Pergunta **1**

Por responder

Nota: 4,00

Marcar pergunta

```
#define SIZE 8

int main(void)
{
    static int list[SIZE];
    int rv, n;
    int sum = 0, nit = 0;
    int *pt;

    for (pt = list; pt < (list + SIZE); pt++) {
        *pt = read_int();
    }
    for (n = 0; n < SIZE; n++) {
        if (list[n] >= 0) {
            sum += list[n];
            nit++;
        }
    }
    if (nit > 0) {
        print_int10(sum / nit);
        rv = 0;
    } else {
        print_string("Media invalida!\n");
        rv = -1;
    }
    return rv;
}
```

```
# Copie para o topo da área de resposta
# e preencha com os registos usados
# Mapa de registos
# rv :
# n  :
# sum:
# nit:
# pt :
```

Copie o mapa de registos para o topo da área de resposta e preencha-o com os registos usados.
Codifique em *Assembly* do MIPS a função `main()`.

Pergunta **2**

Por responder

Nota: 4,00

Marcar pergunta

```
#define SIZE 15

int toi( char * );
int avz( int *, int );

int func2(int *fl, int k, char *av[])
{
   int i;
   int res = -1;

   if ((k >= 2) && (k <= SIZE)) {
      i = 2;
      do {
         fl[i] = toi(av[i]);
         i++;
      } while (i < k);
      res = avz(fl, k);
      print_int10(res);
   } else
      print_string("Invalid argc");
   return res;
}
```

```
# Copie para o topo da área de resposta
# e preencha com os registos usados
# Mapa de registos
# fl:
# k:
# av:
# i:
# res:
```

Copie o mapa de registos para o topo da área de resposta e preencha-o com os registos usados.
Codifique em *Assembly* do MIPS a função **func2()**.

Pergunta **3**

Por responder

Nota: 4,00

Marcar pergunta

```
float func3(float *a, float t, int n)
{
   float oldg = -1.0;
   float g = 1.0;
   float s = 0.0;
   int k;

   for (k = 0; k < n; k++) {
      while ((g - oldg) > t) {
         oldg = g;
         g = (g + a[k] / t);
      }
      s = s + g;
      a[k] = g;
   }
   return s / (float) n;
}
```

```
# Copie para o topo da área de resposta
# e preencha com os registos usados
# Mapa de registos
# a:
# t:
# n:
# oldg:
# g:
# s:
# k:
```

Preencha o mapa de registos e codifique em *Assembly* do MIPS a função **func3()**.

Pergunta **4**

Por responder

Nota: 4,00

Marcar pergunta

```
double func4(int nv, t_kvd *pt)
{
   int i, j;
   double sum = 0.0;

   for (i = 0; i < nv; i++, pt++) {
      j = 0;
      do {
         sum += (double) pt->quest[j];
         j++;
      } while (j < pt->nm);

      pt->acc = (int) (sum / pt->grade);
   }
   return (pt->grade * (double) pt->cq);
}
```

```
# Copie para o topo da área de resposta e
# substitua xx pelo valor adequado
# typedef struct
# {                          Align  Size  Offset
#    int acc;                 xx     xx    xx
#    unsigned char nm;        xx     xx    xx
#    double grade;            xx     xx    xx
#    char quest[14];          xx     xx    xx
#    int cq;                  xx     xx    xx
# } t_kvd;                    xx     xx
# Copie para o topo da área de resposta e
# preencha com os registos usados
# Mapa de registos
# nv:
# pt:
# i:
# j:
# sum:
```

Preencha o mapa de registos e a tabela com os dados da estrutura. Codifique em *Assembly* do MIPS a função **`func4()`**.

Pergunta **5**

Por responder

Nota: 4,00

Marcar pergunta

```
        .data
AA:     .asciiz  "#F47D3FA2"
BB:     .word    5
CC:     .word    0x52, 0x126C, 0x3A, 0x139A8, 0xABA, 0x7C38
        .align   2
DD:     .space   4
        .text
        .globl  main
main:   la      $t0, CC
        la      $t1, BB
        lw      $t1, 0($t1)
        sll     $t1, $t1, 2
        addu    $t1, $t0, $t1
        xor     $t2, $t2, $t2
L1:     lw      $t3, 0($t0)
        lw      $t4, 0($t1)
        sw      $t3, 0($t1)
        sw      $t4, 0($t0)
        andi    $t4, $t4, 0x0F
        add     $t2, $t2, $t4
        addiu   $t0, $t0, 4
        addiu   $t1, $t1, -4
        blt     $t0, $t1, L1
L2:     la      $t5, DD
        sw      $t2, 0($t5)
        la      $t3, AA
        lw      $v0, 0($t3)
        jr      $ra
```

Carater	ASCII
'!'	0x21
'#'	0x23
'$'	0x24
'%'	0x25
'*'	0x2A
'+'	0x2B
'-'	0x2D
'0'	0x30
'@'	0x40
'A'	0x41
'Z'	0x5A
'a'	0x61
'z'	0x7A

Analise o código *assembly* e responda às questões seguintes. Considere que o segmento de dados do programa com

```
L2:     la      $t5, DD
        sw      $t2, 0($t5)
        la      $t3, AA
        lw      $v0, 0($t3)
        jr      $ra
```

Tempo restante

Analise o código *assembly* e responda às questões seguintes. Considere que o segmento de dados do programa começa no endereço `0x1001003C` e que a sua primeira instrução está armazenada no endereço `0x00400038`.

O número total de posições de memória ocupado pela *string* "**AA**"é: Escolha...

O número total de posições de memória ocupado pelo segmento de dados é: Escolha...

O endereço a que corresponde o *label* "**L2**" é (tenha em atenção as instruções virtuais do código): Escolha...

O valor do registo **$t0** após a execução da primeira instrução do trecho de código é Escolha...

Se "**CC**" referenciar um *array* de inteiros, o endereço de memória do elemento **CC[3]** é: Escolha...

O valor do registo **$t1** calculado na instrução "**addu $t1,$t0,$t1**" é: Escolha...

O valor do registo **$t1** no final da execução do trecho de código é: Escolha...

No final da execução do programa, o valor armazenado em **CC[2]** é: Escolha...

No final da execução do programa o valor armazenado na variável "**DD**" é: Escolha...

O valor de retorno da função *main()* é: Escolha...

Pode usar esta área como folha de rascunho