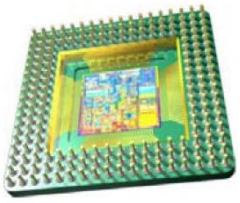

Arquitectura de Computadores

LIC. EM ENGª INFORMÁTICA
FACULDADE DE CIÊNCIA E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Lab 10 – 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");
     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<=5, com z_0=1).

2. Manipulação de array de inteiros

Escreva duas funções em linguagem Assembly do MIPS (`manipula_array()` e `inverte_array()`) que recebem como parâmetros de entrada um ponteiro para um array de inteiros e o tamanho do array. A função `manipula_array()` será chamada na

função `main()` escrita em linguagem C e cujo código é fornecido. A função `manipula_array()` irá manipular esse mesmo array, multiplicando por 2 cada um dos seus elementos. De seguida, dentro da função `manipula_array()` deverá ser chamada a função `inverte_array()` que irá permitir inverter a ordem de todos os elementos do array manipulado (o 1º elemento troca de posição com o último, o 2º elemento com o penúltimo, etc…). Na função `main()` escrita em C é feito o print do array original e do array manipulado e invertido.

`main() -> manipula_array(int*, int) -> inverte_array(int*, int)`

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

3. Binariza em Assembly

i) Lembra-se da função "binariza" do LAB 6? Na altura foi fornecido um ficheiro `main.c` que chamava uma função `bin_img()`. A vossa missão foi programar `bin_img()`, criar o respetivo código objeto, e ligá-lo com `main.o` para obter uma aplicação final. Esta função percorria a imagem e colocava os píxeis a preto e branco conforme o valor de um limiar. Desta vez pretende-se que programe a função `bin_img()` diretamente em *assembly*. Teste a sua solução.

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

ii) Ainda nas condições do exemplo 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)` e que devolva um limiar calculado como a média de todos os pixéis na imagem.