

Nº Mec.: _____ Nome: _____

NOTE BEM: Leia atentamente todas as questões, comente o código usando a linguagem C e respeite a convenção de passagem de parâmetros e salvaguarda de registos que estudou. Na tradução para o *Assembly* do MIPS respeite rigorosamente os aspetos estruturais e a sequência de instruções indicadas no código original fornecido.

O código em C apresentado pode não estar funcionalmente correcto, pelo que **não deve ser interpretado**.

Este teste é constituído por 4 folhas.

1) Analise o programa *Assembly* seguinte e responda às questões que se seguem:

```

      .data                      # 0x10010000
X1:   .ascii "TEST1"            #
      .align 2                  #
X2:   .space 20                  #
X3:
      .text                      # 0x00400000
      .globl main
main:  la      $t4,X2            #
      ori     $t5,$0,4          #
      xor     $t0,$t0,$t0       #
      xor     $t1,$t1,$t1       #
L1:   beq     $t0,$t5,L2        #
      add     $t2,$t0,$t0       #
      add     $t3,$t2,$t2       #
      addu    $t3,$t3,$t4       #
      sw      $t2,0($t3)        #
      add     $t1,$t1,$t2       #
      addi    $t0,$t0,1         #
      j       L1               #
L2:   sw      $t1,4($t3)        #
      jr      $ra              #
```

- a) Qual o espaço total de memória ocupado pela *string* "X1"? _____
- b) Qual o endereço de memória a que corresponde o *label* "X2"? _____
- c) Se "X2" for o endereço inicial de um *array* de inteiros, qual a dimensão máxima desse *array*? _____
- d) Se "X2" for o endereço inicial de um *array* de inteiros, qual o endereço de memória da posição X2 [3] desse *array*? _____
- e) Qual o número total de bytes de memória usado pelo segmento de dados (X3-X1)? _____
- f) Considerando que a primeira instrução do trecho de código fornecido está armazenada a partir do endereço 0x00400000, quais os endereços a que correspondem os *labels* "L1" e "L2"? (note que a instrução "la" é decomposta em duas instruções nativas).
L1: _____ L2: _____
- g) Quantas vezes é realizado o ciclo de programa? _____
- h) Qual o valor da *word* de 32 bits armazenada pelo programa na posição X2 [3] do *array*? _____
- i) Qual o valor do registo \$t1 no fim do programa?
\$t1: _____
- j) Qual o endereço de memória acedido pela instrução "sw \$t1, 4(\$t3)"? _____

```
int split_odd(int *a, int N, int *p_odd )
{
    int n_even = 0;
    int *p;

    for( p = a; p < (a + N); p++ )
    {
        if( (*p % 2) != 0 )
        {
            *p_odd = *p;
            p_odd++;
        }
        else
            n_even++;
    }
    return (N - n_even);
}
```

Variável	Registro(s)
a	
N	
p_odd	
n_even	
p	
*p	

[illegible][illegible]

```
#define SIZE 7
int splito(int *, int, int *);

int main(void)
{
    static int val[SIZE] = {8, 4, 15, -1987, 9, 27, 16};
    static int odd[SIZE];
    int nodd, i;

    nodd = splito( val, SIZE, odd );
    print_string("Result is:");
    for( i=0; i < nodd; i++ ) {
        print_int10( odd[i] );
    }
    return 1;
}
```

Variável	Registo(s)
nodd	
i	

[illegible][illegible]

```
int isn( char, char );

int count(char *p, char c)
{
    int n=0;
    while ( *p != '\0' )
    {
        n = n + isn(*p, c);
        p++;
    }
    return n;
}
```

Variável	Registro(s)
p	
c	
n	
*p	

[illegible][illegible]