LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE COMPUTADORES

Introdução à Arquitetura de Computadores

Tutorial do P3

2015 / 2016

INSTITUTO SUPERIOR TÉCNICO
Paulo Lopes, José Costa (revisto em 2015 por Rodrigo Miragaia Rodrigues)

Conteúdo

Edição de ficheiros de texto 1
Instalar o assembler e simulador do P3 1
Assemblar e executar um programa 2
Depurar um programa Assembly (*debugging*) 3
Referências 6

Edição de ficheiros de texto

Em vários pontos deste tutorial terá de fazer a edição de ficheiros de texto. Para tal poderá escolher um editor do seu agrado (por exemplo, vi, emacs, etc.). Neste tutorial daremos uma breve introdução ao editor emacs. Este editor está incluído em quase todas as distribuições de Linux e também nalgumas versões de MacOS. No windows terá de fazer a instalação do emacs. Alternativamente, em qualquer dos sistemas operativos, pode instalar e usar outro editor, como seja o sublime ou o vi, ou, nos laboratórios, o gedit, ou, em Windows, o notepad++.

De notar que esta explicação muito breve não substitui a consulta de um tutorial do editor emacs.

Para editar um ficheiro de texto, por exemplo com o nome a.txt, deverá começar por correr o comando:

emacs a.txt <enter>

Nesta altura já pode usar as teclas com as setas nas quatro direções para movimentar o cursor, escrever novo texto e apagar o texto existente.

No emacs há duas teclas importantes para gerar comandos que permitem executar outras tarefas (como por exemplo gravar ficheiros e sair do editor): a tecla "Alt" (referida no emacs como tecla "Meta", e abreviada como M-). Por exemplo M-x quer dizer que se carrega simultaneamente no Alt e no x. A outra é a tecla Ctrl, abreviada como C-.

Os comandos essenciais para editar um ficheiro são:

Gravar o ficheiro: C-x C-s (ou seja, mantendo o Ctrl, carregar primeiro no x e depois no s).

Sair sem gravar: C-x C-c, seguido da resposta a duas confirmações.

Abrir novo ficheiro para edição numa janela emacs: C-x C-f, sendo depois perguntado qual o nome do ficheiro a abrir.

Procurar texto: C-s (para a frente) ou C-r (para trás), seguido do texto a procurar, seguido dos comandos C-s ou C-r repetidamente para procurar várias ocorrências.

Sempre que for preciso cancelar um comando que está a ser introduzido, pode-se usar a combinação C-g (Control-g) para voltar ao modo de edição do ficheiro.

Instalar o assembler e simulador do P3

Os autores do livro disponibilizam um assembler e um simulador do P3, bem como o respectivo manual, no endereço abaixo. Note que este manual é uma referência importante para o desenvolvimento de código em Assembly do P3.

http://algos.inesc-id.pt/arq-comp/?Material_Did%C3%A1tico___Processador_P3

Assemblar e executar um programa

Obtenha os ficheiros "tutorial_1.as" e "tutorial_2.as" que estão disponíveis na página da cadeira, e coloque-os na diretoria onde está o assembler.

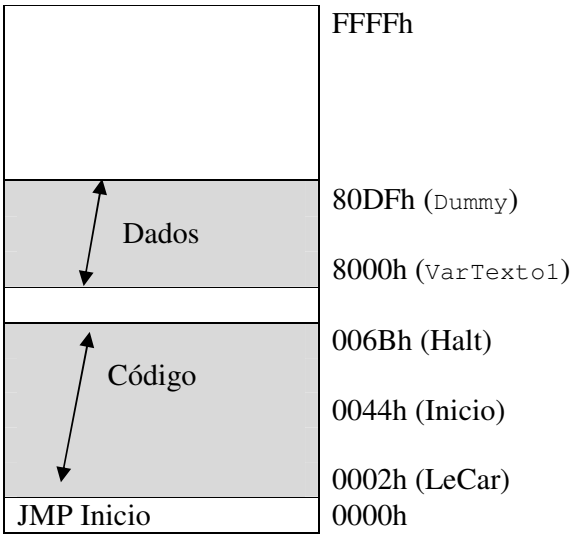
Começando pelo ficheiro "tutorial _1.as" execute as seguintes funções:

Proceda à assemblagem do programa, gerando assim o ficheiro executável (tutorial_1.exe) e o ficheiro de referências (tutorial_1.lis). Para isso proceda como se indica em seguida:

```
> p3as.sh tutorial_1.as  <enter>     obtendo:
p3as, Version 1.3, last modified Mar 20 2006
Assembling completed with success, object file: tutorial_1.exe
References file: tutorial_1.lis
```

Examine o ficheiro "tutorial_1.lis" usando o editor de texto que escolheu. Anote o valor dos endereços correspondentes às etiquetas `VarTexto1`, `Dummy`, `LeCar`, `Inicio` e `Halt`. Verifique que o mapa de memória do programa é o que se encontra em baixo.

Mapa de memória:



Utilize o simulador p3sim para testar e executar o programa. Para isso, evoque o simulador da seguinte forma:

```
> p3sim.sh   <enter>
```

Seguidamente, efetue o carregamento do programa executável tutorial_1.exe, através da execução da opção ***Carrega Programa***, existente no menu ***Ficheiro***. Verifique a alteração do conteúdo da janela de código. Analise o código desassemblado (que

encontra na janela de código) e compare-o com o do programa fonte (ficheiro "tutorial_1.as").

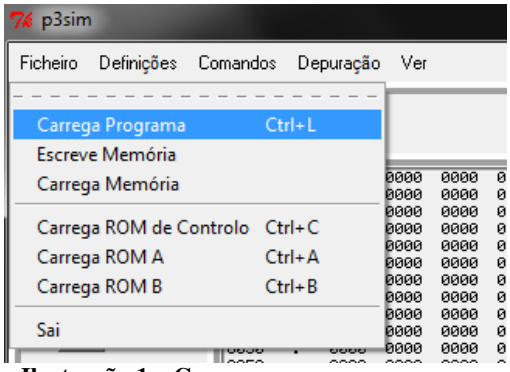


Ilustração 1 – Carrega programa.

Antes de iniciar a execução do programa, abra a janela de Entradas/Saídas. Para tal, seleccione a opção ***Janela Texto***, existente no menu ***Ver***.

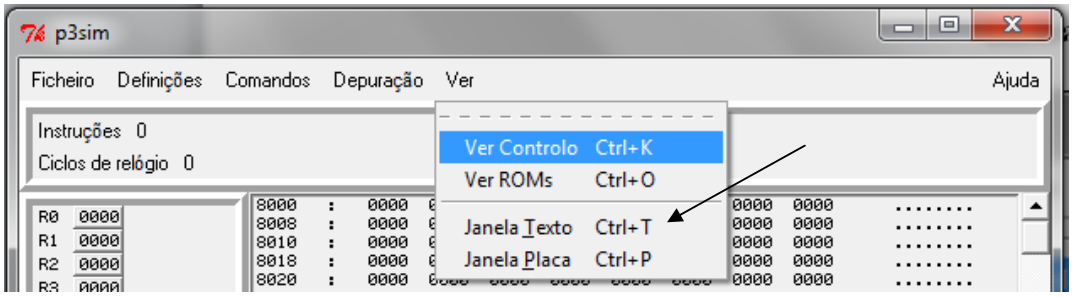


Ilustração 2 – Ver Janela de Texto

Seguidamente, execute o programa, selecionando o botão ***Corre***.

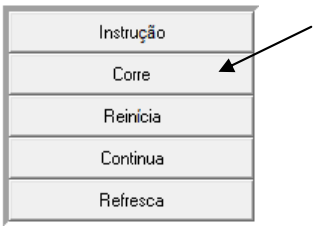


Ilustração 3 – Botão Corre.

Depurar um programa Assembly (*debugging*)

Compile o ficheiro tutorial_2.as e carregue-o para simulador. Deixe o ficheiro correr durante uns segundos e pressione em Parar.

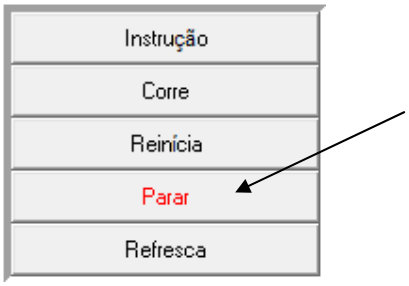


Ilustração 4 – Botão Parar.

Pode então verificar os valores dos registos R0 a R7, nomeadamente:

R0: 0000h – R0 tem sempre o valor 0

R1: 000Fh – Foi carregado com o valor de ConstUmByte que é 0Fh.

R2: FFFFh – Foi carregado com o valor 65535 em decimal que é FFFFh em hexadecimal.

R3: FFFFh – Foi carregado com o valor de R2 e por isso tem o mesmo valor que este.

R4: 0064h – Foi carregado com o valor da posição de memória 0Fh+8001h=8010h em que temos 64h (carater 'd').

R5: 000Fh – Foi carregado com o valor da posição de memória 8001h correspondente à variável VarOutroByte que foi inicializada com o valor 0Fh.

R6: AE20h – Foi carregado com o valor da posição de memória zero, pois este é o valor inicial de SP. Na posição de memória 0000h está a primeira palavra da instrução "MOV R0, LetraA", que é codificada com AE20h.

R7: 0000h – Foi carregado com o valor da posição de memória 000Ch (valor de PC) + 0Fh (ConstUmByte) = 0001Bh a qual se encontra na zona de programa não tendo sido inicializada e tendo por isso o valor zero.

Pode também ver o conteúdo da memória. Por exemplo, observe o valor que se encontra na posição 8003h da memória.

M[8003h]: 0041h - Corresponde ao valor ASCII da letra 'A' o primeiro carater da sequência de carateres de 'Arquitectura de Computadores'.

Pode também alterar o estado inicial do processador. Por exemplo, através da utilização do comando ***Escreve Registo***, do menu ***Depuração***, inicialize os registos R1 a R7 com o valor FFFFh. Utilizando agora o comando ***Escreve Memória***, do mesmo menu, inicialize as posições de memória correspondentes às variáveis `VarUmByte` e `VarOutroByte`, também com o valor FFFFh.

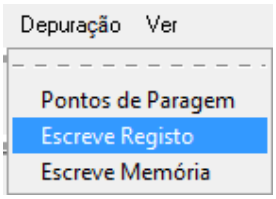


Ilustração 5 – Escreve Registo e Escreve memória.

Repita a execução do programa utilizando o comando ***Reinicia*** seguido de ***Corre***. Compare a informação dos registos com os valores anteriores.

Deverá obter:

R0: 0000h – Igual.
R1: 000Fh – Igual.
R2: FFFFh – Igual.

R3: FFFFh – Igual.
R4: 0064h – Igual.
R5: FFFFh – É carregado com o valor da posição de memória 8001h que desta vez é inicializada com o valor FFFFh.
R6: AE20h – Igual.
R7: 0000h – Igual.

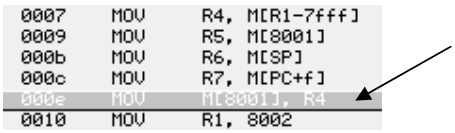
Vamos de seguida exemplificar a inserção de pontos de paragem (breakpoints).

Acrescente no final do programa referido, antes da instrução Halt: BR Halt, os seguintes conjuntos de instruções:

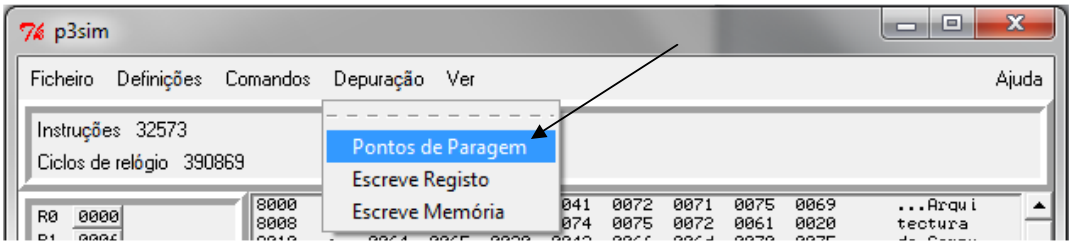
a) MOV M[`VarOutroByte`], R4
b) MOV R1, `VarUmaWord`
   MOV M[R1], R6
c) MOV R1, M[`VarOutroByte`]
   MOV M[`VarUmByte`], R1

Compile novamente o programa e carregue para o simulador. Coloque um ponto de paragem em a). Para colocar o ponto de paragem faça:

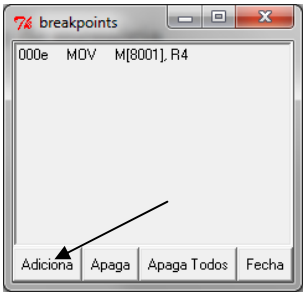
Selecione a instrução no endereço 000Eh, que corresponde á instrução a), na janela com o programa desassemblado.



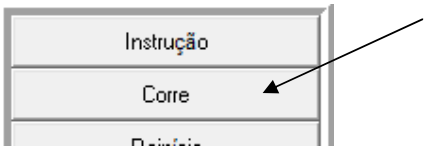
Selecione pontos Depuração→Pontos de paragem do Menu,



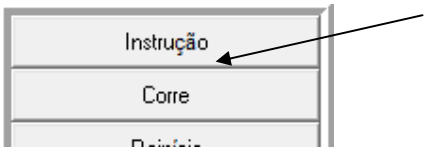
Na caixa de diálogo que surge no ecrã seleciona Adiciona.



Mande correr o programa até ao ponto de paragem.



Corra cada uma das instruções passo a passo utilizando o botão Instrução,



Verifique o efeito de cada um dos conjuntos de instruções. Em particular, o programa:

a) Utiliza o modo de endereçamento direto para copiar o conteúdo do registo R4 para a posição de memória `VarOutroByte`.
b) Utiliza o modo de endereçamento indireto por registo para copiar o conteúdo do registo R6 para a posição de memória `VarUmaWord`. Começa por colocar a posição de memória no registo R1.
c) Utiliza duas instruções de endereçamento direto para copiar o conteúdo da posição de memória `VarOutroByte` para a posição de memória `VarUmByte`. Utiliza o registo R1 como registo auxiliar.

Referências

[1] G.Arroz, J.C.Monteiro, A.Oliveira, "Manual do Simulador do P3", IST, 2003