Guião de Laboratório
de
Arquitectura de Computadores

Simulação 4.13 – Expressões booleanas

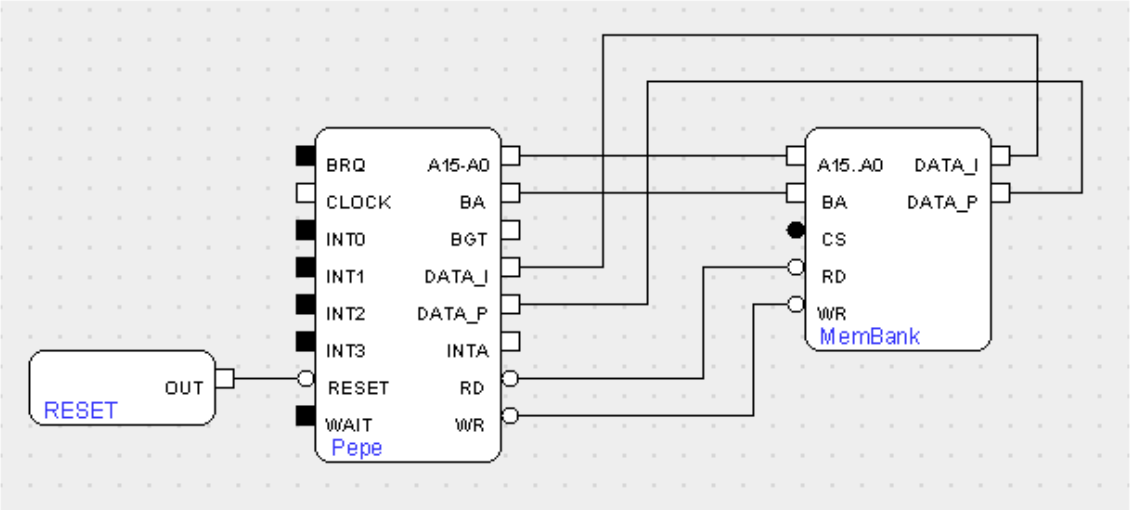
1 – Objectivos

Esta simulação ilustra o funcionamento das expressões booleanas, tendo por base o Programa 4.14 e o Programa 4.15. Os aspectos cobertos incluem os seguintes:

- Execução passo a passo e com pontos de paragem do programa;
- Verificação do conteúdo da memória e da existência de duas células de memória seguidas com o valor zero (se não existirem, têm de se criar alterando directamente o conteúdo de uma ou duas células de memória);
- Verificação da evolução dos registos relevantes e da memória, iteração a iteração.

2 – Circuito

O ficheiro "pepe.cmod" implementa o circuito da Fig. 4.7. A simulação 4.1 contém indicações mais detalhadas sobre a sua utilização no simulador.



3 – Simulação do programa 4.14

Carregue este circuito no simulador e passe para Simulação.

Abra o painel do PEPE e compile e carregue ( ) o ficheiro "programa4-14.asm".

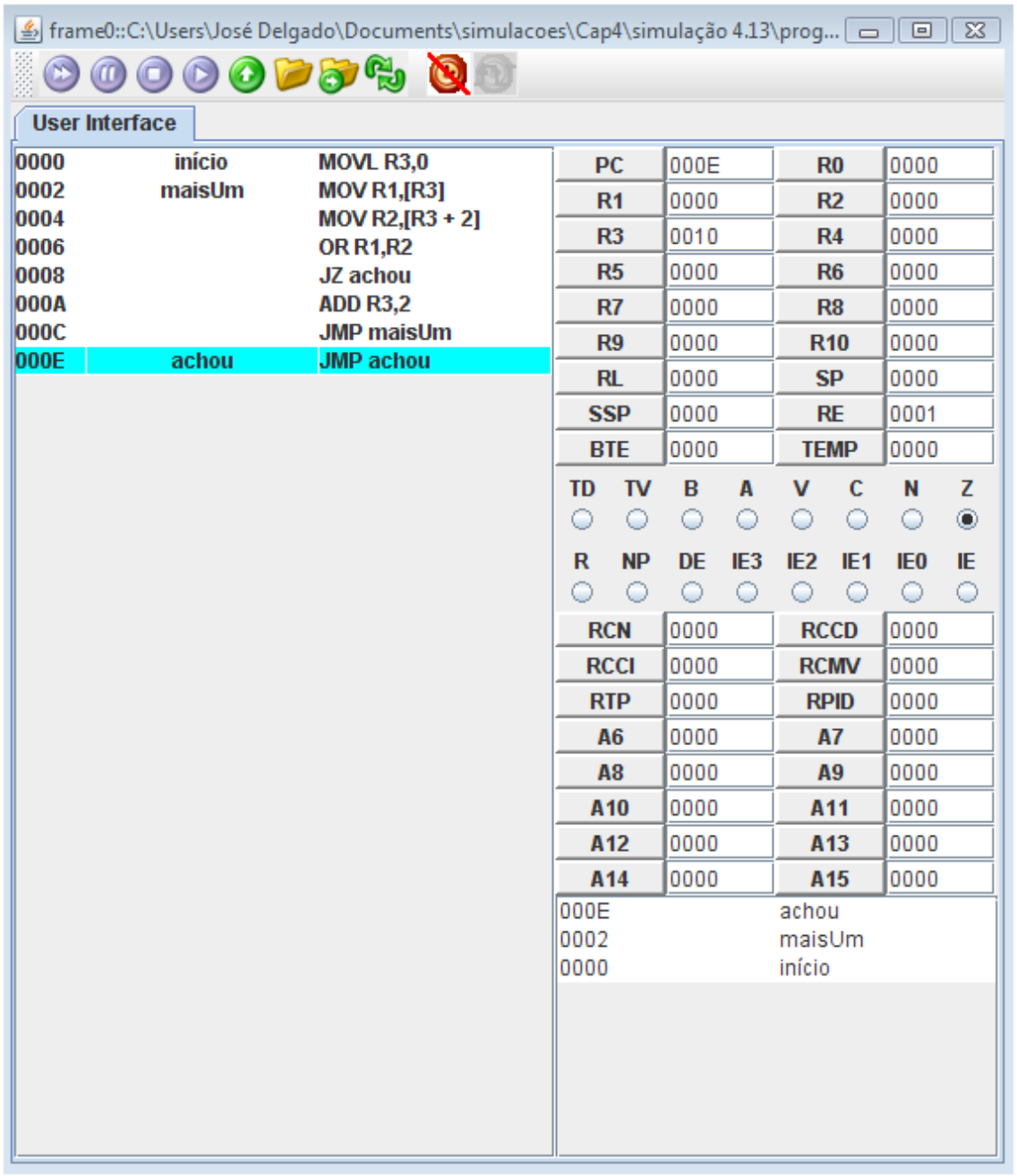
Execute as instruções passo a passo (botão) e vá vendo os registos, percebendo o que o programa vai fazendo, tomando em conta os comentários.

Imediatamente antes da execução da instrução OR R1, R2, anote os valores dos registos R1 e R2. Verifique que, após a instrução, o valor no R1 apresenta o resultado correcto da operação OR.

Depois da primeira iteração, ou quando já tiver percebido o funcionamento, coloque um ponto de paragem na última instrução do programa e passe para modo de execução contínua (botão ). Desta forma, poderá chegar mais rapidamente ao fim do programa.

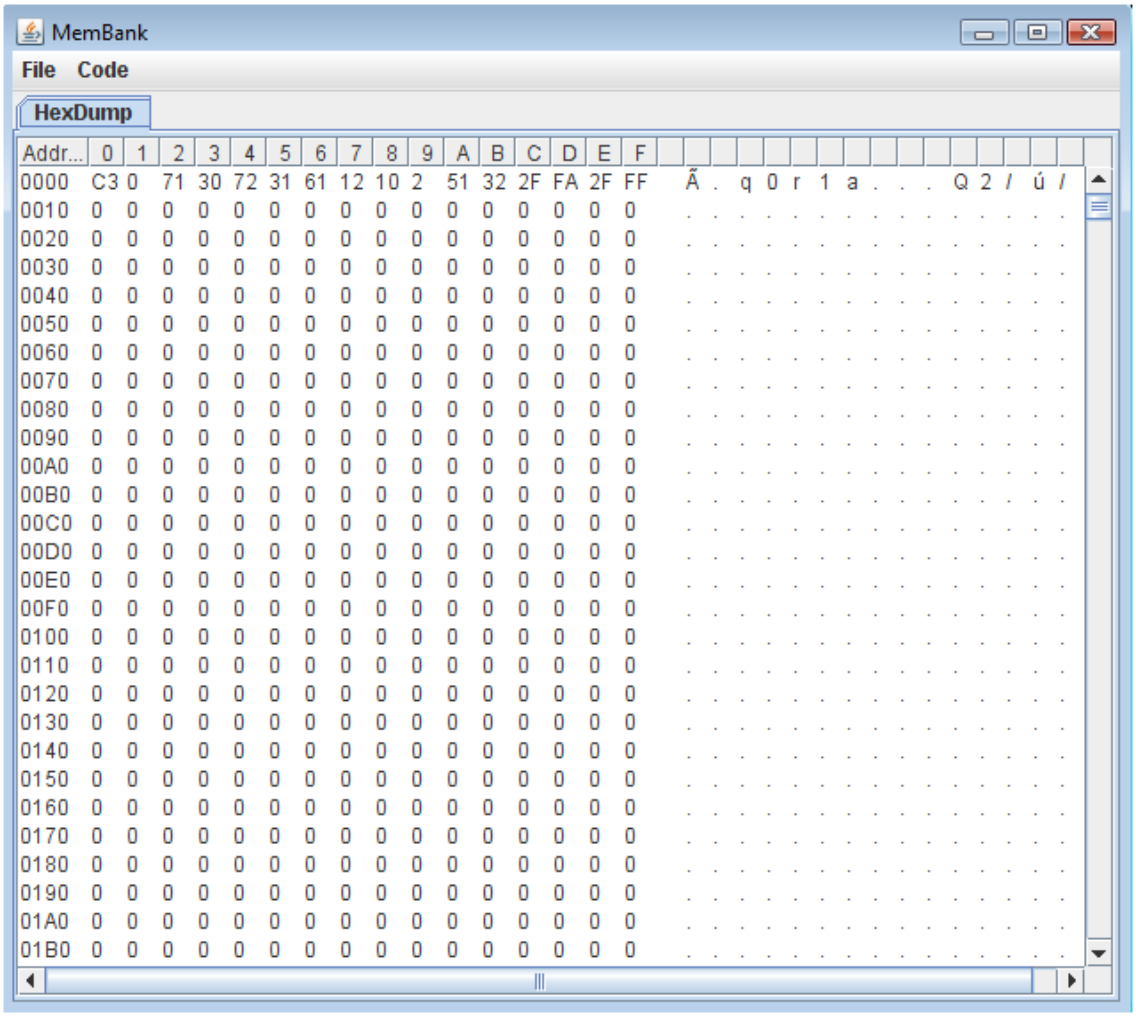
Pode também colocar pontos de paragem intermédios onde pretender inspeccionar o estado do programa.

Quando o programa terminar, terá este aspecto:



Note-se que R3 indica o endereço 0010H, que deverá assim ser a primeira palavra com o valor 0000H seguida de outra palavra cujo valor também é 0000H.

Tal pode verificar-se abrindo o painel da memória. O programa ocupa os endereços 0000H a 000FH.



Como variante, e para testar a instrução AND, substitua o OR no programa 4.14 por um AND, recompile e execute. Esta variante já está disponível no ficheiro "programa4.14AND.asm".

Execute as instruções passo a passo (botão ) e imediatamente antes da execução da instrução AND R1, R2, verifique os valores dos registos. Verifique que, após a instrução, o valor no registo destino apresenta o resultado correcto da operação AND.

Neste exemplo, o resultado em R3 será ligeiramente diferente, neste caso 000EH, pois este é a primeira palavra cujo AND pela palavra no endereço seguinte dá zero (isto porque a palavra no endereço 0010H é zero). Mas calhou. Nada impedia de haver um AND de duas palavras anteriores que desse zero, sendo ambas as palavras diferentes de zero (bastava os 1s numa palavra e noutra estarem desencontrados, em posições diferentes).

4 – Simulação do programa 4.15

Carregue este circuito no simulador e passe para Simulação.

Abra o painel do PEPE e compile e carregue ( ) o ficheiro “programa4-15.asm”.

Execute as instruções passo a passo (botão) e vá vendo os registos, percebendo o que o programa vai fazendo, tomando em conta os comentários.

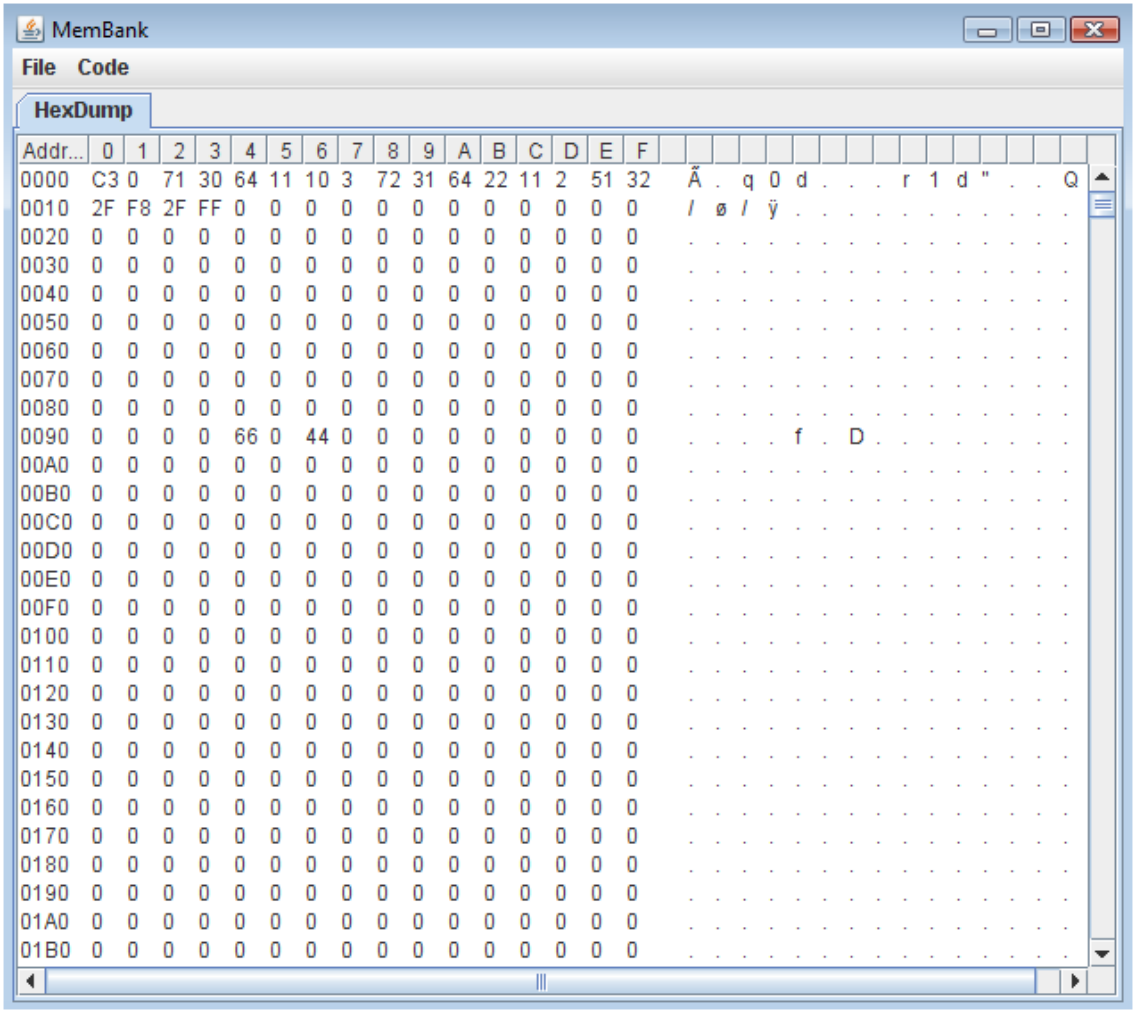
Imediatamente antes da execução das instruções TEST R1, R1 e TEST R2, R2, anote os valores dos registos (R1 ou R2) e dos bits de estado. Verifique que, após a instrução, o registos não mudaram de valor e o bit de estado está activo se o registo for zero.

Neste caso, o programa pára logo na primeira iteração, pois as duas primeiras palavras são diferentes de zero.

Por isso, coloque o PC a zero (faça enter) e execute uma instrução em passo a passo (botão). Coloque 20H em R3 (ou outro endereço, desde que seja depois do fim do programa). Desta forma, o programa começará a procurar com as palavras a zero.

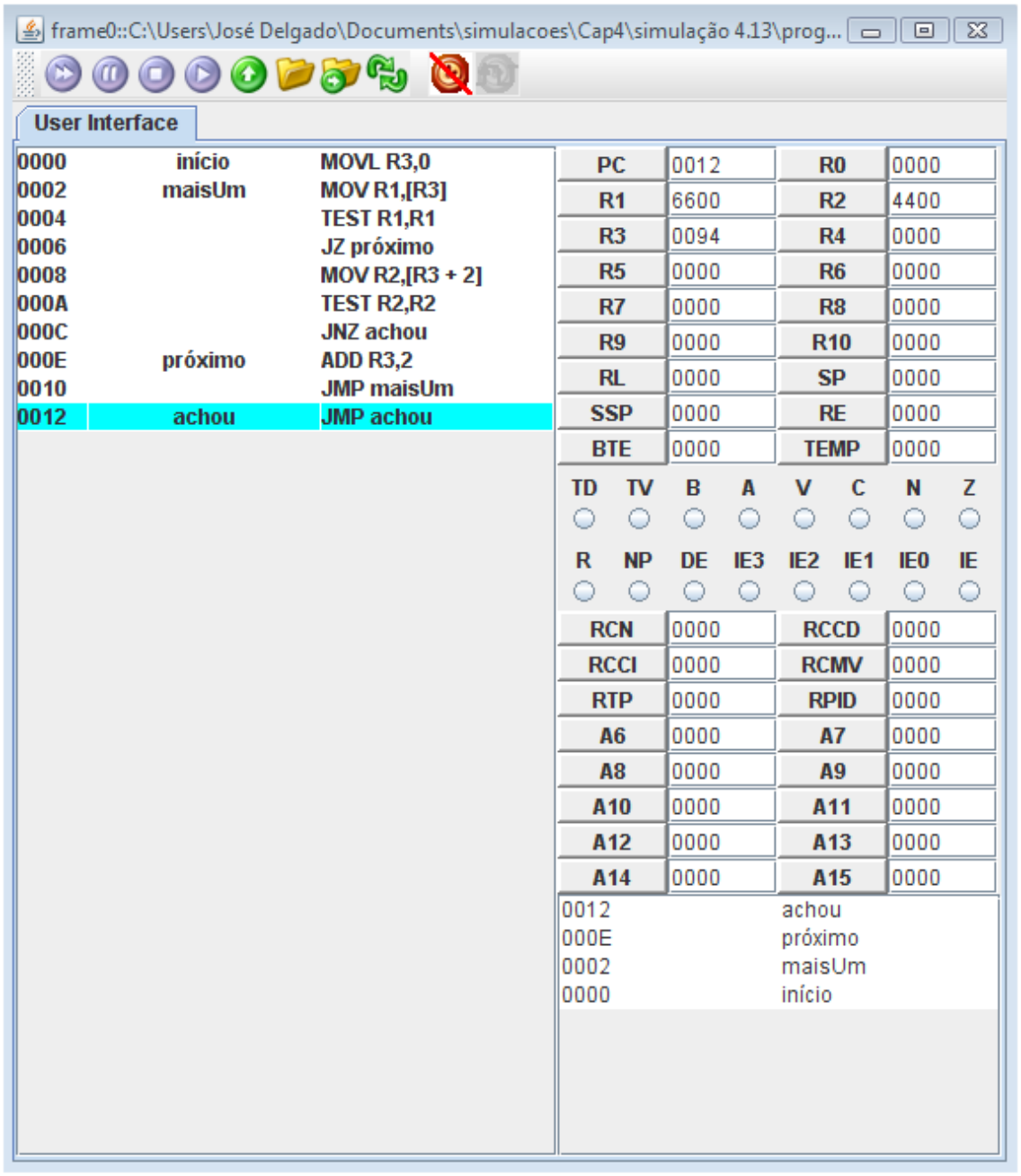
No painel da memória, altere duas palavras consecutivas (onde quiser, mas após o valor que colocu em R3) para serem diferentes de zero.

Para exemplificar, alteraram-se as palavras da memória nos endereços 94H e 96H, tal como se pode ver.



Addr...	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	C3	0	71	30	64	11	10	3	72	31	64	22	11	2	51	32
0010	2F	F8	2F	FF	0	0	0	0	0	0	0	0	0	0	0	0
0020	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0030	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0040	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0070	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0080	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0090	0	0	0	0	66	0	44	0	0	0	0	0	0	0	0	0
00A0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00B0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00C0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00E0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00F0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0130	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0140	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0160	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0170	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0180	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0190	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01A0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01B0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Executou-se o programa e o resultado foio seguinte:



Ou seja, em R3 está o endereço da primeira do par de palavras, e estas estão em R1 e R2, tal como esperado.