



MICROCONTROLADORES

Guia 4

Codificação de Fluxogramas

Autores:

Jorge Cabral, José Mendes

1 Objectivo

Apresentar um guia básico de como codificar em linguagem *assembly* fluxogramas. Neste guia, o fluxograma que soluciona o exercício do slide A04-6 será codificado para *assembly* do MCS-51 e iremos usar o Keil μ Vision 4 para simular e depurar programas em *assembly* com mais que um ficheiro de código fonte.

1.1 Codificação do fluxograma

O fluxograma, que soluciona o exercício proposto no slide A05-3, foi já estudado e apresentado na aula teórica.

O fluxograma e a respectiva conversão para linguagem *assembly* são apresentados na Figura 1, que se segue:

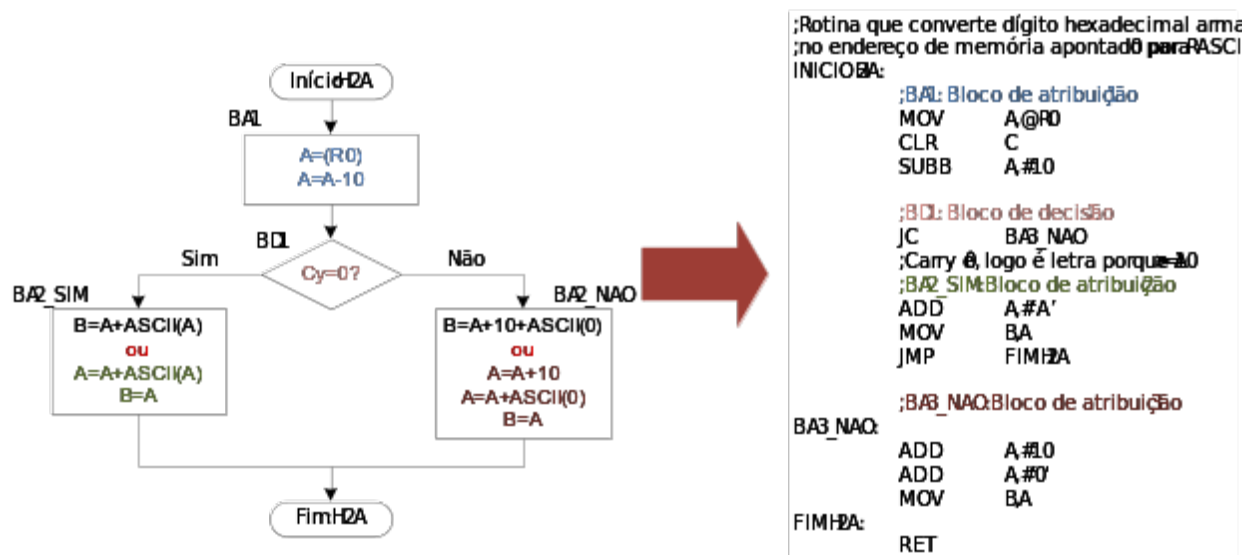


Figura 1 – Fluxograma e respectiva codificação

Salientar apenas que o fluxograma foi codificado como sendo uma subrotina, cuja variável de entrada é o registo R0 que contém o endereço da memória de dados interna onde está armazenado o dígito hexadecimal a converter e a variável de saída é o registo B. A subrotina utiliza ainda o registo Acumulador como registo auxiliar.

1.2 Descrição

O ambiente μ Vision 5 permite três formas diferentes de simulação:

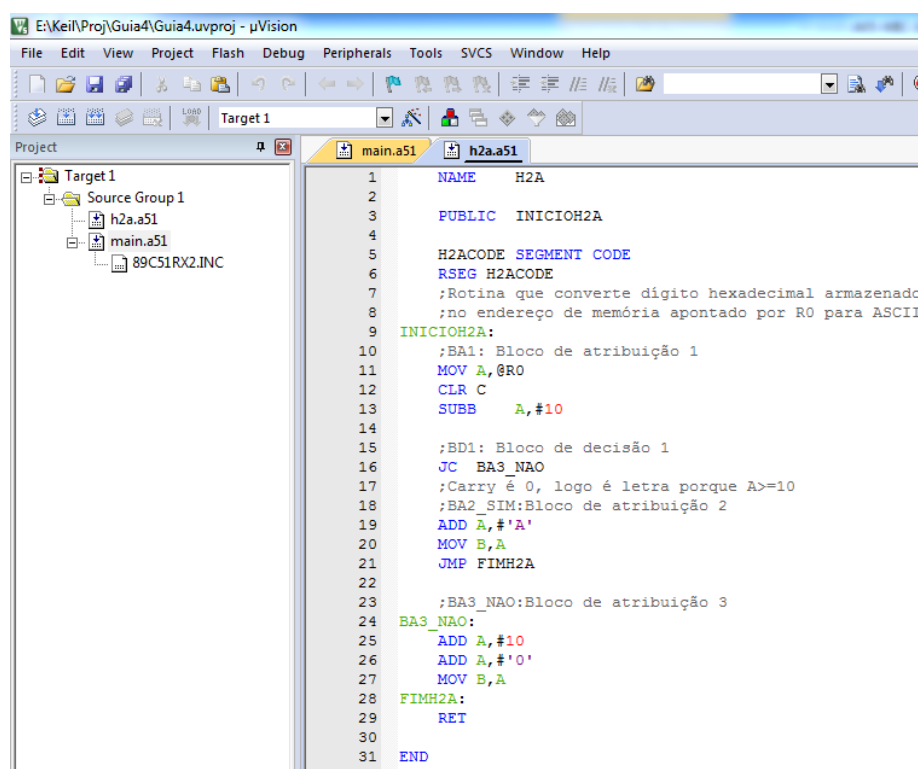
- a) Simulação passo-a-passo,
- b) Simulação contínua com *breakpoints*,

Crie e “assemble” o projecto Guia4.

Seguir os passos habituais para criação de um projecto mas agora com dois ficheiros de código: h2a.a51 e main.a51.

No ficheiro h2a.a51 está colocado o código *assembly* que implementa a subrotina que permite converter um dígito hexadecimal para ASCII. O código *assembly* presente neste ficheiro é o apresentado na Figura 1, com a excepção de algumas diretivas para o *assembler* da família MCS-51. O ficheiro h2a.a51 é apresentado na Figura 2.

No ficheiro main.a51 está o código principal, onde se armazenam os valores dos dígitos hexadecimais na memória e se invoca, através de um *loop*, a subrotina implementada no ficheiro h2a.a51, guardando o resultado da conversão de dígito hexadecimal para o correspondente ASCII na memória de dados externa. O ficheiro main.a51 é apresentado na Figura 3.



```
1 NAME H2A
2
3 PUBLIC INICIOH2A
4
5 H2ACODE SEGMENT CODE
6 RSEG H2ACODE
7 ;Rotina que converte dígito hexadecimal armazenado
8 ;no endereço de memória apontado por R0 para ASCII
9 INICIOH2A:
10 ;BA1: Bloco de atribuição 1
11 MOV A,@R0
12 CLR C
13 SUBB A,#10
14
15 ;BD1: Bloco de decisão 1
16 JC BA3_NAO
17 ;Carry é 0, logo é letra porque A>=10
18 ;BA2_SIM: Bloco de atribuição 2
19 ADD A,#'A'
20 MOV B,A
21 JMP FIMH2A
22
23 ;BA3_NAO: Bloco de atribuição 3
24 BA3_NAO:
25 ADD A,#10
26 ADD A,#'0'
27 MOV B,A
28 FIMH2A:
29 RET
30
31 END
```

Figura 2 – Ficheiro h2a.a51

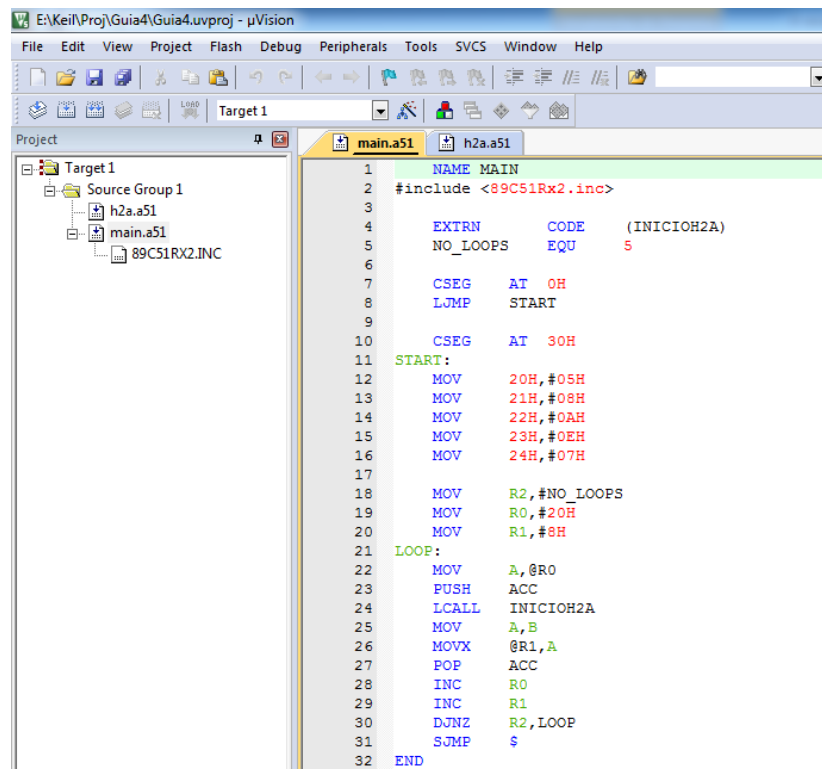


Figura 3 - Ficheiro main.a51

Identifique as novidades no exemplo acima apresentado.

Observe os conteúdos da memória de programa, estabeleça as ligações entre os locais dos códigos.

Use a janela de *disassembly*.

Efectue a depuração passo a passo;

Esteja atento às alterações da memória de dados, interna e externa do microcontrolador.

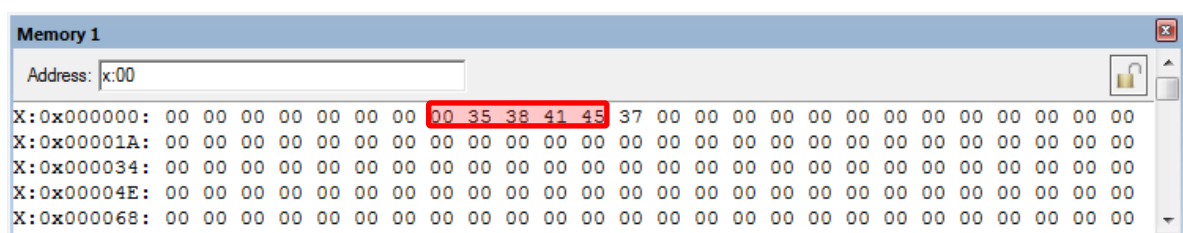


Figura 4 – Janela de memória

Qual o objectivo de desenvolver este programa em dois módulos?

Explique as diretivas: PUBLIC; EXTRN; SEGMENT e RSEG

Qual o endereço na memória de código das etiquetas BA3_NAO e FIM_H2A?

Repare que pode extrair algumas informações úteis consultando os ficheiros gerados pelo Keil. Por exemplo, passe para o modo de edição e seleccione 'File/Open' e abra os ficheiros com extensão '.lst' explicando o seu conteúdo.

Qual o objetivo de incluir a seguinte linha?
`#include <89C51Rx2.inc>`

Consulte ainda o ficheiro com extensão '.M51'.

Que informação tem disponível?

1.3 Leitura de valores da memória de código

No programa que implementou, as posições 20h a 24h da memória de dados interna foram inicializadas com os valores a converter para ASCII.

Recorrendo à directiva DB, escreva a partir da posição de memória 1000H os seguintes valores:
10; 7; 12; 3;15;10;15;14;8;7;0

Modifique o programa anterior para que este leia cada um destes valores e invoque a rotina H2A para os converter para hexadecimal.