

## Trabalho Prático 7

### Controlo de Fluxo de Execução

#### Objetivos

- Instruções de salto relativo e salto absoluto.
- Programas *Assembly* com estruturas de controlo de fluxo.

#### Introdução

Em condições normais as instruções dum programa, armazenadas em memória, são executadas pelo processador (CPU) duma forma sequencial. Isto é, o processador lê uma instrução dum dado endereço, descodifica-a e executa-a. Em seguida, lê a instrução seguinte a partir do endereço igual ao endereço anterior incrementado de 4, e assim sucessivamente.

Contudo a maioria dos programas implementa algoritmos onde alguma forma de não-sequencialidade é necessária, requerendo do processador um tipo de instruções que a suporta.

O MIPS inclui instruções que permitem alterar o fluxo de execução do código: (i) as instruções de salto condicional **beq** (*branch if equal*) e **bne** (*branch if not equal*) e (ii) a instrução de salto incondicional **j** (*jump*). As estruturas usadas em programação de alto nível (e.g., em *Java*, *C*, etc) do tipo *if*, *if/else* e ainda os ciclos iterativos *while* e *for*, são implementadas em *Assembly* com base nestas instruções de salto.

#### Guião

##### 1. Estruturas de Controlo de fluxo

###### 1.1 Codifique em *Assembly* o seguinte programa:

```
void main(void)
{
    int a;
    char prompt1[] = "Introduza um numero\n";
    char strpar[] = "O numero é par\n";
    char strimp[] = "O numero é impar\n";

    print_str( prompt1 );
    a = read_int();

    if ((a & 1) == 0){
        print_str( strpar );
    }
    else {
        print_str( strimp );
    }
}
```

**1.2** Codifique em *Assembly* o seguinte programa:

```

void main(void)
{
    int a, i;
    char prompt1[] = "Introduza um numero\n";

    print_str( prompt1 );
    a = read_int();

    for(i = 0; i < a; i++)
    {
        print_char( '-' );
    }
}

```

**1.3** Codifique em *Assembly* o seguinte programa, e teste-o determinando até que número introduzido funciona corretamente.

```

void main(void)
{
    int i,n, f;
    char prompt1[] = "Introduza um numero\n";
    char result[] = "O fatorial do número inserido é: ";

    print_str( prompt1 );
    n = read_int();
    f=1;
    for ( i = n; i>0; i--)
    {
        f = f*i;
    }

    print_str( prompt );
    print_int( f );
}

```

## 2. Exercícios Adicionais

**2.1** Escreva numa linguagem de alto nível (e.g., C) um programa que leia cinco números inteiros (positivos ou negativos) e determine a soma dos números positivos.

**2.2** A abordagem seguida no programa 4 da aula passada (para imprimir os caracteres hexadecimais de um número presente num registo) resulta num programa ineficiente por ter uma grande repetição de instruções. Uma possível codificação alternativa usando um ciclo **for** é:

```
void main(void)
{
    char prompt1[] = "Introduza um numero: ";
    char result[] = "\n O número em hexadecimal e': ";
    int n, num;

    print_str( prompt1 );
    num = read_int();
    print_str( result );

    for(n=0; n<8;n++)
    {
        print_int16((num & 0xF0000000) >> 28);
        num = num << 4;
    }
}
```

**2.3** Altere o programa anterior de modo a imprimir o número em binário em vez de hexadecimal.