

# LAB #3: INTRODUÇÃO AO ASSEMBLY

## 1. MOTIVAÇÃO & OBJECTIVOS

Neste laboratório vamos introduzir a programação do processador didático P3 em linguagem Assembly (ASM), estabelecendo o paralelismo com a linguagem de Python (linguagem de alto nível que está a ser estudada em simultâneo em Fundamentos de Programação). A ideia é começar a fazer a ponte entre o software e o hardware, começando para tal por perceber como é que pequenos conjuntos de instruções são traduzidos de uma linguagem de alto nível para Assembly e observar sua a execução no P3.

Antes de prosseguir com este guia de laboratório deverá estudar e seguir o Tutorial do P3, bem como folhear brevemente o manual do P3 (ambos disponíveis na página do fénix da cadeira). Ambos contêm indicações valiosas para conseguir usar as ferramentas do P3.

O trabalho de casa para a semana 3 é a resolução de **todos** os exercícios em grupos de 2, devendo a solução ser entregue em papel no início da aula de laboratório de cada grupo.

## 2. AMBIENTES DE DESENVOLVIMENTO P3

Para poder executar programas em Assembly P3 é necessário ter acesso ao seguinte conjunto de ferramentas, disponíveis a partir da página da cadeira no fénix. Embora o sistema operativo recomendado seja o Linux, estas ferramentas também funcionam em Windows.

### FERRAMENTAS P3

#### ASSEMBLER

Aplicação que transforma o código fonte Assembly num programa a ser executado pelo processador. O código Assembly consiste numa sequência de mnemónicas que representam as instruções suportadas pelo processador (*Instruction Set*) ordenadas de forma a executar uma determinada funcionalidade.

Um programa Assembly é habitualmente separado em 3 zonas distintas: definição de constantes, definição de variáveis e o código. No entanto para programas muito simples, como os que são mostrados de seguida, pode não haver necessidade de ter as três zonas.

Para “assemblar” um ficheiro basta ter uma janela terminal aberta na diretoria/pasta onde se encontra o ficheiro fonte .as e executar: `p3as <nome do ficheiro.as>`

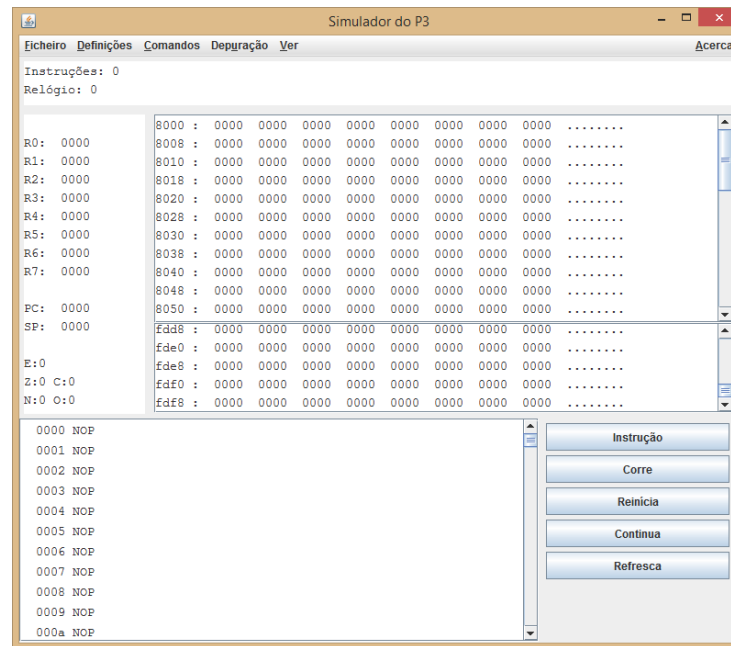
Caso esteja a usar um sistema operativo alternativo (p.ex. windows) deve usar a versão correspondente: `p3as-win.exe <nome do ficheiro.as>`

#### SIMULADOR

Aplicação que corre num computador via Java que permite correr os programas tal como num processador P3 passo-a-passo e analisar o conteúdo de todas as suas estruturas internas.

Para lançar o simulador a partir da linha de comandos (Linux/Windows) usar no diretório onde se encontra o ficheiro `p3sim.jar`: `java -jar p3sim.jar`

A janela principal do simulador tem o aspeto tal como a figura em baixo:




### 3. ESCREVER UM VALOR NUM REGISTO


Vamos introduzir o ciclo de desenvolvimento de código em Assembly através de um programa que escreve um valor num registo do P3 em Assembly.

O programa vai somente copiar o valor decimal 31 para um registo do processador. As ferramentas de desenvolvimento do P3 permitem-nos ler o conteúdo dos registos do processador e verifica o correto funcionamento (ou não) do programa.

Corra na Shell do Linux/Win: `p3as-win.exe hello.as` para produzir o executável para o P3. Para observar a execução do programa no P3 é necessário usar o simulador, e executar o programa passo-a-passo, ou instrução-a-instrução, em modo de depuração ou *debug* (tal como explicado no Tutorial do P3).

Programa ASM P3: hello.as			
	ORIG	0000h	
INICIO:	MOV	R1, 31	

 **Exercício:** Qual o valor que aparece no simulador associado ao registo R1 após a execução da instrução MOV? Qual é a representação que está a ser usada para esse valor (decimal, hexa, ou binário)?

 **Exercício:** Altere e execute a nova versão do programa para determinar qual o código ASCII do carácter 'H'. Entregue o código do novo programa e indique o número que corresponde ao código ASCII do carácter 'H'.

#### 4. OPERAÇÕES ARITMÉTICAS SOBRE VALORES GUARDADOS EM VARIÁVEIS.

As duas passagens de código abaixo representam programas equivalentes em Python e em ASM para escrever valores em variáveis e manipulá-los. Comece por estudar os programas, para entender a correspondência entre as instruções.

Programa em Python: lab3-3.py	Programa ASM P3: lab3-3.as
<pre> a = 8 b = 0x16 r = a + b print "a = %d \n" %a print "b = %d \n" %b print "r = %d \n" %r </pre>	<pre> ORIG      8000h Temp1    TAB      1 ValA     WORD     8 ValB     WORD    16h ValR     WORD     0 ORIG     0000h INICIO:  MOV      R1, M[ValA]           MOV     R2, M[ValB]           ADD     R1, R2           MOV     M[valR], R1 FIM:     BR      FIM </pre>

✂Exercício: escreva um programa em ASM P3 (ou altere o programa anterior) que tenha numa variável em memória o lado de um quadrado e calcule o respetivo perímetro e área, colocando o resultado em R1 e R2, respetivamente.

✂Exercício: escreva um programa em ASM P3 que calcule:  $x = 20 - m + (n + 40)$ . Onde x, m e n são registos do processador P3 (por exemplo, R1, R2, e R3).