

# Guião de Laboratório de Arquitectura de Computadores

## Simulação 5.10 – Tabelas de uma só dimensão

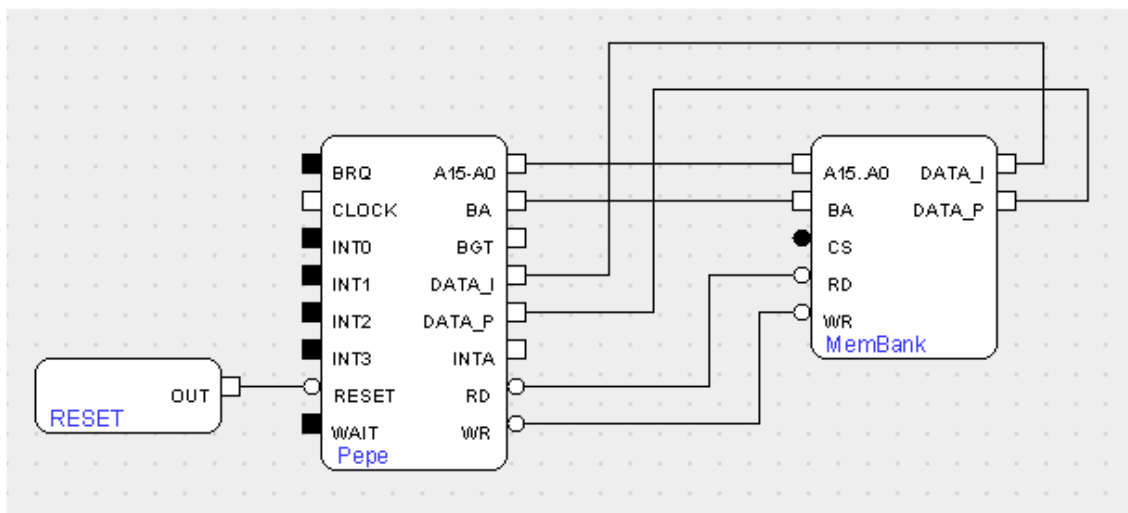
### 1 – Objectivos

Esta simulação ilustra o funcionamento das tabelas monodimensionais, tomando os programas 5.6, 5.7 e 5.8 como base. Os aspectos cobertos incluem os seguintes:

- Funcionamento das directivas TABLE, WORD e STRING;
- Disposição dos elementos das tabelas em memória;
- Cálculo dos endereços dos elementos de 8 e de 16 bits;
- Acesso à memória em 8 e 16 bits;
- Execução do programa passo a passo;
- Funcionamento dos programas e da evolução dos valores dos registos relevantes.

### 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 5.6

Carregue este circuito no simulador e passe para Simulação.

Abra o painel do PEPE e compile e carregue (📁) o ficheiro “programa5-6.asm”, que ilustra a procura de uma palavra específica numa tabela de 16 bits construída com directivas WORD.

[illegible]

Note que R1 tem sempre o endereço de base da tabela e não é alterado.

Verifique que, após a instrução, R0 fica com a palavra da tabela.

Verifique que o ciclo é iterado até se achar a palavra indicada por “chave”.

Faça reset no PEPE ou no simulador.


Esta directiva apenas reserva espaço, tantas palavras quantas as indicadas. A contrário das directivas WORD, é preciso inicializar a tabela com MOVs. Naturalmente, esta directiva é preferível apenas quando não se sabe o conteúdo da tabela em tempo de compilação.

Aproveite para notar a utilização de uma constante que somada à base da tabela dá o endereço real de cada elemento. Esta constante só pode variar entre -16 e +14, com valores pares.

O resto do programa é igual ao programa 5.6

## 5 – Simulação do programa 5.8

Faça reset no PEPE ou no simulador.

Abra o painel do PEPE e compile e carregue (  ) o ficheiro “programa5-8.asm”, que procura também um elemento numa tabela, mas agora no reino dos 8 bits.

A tabela é construída com a directiva `STRING`, que permite especificar constantes de um byte e strings, que são colocadas sequencialmente em memória.

Abra o painel da memória e confirme que as constantes especificadas em STRING já estão carregadas, em bytes consecutivos, a partir do endereço 1000H.

[illegible]

Execute o programa passo a passo.

Imediatamente antes da instrução `MOVB R0, [R1]`, anote os valores de R0 e R1 e verifique que, após a execução da instrução, o byte de menor peso de R0 contém o byte de memória cuja endereço está contido em R1, e que o byte de maior peso de R0 está a zero.

Ao contrário dos exemplos anteriores, que mantinham a base da tabela e somavam-lhe um valor para obter o endereço real do elemento, aqui o R1 contém directamente o endereço do elemento, e vai variando para varrer a tabela. Isto deve-se que a instrução **MOVB** não suporta endereçamento baseado nem indexado (apenas indirecto), por limitações do número de instruções codificáveis em 16 bits (ver Tabela 4.31).

Verifique que o program acha o valor pretendido (0ABH) na posição 5 da tabela (começando em 0). O valor final de R4 deve ser 5.