LAB #2: INTRODUÇÃO AO ASSEMBLY

1. MOTIVAÇÃO E OBJECTIVOS

Neste laboratório vamos introduzir a programação do processador didático P4 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 P4.

Os programas desenvolvidos no laboratório 2 devem ser submetidos no Fénix até ao final da aula. Apenas os pontos marcados com "🛠 Exercício" são para resolver. O resto do texto serve de introdução a estes. Uma vez que existirá um exercício surpresa a ser obrigatoriamente resolvido durante a aula, recomenda-se vivamente que os restantes exercícios sejam resolvidos previamente.

2. AMBIENTES DE DESENVOLVIMENTO P4

O tutorial desta secção sobre o desenvolvimento de código Assembly P4 deve ser preparado em casa e não será avaliado no laboratório.

O ambiente de desenvolvimento P4 encontra-se disponibilizado numa versão web em

http://p4.rnl.tecnico.ulisboa.pt

Deve utilizar este ambiente para o desenvolvimento dos programas propostos no laboratório.

Para desenvolver um programa usando este ambiente, deve seguir os seguintes passos:

1) Comece por escrever o código utilizando o editor de texto disponibilizado. Pode começar com o código "esqueleto" disponibilizado em "Template" no menu dropdown, ainda que este tenha algumas funcionalidades que apenas serão lecionadas dentro de algumas semanas. Um programa Assembly é habitualmente separado em 3 zonas distintas: definição de constantes, definição de variáveis e código. No entanto para programas muito simples, como os que são mostrados de seguida, pode não haver necessidade de ter as três zonas. Para o caso dos programas desenvolvidos neste laboratório comece um ficheiro em branco, clicando no botão "Novo";
2) Assemble o código, pressionando o botão "Assemblar", que gera o ficheiro em linguagem máquina a ser executado pelo processador;
3) Carregue o programa assemblado para a memória do simulador, utilizando o botão "Carregar";
4) Simule o programa, utilizando o botão "Correr". No caso do ambiente de desenvolvimento fornecido, o botão "Correr" também assembla e carrega automaticamente o código;

SIMULADOR

O simulador permite executar o programa e observar os resultados da sua execução num dos periféricos do P4, na memória, e nos registos do processador. A execução pode ser parada utilizando o botão "Parar" e retomada utilizando o botão "Iniciar". Além disso, o programa pode ser executado passo a passo (uma instrução de cada vez), utilizando o botão "Passo". É ainda possível reiniciar a execução do programa, através do botão "Reiniciar".

Quando estiver a fazer a depuração (debug) do programa, pode ainda utilizar pontos de paragem, clicando por cima do número da linha de código no editor de texto. O simulador irá parar a execução do programa sempre que atingir esta linha. Para remover o ponto de paragem, basta clicar novamente no número da linha. Esta funcionalidade é muito útil na fase de depuração.

O simulador é formado por vários componentes. Na linha inicial é possível selecionar a frequência de funcionamento. De seguida são apresentados os periféricos da placa e os valores dos registos. Por baixo destes componentes encontra-se o programa em Assembly (convertido a partir da linguagem máquina, ou seja, "desassemblado"), a memória de programa e a memória de dados. Nestes componentes pode escolher qual é a zona de memória visualizada, utilizando uma caixa de texto.

Note que, no caso do P4, os espaços de endereçamento da memória de dados e de programa são independentes, já que existem duas memórias separadas para programa e dados, cada uma com 32 KiB (o bit mais significativo do endereço é ignorado). Assim, os dados e o programa são normalmente colocados no endereço 0000h.

3. EXERCÍCIO SURPRESA

🛠 Exercício A) O enunciado deste exercício será dado pelo professor durante o laboratório.

5. 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: lab2.py	Programa ASM P4: lab2.as
a = 8 b = 0x16 r = a + b	ORIG 0000h A WORD 8 B WORD 16h R WORD 0 INICIO: MVI R7, A LOAD R1, M[R7] MVI R7, B LOAD R2, M[R7] ADD R1, R1, R2 MVI R7, R STOR M[R7], R1 FIM: BR FIM

🛠 Exercício B) Escreva um programa em ASM P4 que dadas três variáveis X, Y e Z em memória, correspondentes aos comprimentos dos lados de um triângulo, calcule o respetivo perímetro, colocando o resultado em R1. Simule o programa.

🛠 Exercício C) Escreva um programa em ASM P4 que calcule: x = 30 – m + (2n + 10). Onde x, m e n são variáveis em memória. Comece por ler as variáveis em memória, depois faça as contas em registos e no final guarde o resultado na memória. Simule o programa e chame o docente para mostrar o resultado.

6. SUBMETER NO FÉNIX

Junte num zip os ficheiros de texto e de código com as respostas às perguntas A, B e C e submeta no sistema Fénix através da entrega de trabalhos disponibilizada para o efeito (Laboratório 2). Para criar o ficheiro zip (com o nome iac_lab2.zip) pode utilizar o comando "zip –r iac_lab2 diretoria", onde "diretoria" é a diretoria onde se encontram os ficheiros com as respostas do laboratório.