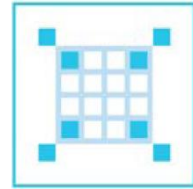


Arquitectura de Computadores

LIC. EM ENG^a INFORMÁTICA
FACULDADE DE CIÊNCIA E TECNOLOGIA
UNIVERSIDADE DE COIMBRA



Lab 9 – Funções no MIPS e Interligação com o C – Chamada de Outras Funções

1. Chamada a Funções Dentro de Funções

Inspirado no conhecido conjunto de Mandelbrot, considere o seguinte programa em C que imprime o valor de z ao fim de n iterações, sendo o valor inicial de z (z_0) e o valor de n indicados pelo utilizador.

$$z(0) = z_0$$
$$z(n+1) = z^2(n) + z_0$$

```
#include <stdio.h>

int Mandelbrot (int z0, n);

int main ()
{
    int z0, n, z;
    printf ("Introduza o valor inicial (z0): ");
    scanf ("%d", &z0);
    printf ("Indique o número de iterações (n): ");
    scanf ("%d", &n);
    z = Mandelbrot (z0, n);
    printf ("O resultado é: %d\n", z);
    return 0;
}
```

Usando a convenção para chamada de funções, escreva em linguagem *assembly* a função `Mandelbrot()` que irá calcular o resultado de z para a n -ésima iteração solicitada pelo utilizador. **O cálculo do quadrado de $z(n)$ deverá ser efetuado através de uma segunda função também implementada em *assembly* e que será invocada pela função `Mandelbrot()`.** Nota: poderá admitir que o resultado das funções não ultrapassa os 32 bits. Utilize para isso valores pequenos para n ($n \leq 5$, com $z_0=1$). Para efeitos de teste considere que deverá obter $Z_4=677$ e $Z_5=458.330$, com $Z_0=1$.

2. Manipulação de array de inteiros

Escreva duas funções em linguagem Assembly do MIPS (`manipula_tabela()` e `inverte_tabela()`) que recebem como parâmetros de entrada um ponteiro para uma

tabela de inteiros e o seu tamanho. A função `manipula_tabela()` será chamada na função `main()` escrita em linguagem C e cujo código é fornecido. A função `manipula_tabela()` irá manipular essa mesma tabela, multiplicando por 2 cada um dos seus elementos. De seguida, dentro da função `manipula_tabela()` deverá ser chamada a função `inverte_tabela()` que irá permitir inverter a ordem de todos os elementos da tabela (o 1º elemento troca de posição com o último, o 2º elemento com o penúltimo, etc...). Na função `main()`, escrita em C, é feita a impressão da tabela original e da tabela resultante.

```
main() -> manipula_tabela(int*,int) -> inverte_tabela(int*,int)
```

Nota: Pode encontrar em anexo o ficheiro `main.c`.

3. Implementação da binarização de uma imagem em *Assembly*

Recorda-se da função de binarização de uma imagem que foi implementada em C na Ficha 7. Pretende-se agora implementar esta função, mas desta vez em *assembly*. Para isso deverá completar o código da função **`bin_img()`** disponibilizado no ficheiro **`bin_img.s`** de forma a esta binarizar a imagem passada em memória. Note que o endereço de memória onde a imagem está guardada é indicado pelo ponteiro **`ptr`**, e a largura e altura da imagem são passadas diretamente em **`w`** e **`h`**. O limiar de binarização é passado no parâmetro **`limiar`**.

4. Ainda o binarizar de uma Imagem

Nas condições do exercício anterior, implemente uma nova variante da função `bin_img()` que não receba o parâmetro `limiar`. A própria função deverá chamar outra função, também implementada em *assembly*, chamada `calcula_limiar(unsigned char *ptr, int w, int h)` que devolva um limiar calculado como a média de todos os pixéis na imagem.