

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"?  
**6 bytes, 5 + terminador (de 0x10010000 a 0x10010005)**
- b) Qual o endereço de memória a que corresponde o *label* "X2"?  
**0x10010008 (1º endereço múltiplo de 4 a seguir a 0x10010005)**
- c) Se "X2" for o endereço inicial de um *array* de inteiros, qual a dimensão máxima desse *array*?  
**20 / 4 = 5 inteiros**
- d) Se "X2" for o endereço inicial de um *array* de inteiros, qual o endereço de memória da posição X2[3] desse *array*?  
**&X2[3] = 0x10010008 + 4 \* 3 = 0x10010014**
- e) Qual o número total de bytes de memória usado pelo segmento de dados (X3-X1)?  
**28 bytes**
- 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: **0x00400014**      L2: **0x00400034**
- g) Quantas vezes é realizado o ciclo de programa?  
**4 vezes (\$t0=0, \$t0=1, \$t0=2, \$t0=3)**
- h) Qual o valor da *word* de 32 bits armazenada pelo programa na posição X2[3] do *array*?  
**X2[3] = 2\*3 = 6**
- i) Qual o valor do registo \$t1 no fim do programa?  
**\$t1: 2\*0 + 2\*1 + 2\*2 + 2\*3 = 12 = 0x0000000C**
- j) Qual o endereço de memória acedido pela instrução "sw \$t1,4(\$t3)"?  
**&X2[3+1]  
= 0x10010008 + 4 \* 4  
= 0x10010018**

```
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	Registo(s)
a	\$a0
N	\$a1
p_odd	\$a2
n_even	\$t0
p	\$t1
*p	\$t2

## Função terminal

[illegible]

N.º Mec.: \_\_\_\_\_ Nome: \_\_\_\_\_

3) Codifique em *Assembly* do MIPS a seguinte função `main()` :

```
#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	\$t0
i	\$t1

Função intermédia

Label	Instrução em <i>assembly</i>	Comentário em C	Label	Instrução em <i>assembly</i>	Comentário em C
	.data				
	.eqv SIZE,7				
	.eqv PRINT_STR,4				
	.eqv PRINT_INT10,1				
str:	.asciiz "Result is:"				
val:	.word 8,4,15,-1987,9,27,16				
odd:	.space 28	# SIZE * 4 = 28			
	.text				
	.globl main				
main:	subu \$sp,\$sp,4	#			
	sw \$ra,0(\$sp)	#			
	la \$a0,val	#			
	li \$a1,SIZE	#			
	la \$a2,odd	#			
	jal splito	#			
	move \$t0,\$v0	# nodd = splito(val,SIZE,odd)			
	li \$v0,PRINT_STR	#			
	la \$a0,str	#			
	syscall	# print_str("Result is:")			
	li \$t1,0	# i=0			
for:	bge \$t1,\$t0,endif	# while(i < nodd) {			
	sll \$t2,\$t1,2	#			
	la \$t3,odd	#			
	addu \$t3,\$t3,\$t2	# \$t3 = &odd[i]			
	lw \$a0,0(\$t3)	# \$a0 = odd[i]			
	li \$v0,PRINT_INT10	#			
	syscall	# print_int10(odd[i])			
	addi \$t1,\$t1,1	# i++			
	j for	# }			
endif:	li \$v0,1	# return 1			
	lw \$ra,0(\$sp)	#			
	addu \$sp,\$sp,4	#			
	jr \$ra	#			

```
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	\$a0 -> \$s0
c	\$a1 -> \$s1
n	\$s2
*p	\$a0

## Função intermédia

[illegible]