $s2, $s1 # c<b
beq $t3, $zero, end
addi $s3, $s1, 0
j end
label0: # (a>b) && (a>c)
addi $s3, $s0, 0
end:

# not requested
addi $v0, $zero, 1
addi $a0, $s3, 0
syscall
```

2. [2.0] Calcolo del massimo di un vettore. Anche in questo caso non si devono utilizzare macro.

```
main()
{
  int array[8]={0,1,4,2,7,8,4,6};
  int i, x; /* x = massimo del vettore */
  /* acquisisce array (da non fare) */
  i=1;
  x=array[0];
  while (i<8)
    {
       if (array[i]>x)
           x=array[i];
       i=i+1;
    }
  /* stampa x (da non fare) */
}
```

Soluzione

```
.data
array0: .word 0,1,4,2,7,8,4,6
# the two arrays have the same size

.text

# i in $s0, x in $ $s1

# index i
addi $s0, $zero, 4
# init x
lw $s1, array0($zero)

# array size in bytes
```

```
addi $t0, $zero, 32

loop: beq $s0, $t0, exit
# load array[i] in $t2
lw $t2, array0($s0)
slt $t1, $s1, $t2
beq $t1, $zero, label # x>=array[i]
addi $s1, $t2, 0
label: addi $s0, $s0, 4
j loop
exit:
```

3. [2.0] Funzione che calcola la distanza fra due interi.

```
int main()
{
  int x,y;
  int v;

  x=7; y=4;
  /* acquisice x e y (da non fare) */

  v=dist(x,y);
}

int dist(int x, int y)
{
  int result;

  if (a>b)
      result=a-b;
  else
      result=b-a;
  return result;
}
```

Soluzione

```
# $s0 = y

main:
  addi $s0, $zero, 7 # x
  addi $s1, $zero, 4 # y

# operations on $s0 and $s1

  addi $a0, $s0, 0    # argument 1
```

Nome: _____ Cognome: _____

```
  addi $a1, $s1, 0    # argument 2
  jal  dist    # call Function
  addi  $s2, $v0, 0   # returned value

# $s0 = result
dist:
  addi $sp, $sp, -8  # make space on stack to
                     # store two registers
  sw  $t0, 0($sp)    # save $t0 on stack
  sw  $t1, 4($sp)    # save $t1 on stack
                     # not mandatory for t registers
  slt $t0, $a1, $a0  # x>y
  beq $t0, $zero, label
  sub $t1, $a0, $a1  # x-y
  j join
  label: sub $t1, $a1, $a0
  join:
  addi $v0, $t1, 0
  lw $t0, 0($sp)     # restore $t0 from stack
  lw $t1, 4($sp)     # restore $t0 from stack
  addi $sp, $sp, 8   # deallocate stack space
  jr  $ra            # return to caller
```

4. [1.5] Programma che calcola il numero di uni presenti in un intero a 32 bit.

```
int main()
{
  int i, n, x, y;

  n=i=0;
  x=18; /* intero di cui si calcola in no. di uni */

  while (i<32)
    {
       y=x & 1;
       n=n+y;
       x=x>>1;
       i=i+1;
    }
}
```

Soluzione

```
addi $s0, $zero, -68 # x
addi $s1, $zero, 0 # y
addi $s2, $zero, 0 # n
addi $s3, $zero, 0 # i
```

Nome: _____ Cognome: _____

```
addi $t0, $zero, 32

loop: beq $s3, $t0, out_of_loop
andi $t1, $s0, 1 # y = x & 1
add $s2, $s2, $t1 # n = n + y
srl $s0, $s0, 1 # x=x>>1
addi $s3, $s3, 1
j loop
out_of_loop:
```