



# MICROCONTROLADORES

## Guia 2

**KIT 8051 – Sistema de Desenvolvimento 8051**

**Keil  $\mu$ Vision 5: Depuração e Simulação**

**Autores:**

**Revisão: Jorge Cabral, José Mendes**

# 1 Objectivos

Familiarização com o ambiente de desenvolvimento Keil  $\mu$ Vision5. Simulação e depuração de programas simples em linguagem *assembly*.

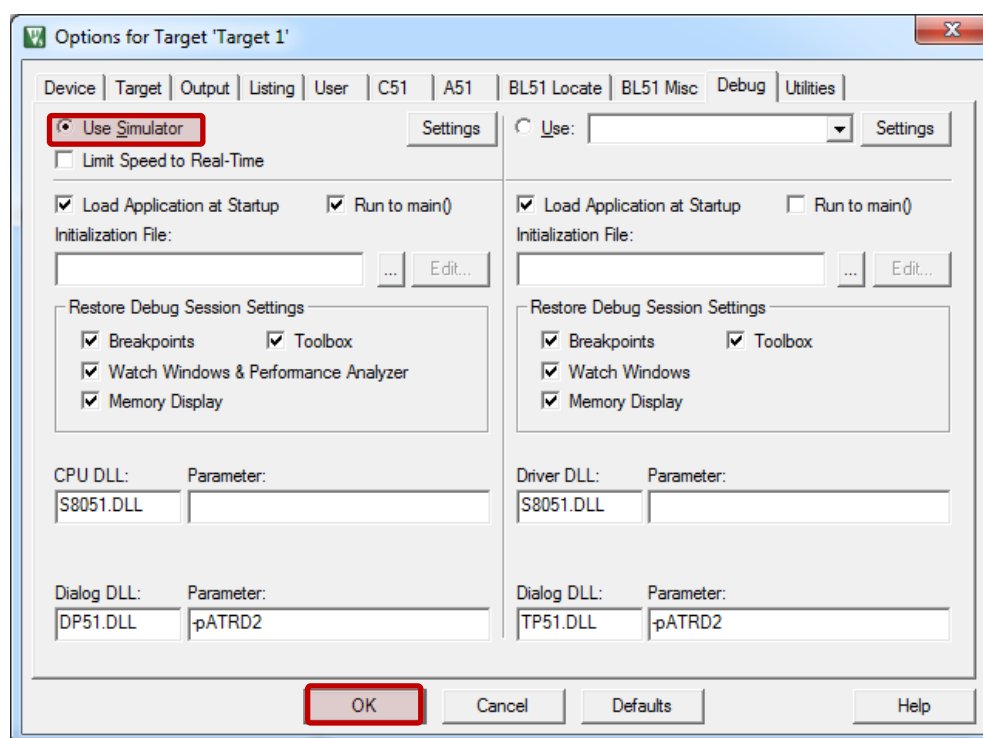
## 1.1 Descrição

O ambiente  $\mu$ Vision5 permite duas formas distintas de simulação:

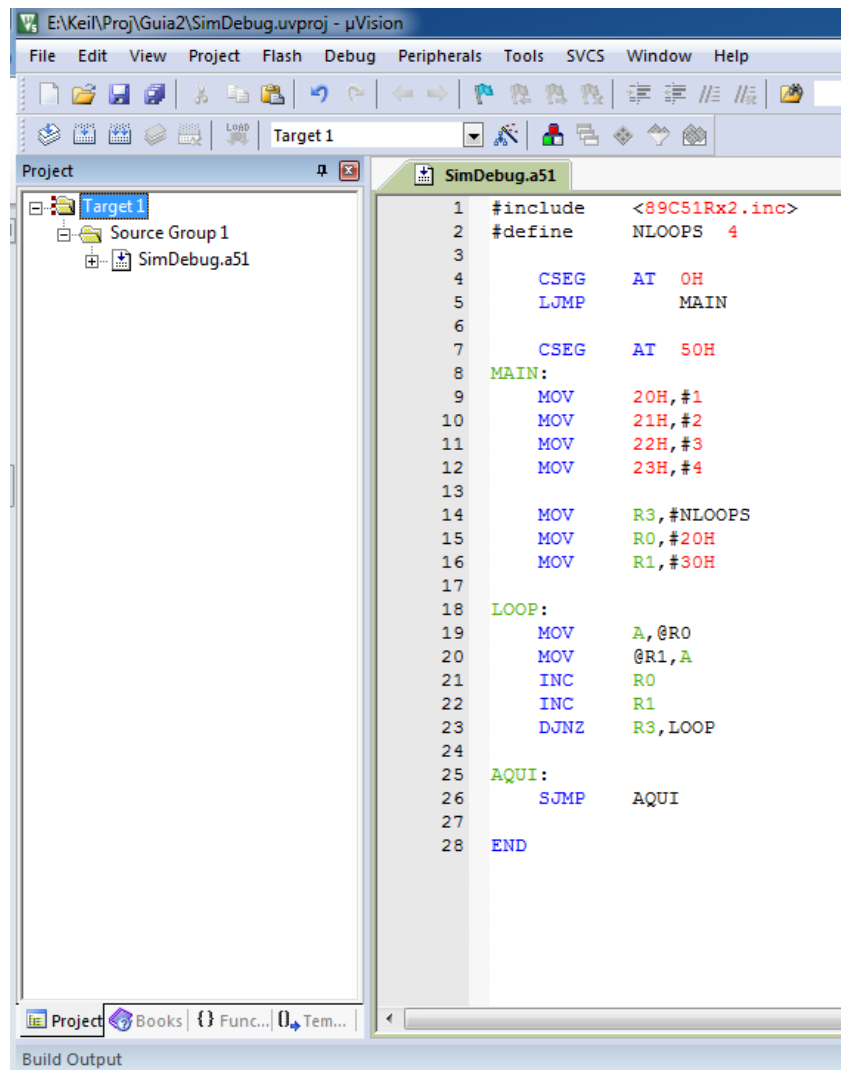
- a) Simulação passo-a-passo,
- b) Simulação contínua com *breakpoints*,


## 1.2 Criar um projecto

Crie e “assemble” o projecto que se segue recorrendo ao guia anterior. Insira a frequência de relógio correcta (12MHz) e certifique-se que a configuração do tab *Debug* é a seguinte:

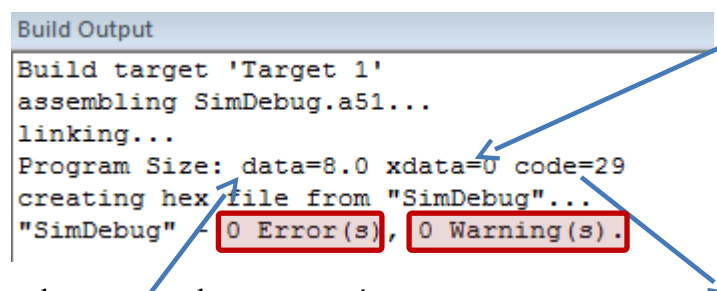


Esta configuração permitirá a utilização do simulador do microcontrolador 8051. Este simulador está incorporado na ferramenta de desenvolvimento Keil  $\mu$ Vision5.



Depois de terminar a edição do código proceda à assemblagem do mesmo através do menu Project-> Built target (F7) (ou a opção Rebuilt all target files) ou ainda através dos *icons* .


Quando a assemblagem do código finalizar, na janela de Build Output deverá aparecer:

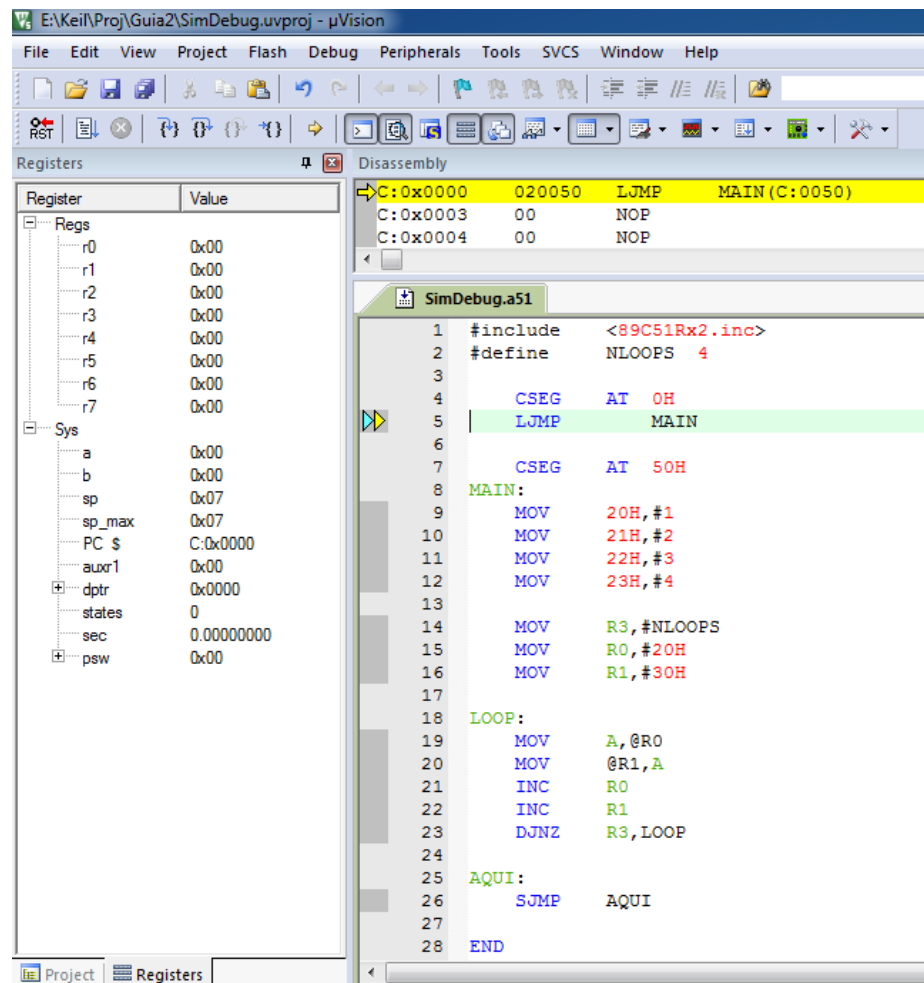


Garanta que o número de erros e o de *warnings* é zero.

Justifique a informação indicada por *Program Size*.

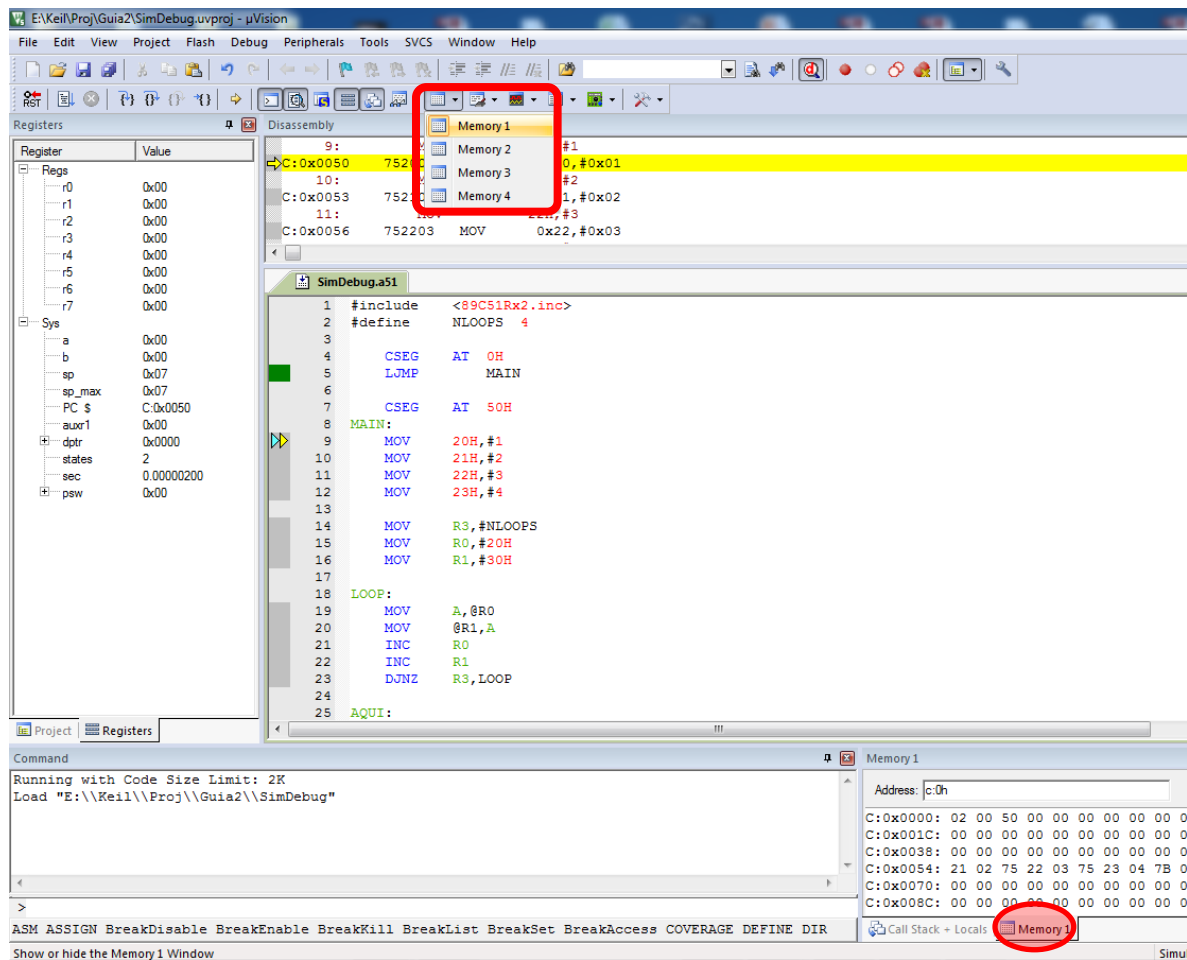
## Depuração Passo a Passo

Active a janela de depuração “clcando” no *icon debug*  e aparecerá a janela representada na figura seguinte. No lado esquerdo é possível visualizar todos os registos do 8051, no topo a janela de *disassembly* que mostra a memória de código e o seu conteúdo (opcodes – códigos da instrução), na janela principal o programa a ser simulado, na barra no fundo é visível o tempo de execução. Há uma série de outras janelas que pode configurar e utilizar para simplificar o processo de validação do programa de código.

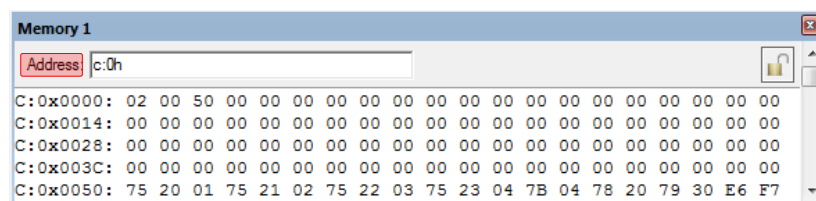


A seta amarela indica a próxima instrução a ser executada ('LJMP MAIN'). Prima F11.

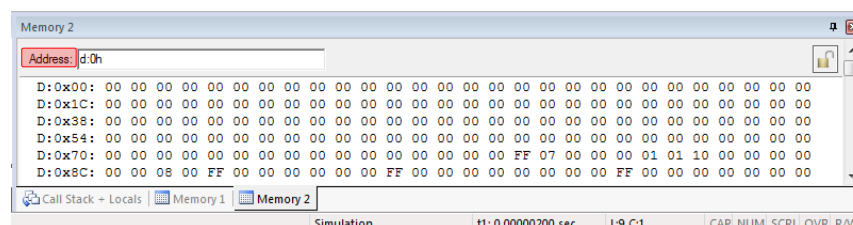
Abra a janela de visualização do conteúdo das memórias, seleccionar icon da “Memory Windows”, como indicado na figura seguinte.



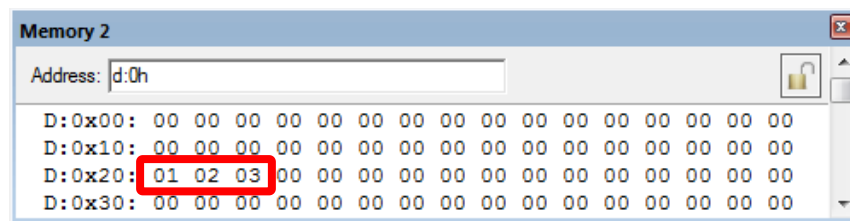
e visualizará a seguinte janela, após escrever no campo “Address” o texto “c:0h”:



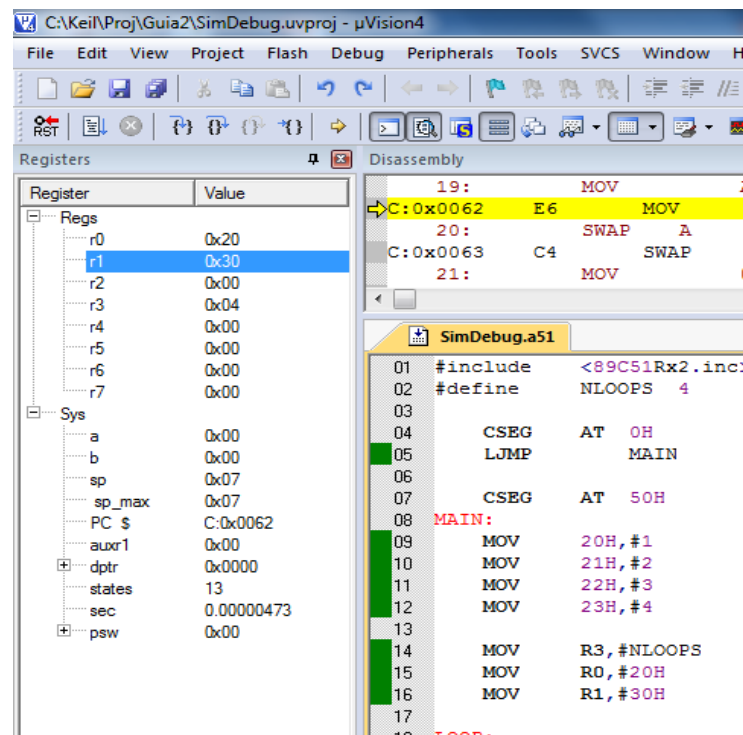
Repita o procedimento e acrescente a janela “Memory #2” e escreva no campo “Address” o texto “d:0h”, deverá obter o seguinte resultado:



Pressione F11 (depuração passo-a-passo) três vezes e verifique que a posição da seta amarela é alterada e que as últimas instruções ficaram com uma marca a verde. Repare ainda que os conteúdos da janela “Memory #2” foram alterados.



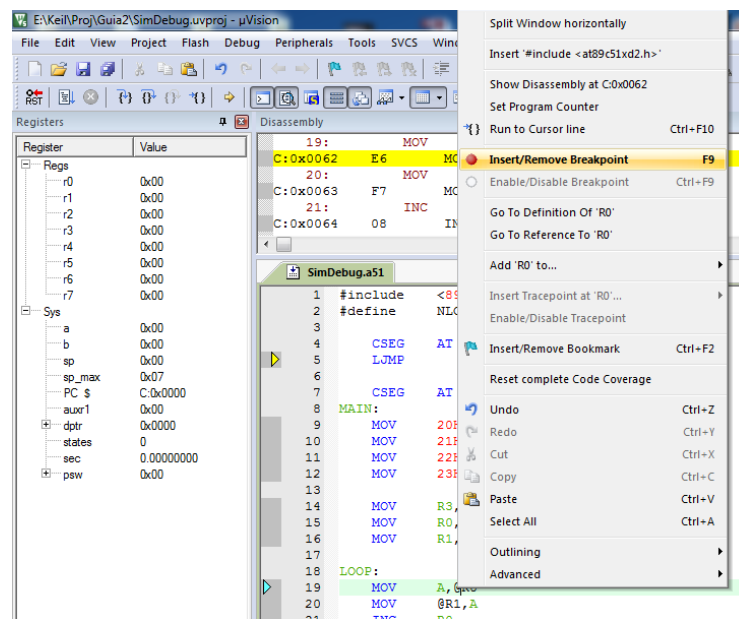
Pressione F11 de novo até que a seta amarela esteja a apontar para a instrução que se encontra após a etiqueta (*label*) “**LOOP:**”. Durante este processo tente compreender o que está a acontecer em termos de mudanças na memória #2, assim como nos valores dos registos R0, R1 e R3.



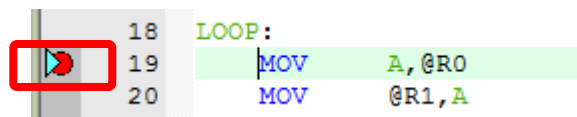
## Depuração com Breakpoints

Saia do modo de depuração, “stop debug” e reinicie o modo debug, “start debug”. Repare que os registos R0, R1, R3 e os conteúdos da memória de dados #2 (d:0h) têm o valor zero.

Coloque o rato sobre a instrução imediatamente após a etiqueta “**LOOP:**” e active o menu com o botão do lado direito do rato, seleccionando a opção “Insert/Remove Breakpoint” (F9).



Como resultado desta acção deve visualizar-se uma marca vermelha ao lado da instrução “loop: mov...”



De forma a entender a depuração por Breakpoints, execute o comando “Run” ou F5.



Observe o sucedido, estamos na mesma situação que na depuração passo a passo, mas com a vantagem de termos premido apenas uma tecla F5 para que todas as instruções do programa sejam executadas até à instrução onde colocamos o *breakpoint*. Como compreenderá estes passos são muito mais relevantes num projecto com muitas mais instruções, evitando perdas de tempo muito significativas na depuração do código.

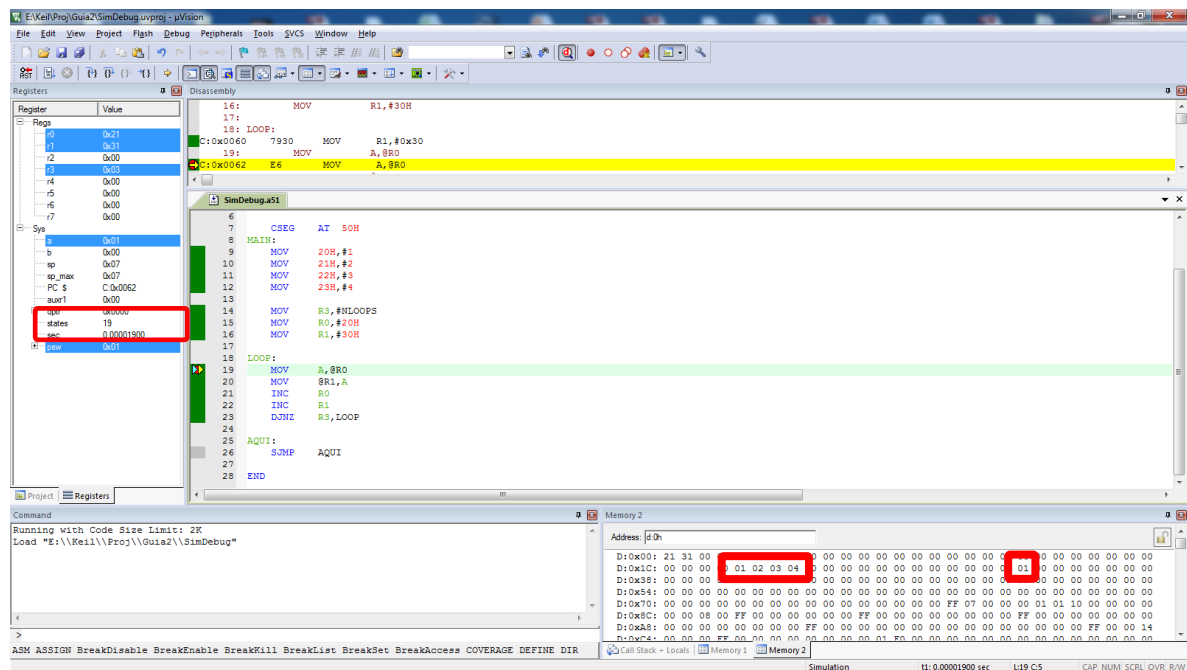
Explore também o comando  CTRL+F10 ou “Run” to Cursor.

Os breakpoints podem ser adicionados, removidos ou mesmo desactivados.

Execute novamente o comando “Run” ou prima F5. Deverá visualizar no computador um resultado equivalente ao apresentado na figura seguinte.

Foram executadas todas as instruções seguintes até à instrução “DJNZ R3, LOOP” e a seta amarela volta a estar localizada na linha do breakpoint, isto é, na instrução “MOV A,@R0”. O que é que aconteceu? Que alterações sofreram os registos e a memória #2?

Repare nas alterações no tempo de execução t1 e no valor da variável interna *states*.



Tente perceber o funcionamento do programa e peça ajuda ao docente caso seja necessário. Volte a usar a depuração passo a passo, premindo para isso F11.

Analise com muito detalhe e atenção que alterações provocam a execução de **cada** instrução (se for necessário, saia do modo *debug*, volte a entrar em modo *debug* e repita todos os passos anteriores).

Verifique que registos e que posições de memória são alteradas. É fulcral que entenda cada uma destas alterações.

Modifique o programa de modo a colocar no porto I/O (Entrada/Saída) P1 o conteúdo da posição de memória apontado por R1 negado e valide as alterações fazendo o *debug* com a visualização do conteúdo do porto I/O P1.

Após a execução de todos os passos anteriores e de ter assimilado todos os passos e alterações produzidas por cada instrução, explique qual o objectivo do programa em assembler apresentado. Como passo intermédio comente cada uma das linhas do programa:

MOV 20h, #01h ;move a constante 1h para o endereço 20h da memória de dados interna

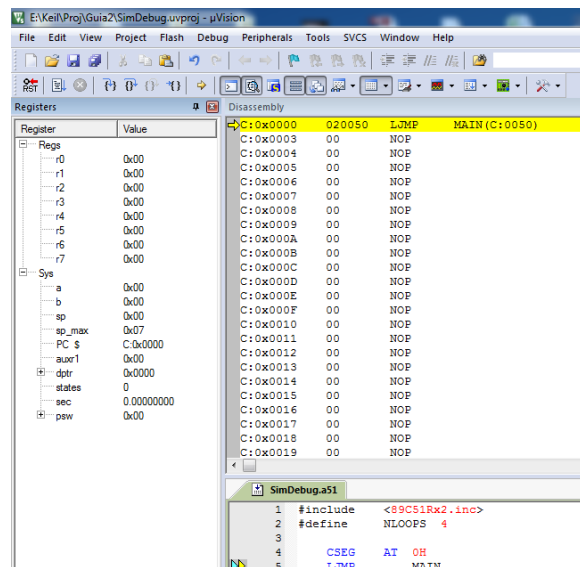
## Código Máquina





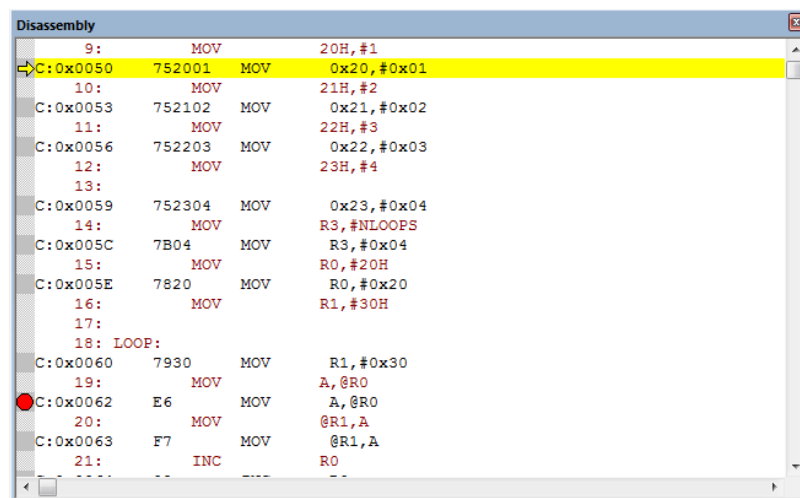
Pare e reinicie a depuração.

Aumente as dimensões da janela “Disassembly”



que lhe permite visualizar o código assembler juntamente com o correspondente código máquina. Veja com atenção o seu código e o código máquina.

Prima F11 uma única vez:



Interprete a informação apresentada quer as linhas quer as colunas, por exemplo:

9:	MOV	20H, #01
C:0x0050	752001	MOV 0x20,#0x01
10:	MOV	21H, #02
C:0x0053	752102	MOV 0x21,#0x02
11:	MOV	22H, #03
C:0x0056	752203	MOV 0x22,#0x03

O que significa “C:0x0050”? O que significa “752001”? Consulte a ajuda do compilador e analise a imagem que se segue extraída do “Atmel 8051 Microcontrollers Hardware Manual”:

**MOV direct,#data**

Bytes: 3

Cycles: 2

Encoding: 

0	1	1	1	0	1	0	1
---	---	---	---	---	---	---	---

direct address

immediate data

Operation: MOV  
(direct) ← #data

Na janela de memória, selecione a *Memory 3* e no campo “Address:” e escreva c:0x050, que observa?

O que significa o valor “020050” apresentado a partir da posição de memória c:0x00?

Interprete a informação apresentada quer ao nível de linhas quer de colunas.

**Repare que poderia extrair algumas informações extra úteis consultando o ficheiro ‘.lst’. Por exemplo, passe para o modo de edição e seleccione ‘File/Open’ e abra o ficheiro com extensão ‘.lst’. Consulte ainda o ficheiro com extensão ‘.M51’.**