LABORATÓRIOS MICROPROCESSADORES

KIT 8051 – Depuração e simulação.

Outubro 2006

1. Objectivo

Apresentar um guia básico de como usar o Keil µVision3 para simular e depurar programas em *assembly*.

2. Descrição

O ambiente µVision3 permite três formas diferentes de simulação:

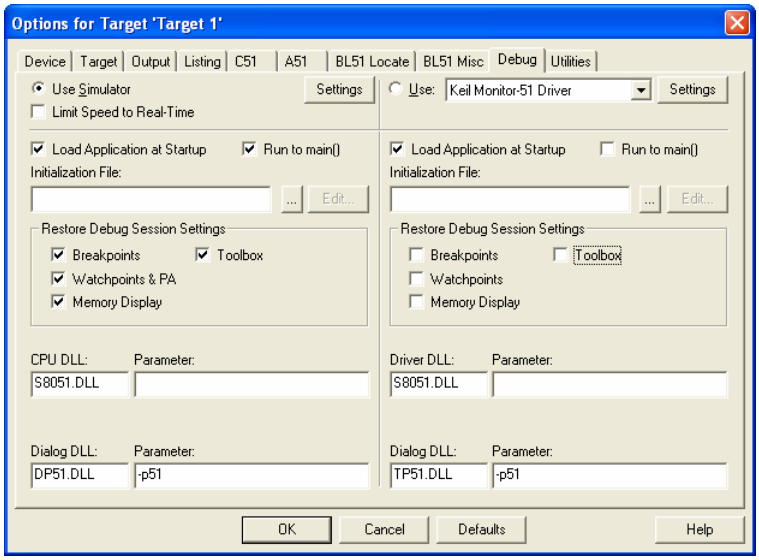
a) Simulação passo-a-passo,

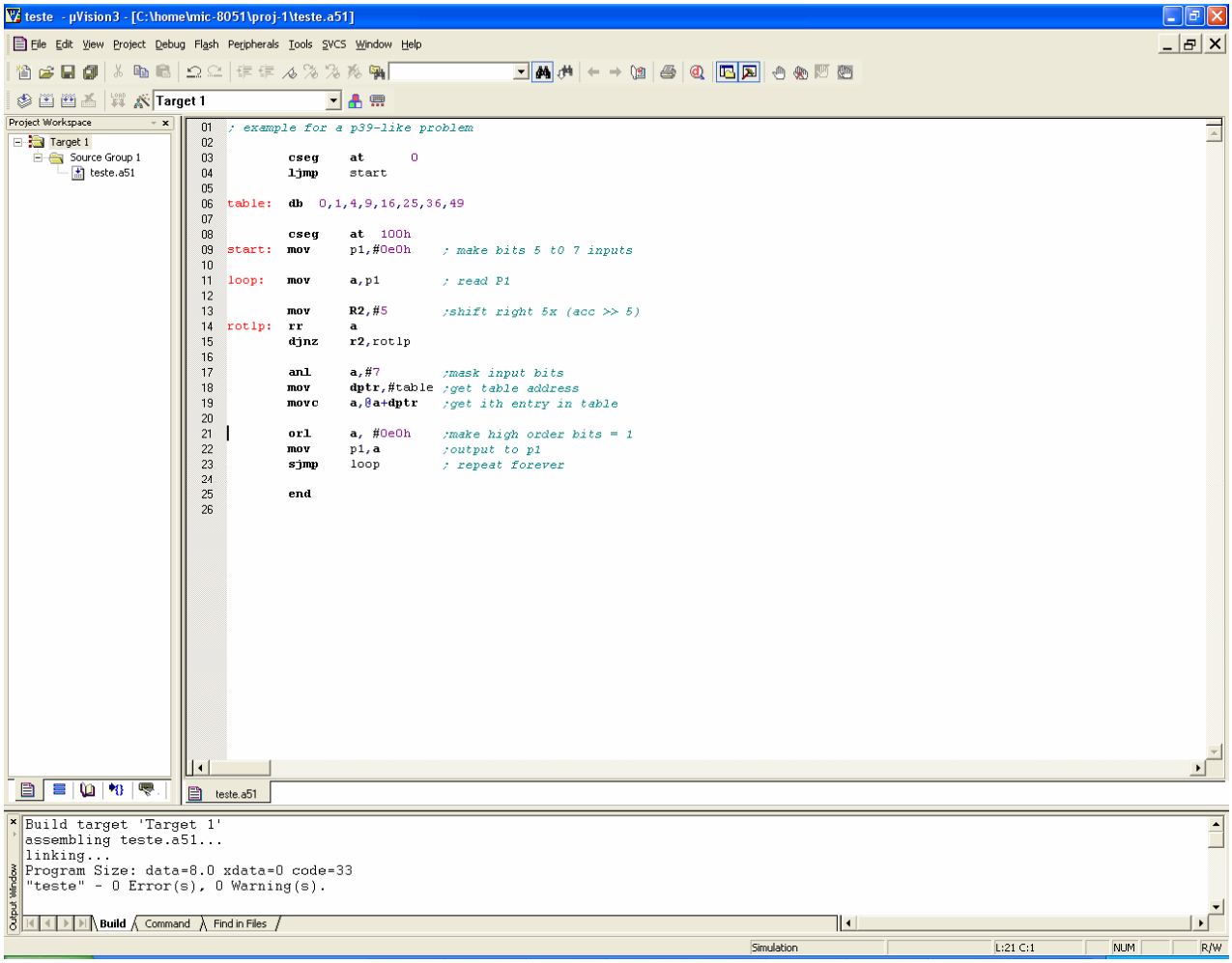
b) Simulação contínua com *breakpoints*,

c) Simulação contínua com actualização periódica do estado seleccionado do microcontrolador.

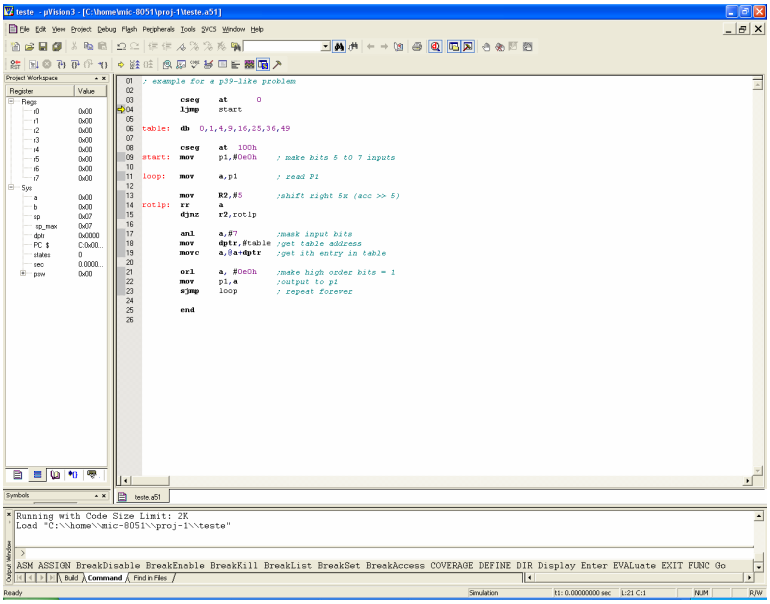
2.1 Criar um projecto

Crie e "assemble" o projecto abaixo, certificando que a configuração do *debug* é a seguinte:



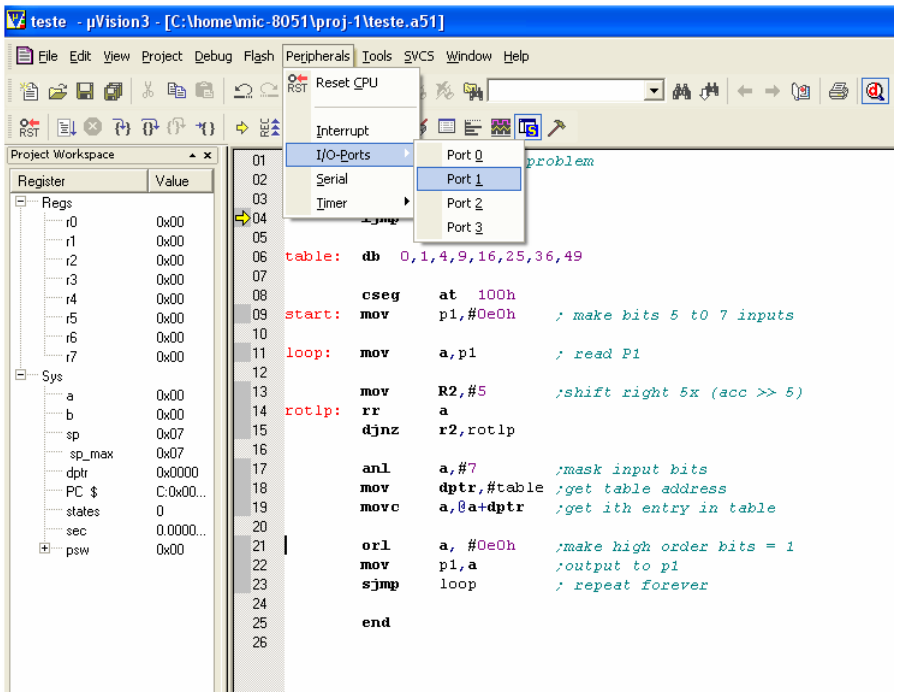


2.2 Active a janela de depuração "clicando" no *icon debug* e aparecerá a seguinte janela com a visualização dos registos:

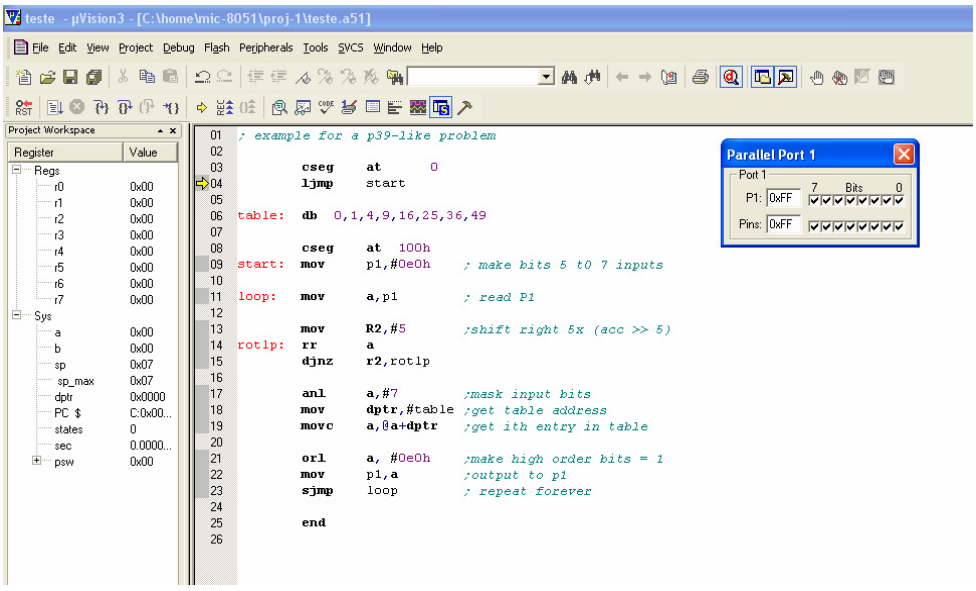


A seta amarela indica a próxima instrução a ser executada (‘*ljmp start*’).

2.3 Abra a janela do porto 1 (P1), seleccionando o item do menu ‘peripherals/IO Port/Port 1’



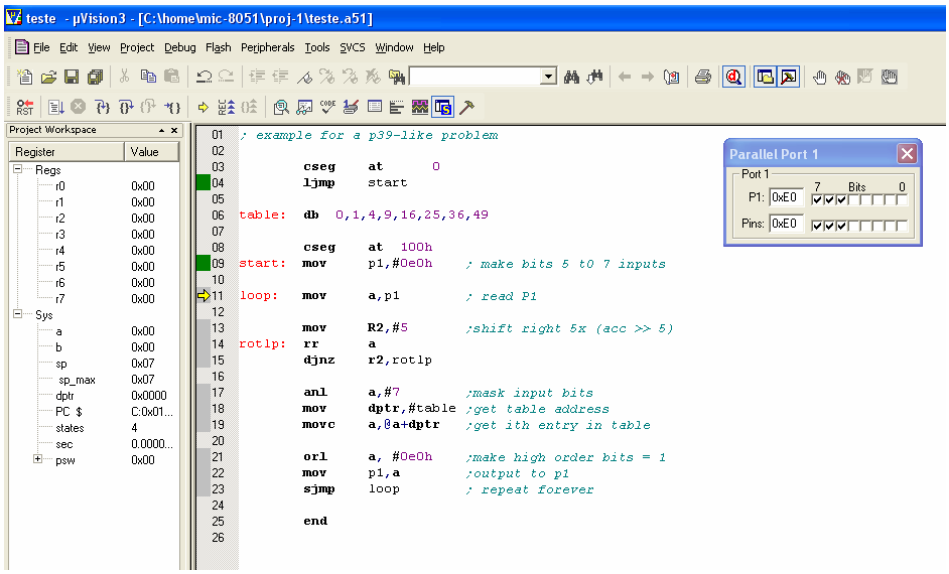
e visualizará a seguinte janela:



2.4 Pressione F11 (depuração passo-a-passo) duas vezes e verifique:

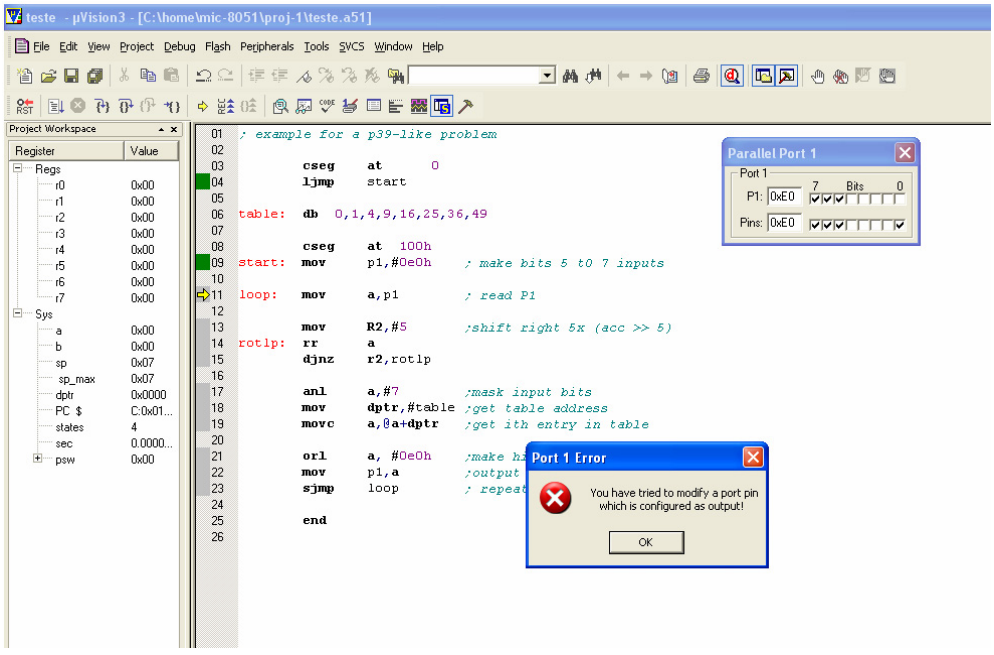
a) A localização da seta amarela moveu para a etiqueta ‘*loop*’ e as duas instruções que foram executadas ficaram com uma marca a verde.

b) A janela do porto 1 apresenta os três *bits* mais significativos a ‘1’ e os restantes a ‘0’.

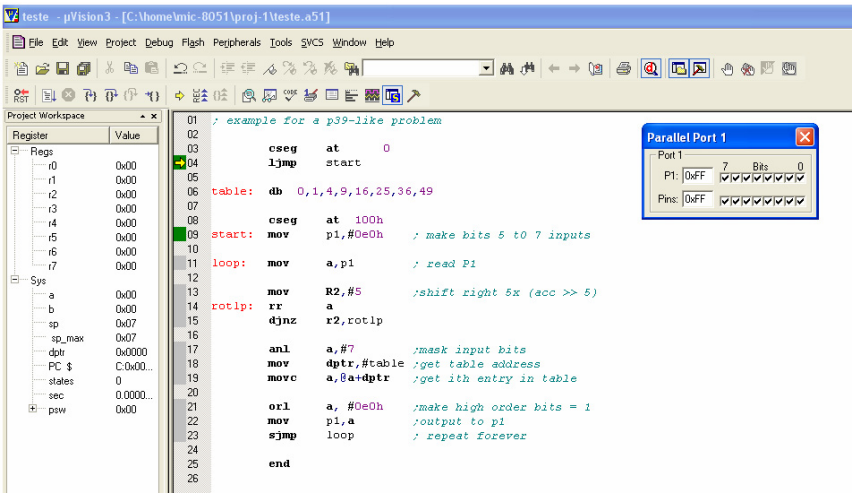


Nota:

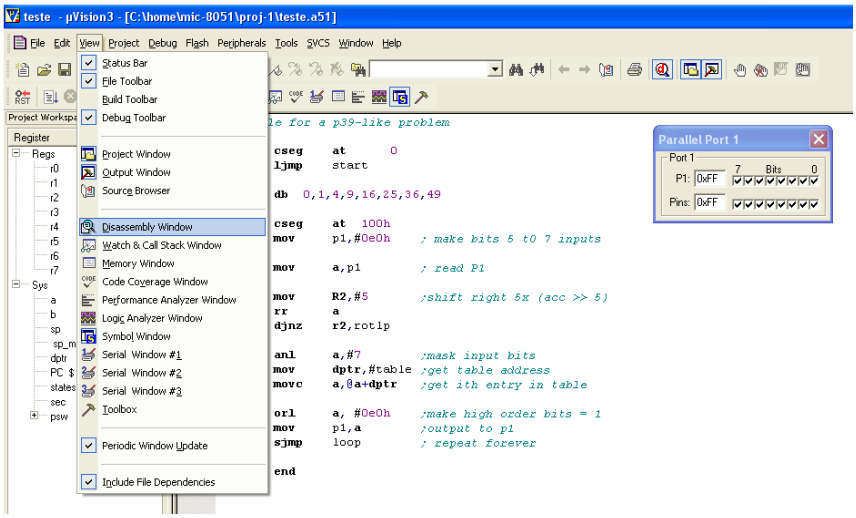
Consideremos o pino 0 que está configurado como pino de saída, logo se tentar alterar o estado do pino externamente ('clicando' sobre o pino P1.0) receberá uma mensagem de erro. Numa implementação física, este procedimento resultaria na danificação do Porto 1.



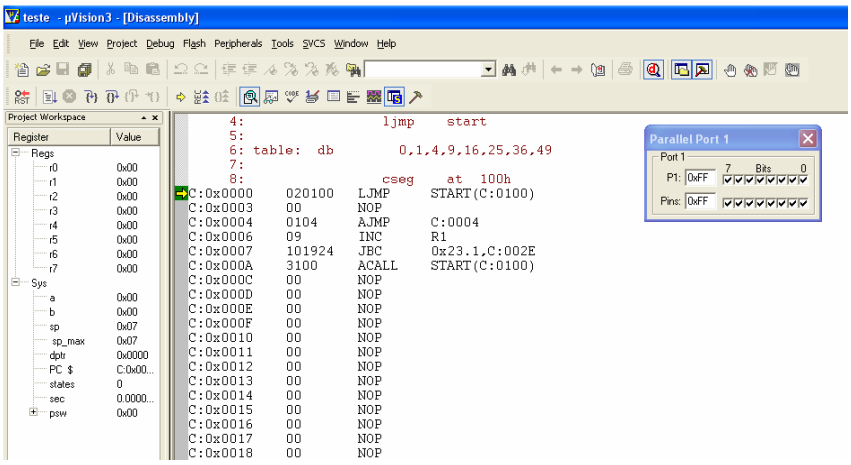
2.5 Pressione o *icon* de *reset*, RST, para repor o estado inicial do processador.



2.6 Seleccione a janela 'view/disassembly window'



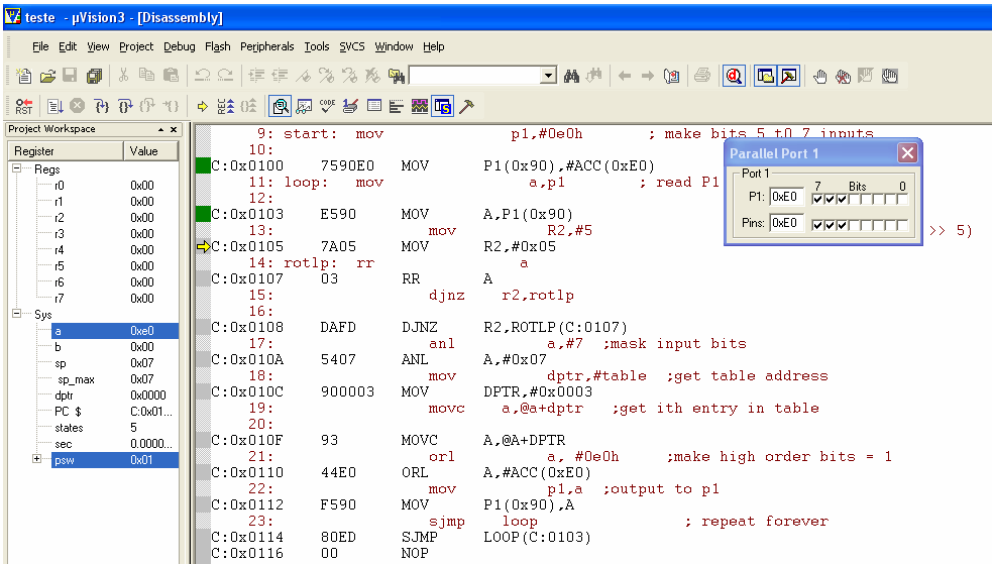
que visualizará o código *assembly* juntamente com o código máquina.



Repare que a localização C:0000H contém o valor 02H, o C:0001H contém 01H, o C:0002H contém 00H ... Repare de igual modo que os *bytes* da tabela, 'table', foram cegamente "disassembladas" como se fossem instruções.

2.7 Ajuste a janela de registos de modo a que o acumulador seja visível, caso este esteja escondido.

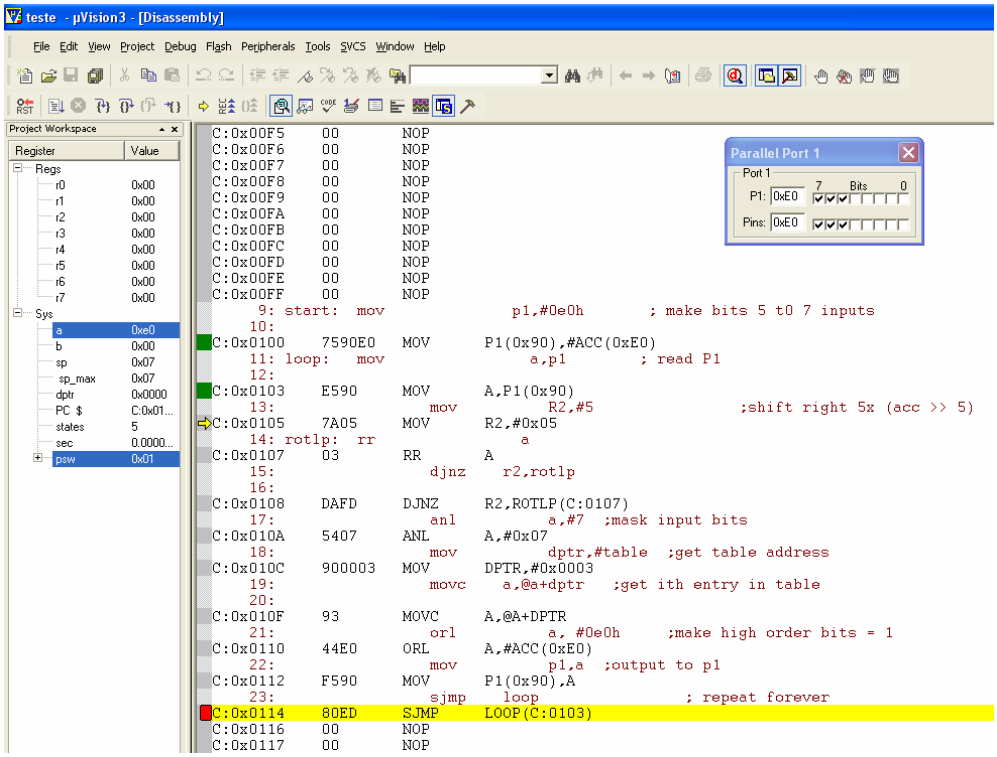
2.8 Pressione F11 até que a instrução '*mov*' na localização C:0103H seja executada.



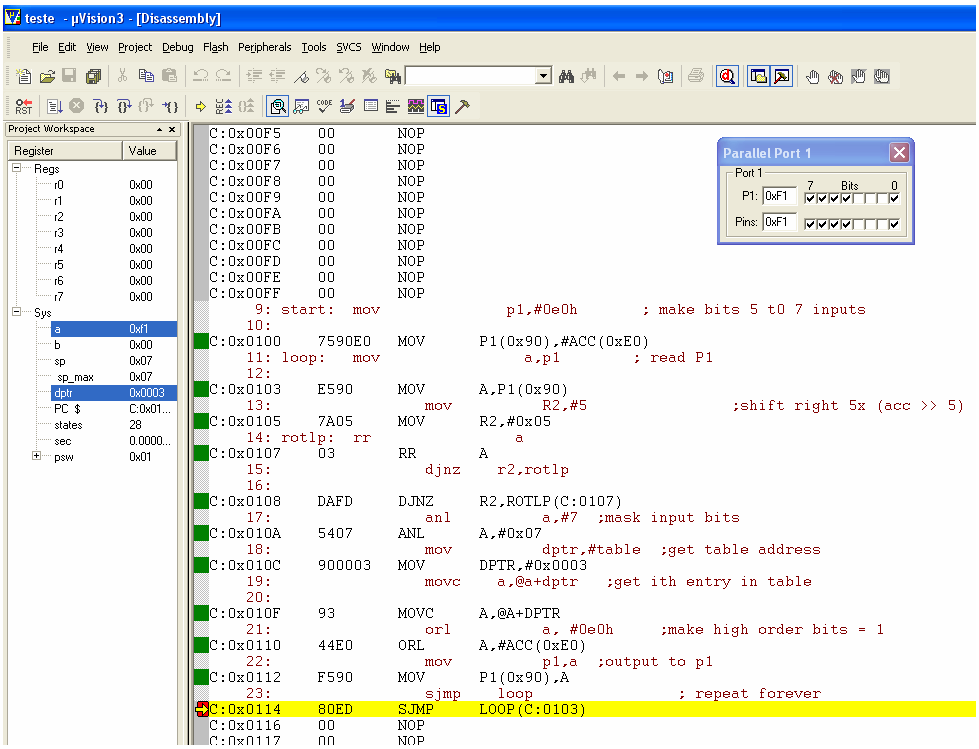
Repare que o registo acumulador teve o seu conteúdo alterado para E0H e ficou colorido a azul (os registos cujos conteúdos foram alterados ao longo da execução são coloridos a azul).

Poderia continuar a pressionar F11 e executar uma instrução de cada vez, mas isto seria contraproducente quando temos programas com milhares de linhas de código – neste caso a melhor estratégia seria o uso de *breakpoints*.

2.9 *Double click* sobre a instrução '*sjmp*' na janela de *disassembly* e aparecerá uma marca vermelha na margem esquerda que indica a inserção de um *breakpoint*.



Se tentar executar o programa no modo contínuo (F5) este parará a execução quando atingir o *breakpoint* – neste caso a localização da seta amarela coincidiria com a localização do '*sjmp*'.

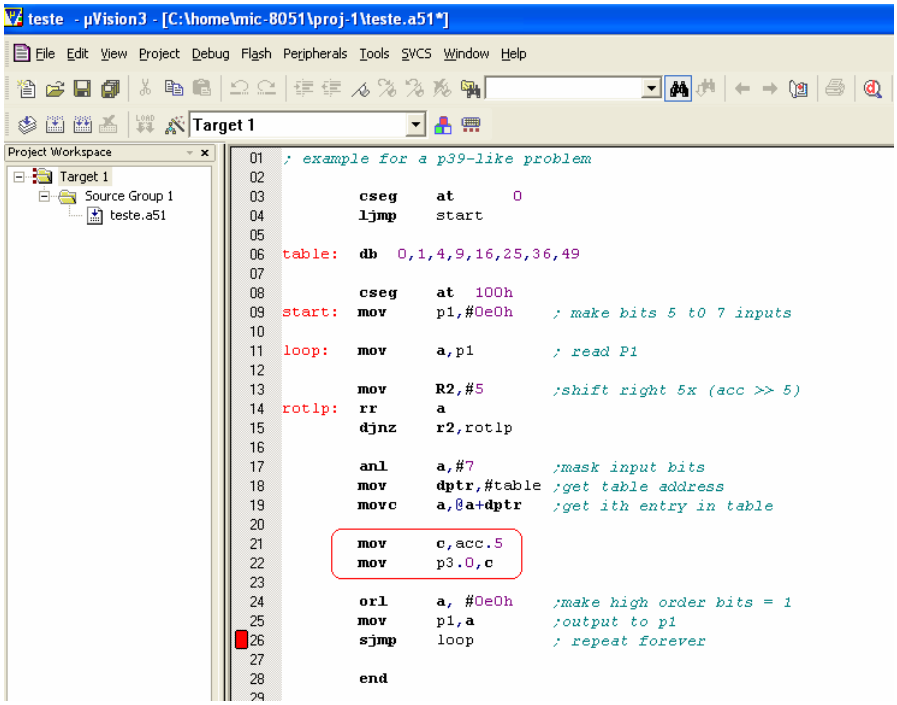


Suponhamos que chegou a conclusão que a simulação revelou uma inconsistência entre o valor previsto e o pretendido. Imagine que a correcção necessária consistiria na execução de duas novas instruções:

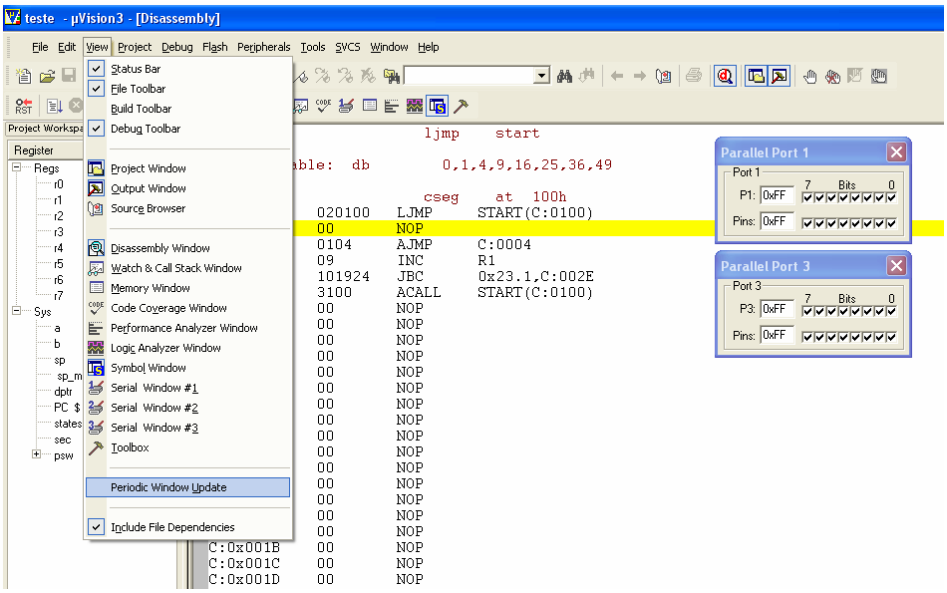
```
mov   c, acc.5
mov   p3.0, c
```

antes da execução da instrução 'orl' na localização C:0110H. **Como alteraria o código anterior?**

2.10 "Clique" o *icon debug*,  , para voltar ao modo de edição e adicione as duas instrução ao código fonte.



"Assemble" a aplicação como anteriormente indicado e volte ao modo de depuração. Abra agora ambas janelas para o porto 1 e para o porto 3. Em vez de executar o código passo-a-passo ou no modo contínuo com breakpoints, active o item de menu 'view/periodic window update'.



Neste modo o código é executado continuamente (F5) e periodicamente o estado dos portos (ou de qualquer outro periférico seleccionado) será actualizado. Experimente alterar o estado dos *bits* 5-7 do porto 1 e verifique o que ocorre.

Nota:

Repare que poderia extrair algumas informações úteis consultando o ficheiro '.lst'. Por exemplo, passe para o modo de edição e seleccione 'File/Open' e abra o ficheiro com extensão '.lst'.