Arquitetura de Computadores
Aulas Práticas 2017/2018

5. Programação em Assembly MIPS R2000

1. O que calcula o seguinte programa?

```
        .data
str:    .asciiz  "The result is ..."

        .text
_main:
        addiu    $t0, $zero, 0
        addiu    $t2, $zero, 0
_loop:
        mul      $t1, $t0, $t0
        add      $t2, $t2, $t1
        addiu    $t0, $t0, 1
        ble      $t0, 100, _loop
        la       $a0, str
        addiu    $v0, $zero, 4
        syscall
        add      $a0, $t2, $zero
        addiu    $v0, $zero, 1
        syscall
        addiu    $v0, $zero, 10
        syscall
```

2. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (soma de uma sequência de inteiros positivos).

```
#include <stdio.h>

int main () {
   int i, sum, upTo;
```

```
   scanf("%d", &upTo);
   sum = 0;
   for (i = 0; i < upTo ; i++)
      sum += i * i;
   printf("sum = %d\n", sum);
   return 0;
}
```

3. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (cálculo do quadrado de inteiros positivos).

```
#include <stdio.h>
int squares[64];

int main () {
   int i, upTo;
   scanf("%d", &upTo);
   for (i = 0; i < upTo ; i++)
      squares[i] = i * i;
   return 0;
}
```

4. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (soma de uma sequência de quadrados de inteiros positivos).

```
#include <stdio.h>
int squares[64];

int main () {
   int i, sum, upTo;
   scanf("%d", &upTo);
   for (i = 0; i < upTo ; i++)
      squares[i] = i * i;
   sum = 0;
   for (i = 0; i < upTo ; i++)
      sum += squares[i];
   printf("sum = %d\n", sum);
   return 0;
}
```

5. Que procedimento realiza o seguinte programa?

```
        .data
msg:    .asciiz  "result is: "
str:    .asciiz  "abracadabra"

        .text
_main:
        la      $a0, str
        jal     _proc
        add     $s0, $v0, $zero
        la      $a0, msg
        addiu   $v0, $zero, 4
        syscall
        add     $a0, $s0, $zero
        addiu   $v0, $zero, 1
        syscall
        addiu   $v0, $zero, 10
        syscall
_proc:
        addiu   $v0, $zero, 0
_loop:
        lb      $t0, 0($a0)
        beqz    $t0, _exit
        addiu   $v0, $v0, 1
        addiu   $a0, $a0, 1
        j       _loop
_exit:
        jr      $ra
```

6. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (soma de uma sequência de quadrados de inteiros positivos utilizando funções auxiliares).

```
#include <stdio.h>
int squares[64];

void storeValues(int n) {
   int i;
   for (i = 0; i < n ; i++)
      squares[i] = i * i;
   return;
}
```

```
int computeSum(int n) {
   int i, sum;
   sum = 0;
   for (i = 0; i < n ; i++)
      sum += squares[i];
   return sum;
}

int main () {
   int upTo;
   scanf("%d", &upTo);
   storeValues(upTo);
   printf("sum = %d\n", computeSum(upTo));
   return 0;
}
```

7. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (cálculo da frequência de cada letra numa frase).

```
#include <stdio.h>
char text[] = "This is the string to be used to generate the histogram";
int histogram[26] = {0};

void computeHistogram() {
   int i = 0;
   while (text[i] !=  '\0') {
      if (text[i] >= 'A' && text[i] <= 'Z')
         histogram[text[i] - 65]++;
      if (text[i] >= 'a' && text[i] <= 'z')
         histogram[text[i] - 97]++;
      i++;
   }
   return;
}

void printHistogram() {
   int i;
   for (i = 0; i < 26 ; i++)
      printf("%c -> %d\n", i + 97, histogram[i]);
   return;
}
```

```
int main () {
   computeHistogram();
   printHistogram();
   return 0;
}
```

8. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (ordenação de um vector de inteiros pelo método da bolha).

```
#include <stdio.h>
int array[] = {32, 6, 51, 63, 22, 29, 91, 39, 66, 47};
int size = 10;

void bsort(int v[], int n) {
   int i, j;
   for (j = 0 ; j < n ; j++) {
      for (i = 0 ; i < n - 1 ; i++) {
         if (v[i+1] < v[i]) {
            int tmp;
            tmp = v[i];
            v[i] = v[i+1];
            v[i+1] = tmp;
         }
      }
   }
   return;
}

int main () {
   int i;
   bsort(array,size);
   for (i = 0 ; i < size ; i++)
      printf("%d ", array[i]);
   printf("\n");
   return 0;
}
```

9. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (versão recursiva ingénua dos números de Fibonnacci).

```
#include <stdio.h>

int fib (int n) {
   return (n < 2) ? n : fib(n - 1) + fib(n - 2);
}

int main () {
   printf("fib(45) = %d\n", fib(45));
   return 0;
}
```

10. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (versão recursiva ingénua dos números de Fibonnacci com memorização dos valores calculados).

```
#include <stdio.h>
int computedFibs[1024] = {0,1,1};

int fib (int n) {
   if (computedFibs[n] == 0)
       computedFibs[n] = fib(n - 1) + fib(n - 2);
   return computedFibs[n];
}

int main () {
   printf("fib(45) = %d\n", fib(45));
   return 0;
}
```

11. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (cálculo do máximo divisor comum de forma iterativa).

```
#include <stdio.h>

int gcd(int i, int j) {
   while (i != j) {
      if (i > j)
         i = i - j;
```

```
      else
         j = j - i;
   }
   return i;
}

int main () {
   int i, j, k;
   scanf("%d", &i);
   scanf("%d", &j);
   k = gcd(i, j);
   printf("mdc(%d,%d) = %d\n", i, j, k);
   return 0;
}
```

12. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (cálculo do máximo divisor comum de forma recursiva).

```
#include <stdio.h>

int gcd(int i, int j) {
   if (j == 0)
      return i;
   else
      return gcd(j, i % j);
}

int main () {
   int i, j, k;
   scanf("%d", &i);
   scanf("%d", &j);
   k = gcd(i, j);
   printf("mdc(%d,%d) = %d\n", i, j, k);
   return 0;
}
```

13. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (cálculo de números primos usando o crivo de Eratóstenes).

```
#include <stdio.h>
int primes[64] = {2};
```

```
int is_prime(int i, int k) {
   int j = 0;
   while (j <= k) {
      if (i % primes[j] == 0)
         break;
      j++;
   }
   return (j > k);
}

int main () {
   int i, j, k, n, prime;
   k = 0;
   scanf("%d",&n);
   for (i = 2; i < n; i++)
      if (is_prime(i,k))
         primes[++k] = i;
   for (i = 0; i <= k; i++)
      printf("%d\n", primes[i]);
   return 0;
}
```

14. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (procura num vetor utilizando pesquisa binária).

```
#include <stdio.h>
int v[] = {2, 9, 13, 15, 26, 31, 37, 49, 51, 53, 54, 62, 66, 73, 75, 84, 91};
int size = 17;
int value = 9;

int bsearch (int value, int low, int high) {
   int middle;
   if (low > high)
      return -1;
   middle = (low + high) / 2;
   if (value > v[middle])
      return bsearch(value, middle + 1, high);
   else if (value < v[middle])
      return bsearch(value, low, middle - 1);
   else
      return middle;
```

```
}

int main () {
   int index;
   index = bsearch(value, 0, size - 1);
   if (index == -1)
      printf("%d is not in vector", value);
   else
      printf("%d is in vector at index %d", value, index);
   return 0;
}
```

15. Escreva o código em assembly MIPS R2000 correspondente ao programa em C que se segue, preservando a sua estrutura funcional (rotação circular de uma sequência de texto para a direita).

```
#include <stdio.h>
char text[64] = "Please shift me to the right...";

int size() {
   int i;
   for (i=0; text[i] != '\0'; i++);
   return i;
}

void shiftOnce() {
   int i, j, last;
   j = size();
   last = text[j-1];
   for (i=j-2; i >= 0; i--)
      text[i+1] = text[i];
   text[0] = last;
   return;
}

void shiftMany(int number) {
   int i;
   for (i=0 ; i < number ; i++)
      shiftOnce();
   return;
}

int main () {
```

```
    shiftMany(5);
    printf("%s\n", text);
    return 0;
}
```