Guião de Laboratório
de
Arquitectura de Computadores (2ª edição)

Simulação 7.3 – Processamento em estágios

1 – Objectivos

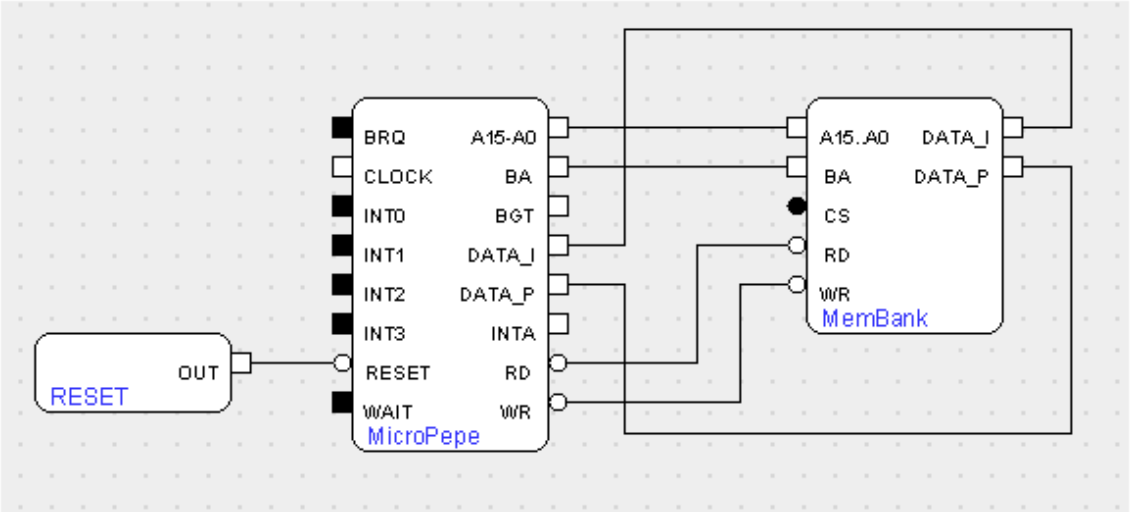
Esta simulação toma como base o conteúdo da secção 7.3 e exemplifica o funcionamento do processamento em estágios, utilizando o próprio PEPE no simulador, que permite:

- Visualizar a arquitectura em estágios (Fig. 7.12) e o valor das ligações;
- Executar as microinstruções passo a passo (várias simultaneamente, em regime de processamento em estágios), examinando o estado dos recursos de hardware após cada ciclo de relógio.

NOTA – Esta simulação só está descrita na 2ª edição do livro.

2 – Circuito

O ficheiro "microPEPE.cmod" implementa o circuito da figura seguinte.



Para utilizar o processamento em estágios do PEPE no simulador é necessário usar uma versão diferente do módulo PEPE, que simula o processador a nível funcional (instruções). A nova versão designa-se microPEPE e simula o processador de forma muito mais detalhada a nível de hardware, permitindo um completo controlo sobre a execução do processador. De resto, é totalmente compatível em especificações com o PEPE normal.

3 – Processamento em estágios no PEPE

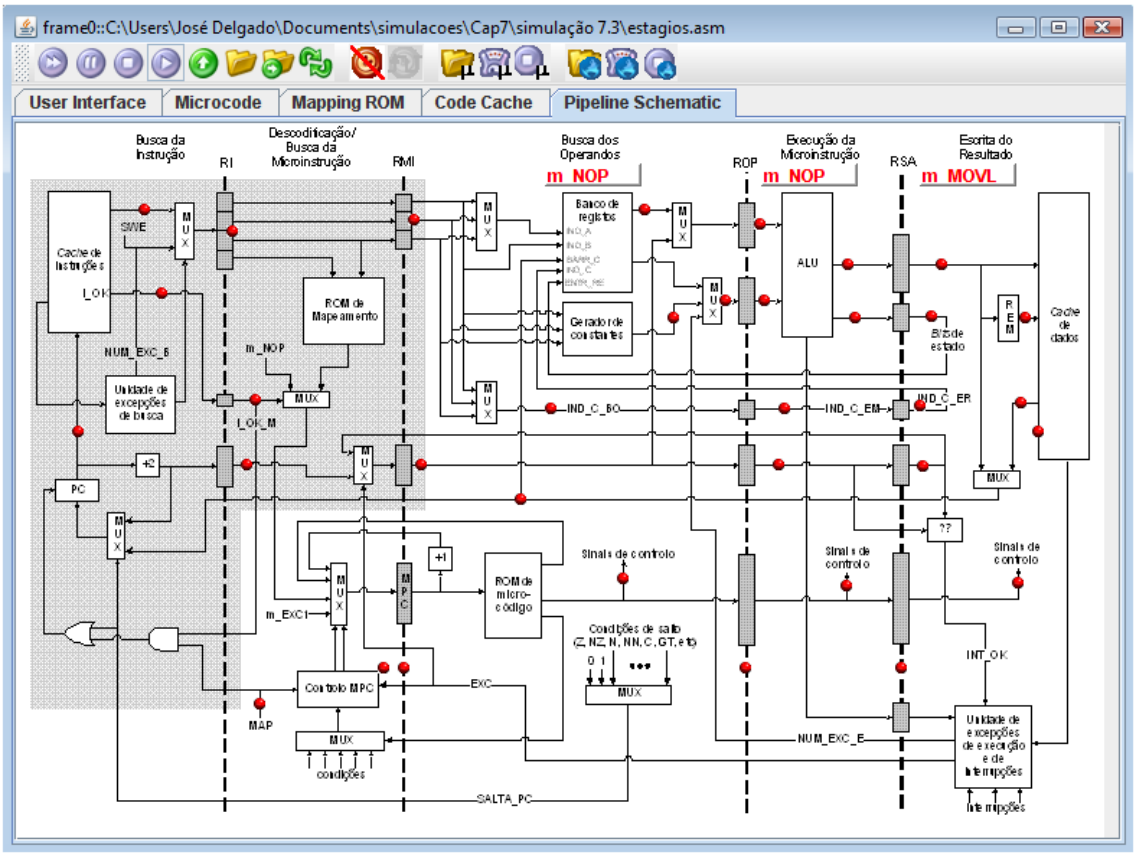
O processamento em estágios do microPEPE activa-se por meio de uma instrução, que liga um bit de controlo do processamento em estágios.

Este processamento obriga a algumas alterações no microcódigo face ao utilizado na simulação 7.2, nomeadamente:

- Quando uma instrução termina, não é o microcódigo que tem de ir fazer a busca de uma instrução. Mesmo antes de uma instrução terminar, já as seguintes têm de estar em processamento na cadeia de estágios. Assim, a busca é feita automaticamente pelo hardware, mesmo antes de saber se há um salto no microcódigo ou não, e mal uma instrução acaba começa imediatamente a execução da seguinte;
- A resolução de conflitos de dados e de controlo tem de ser tida em conta. Se necessário, tem de se trocar a ordem de execução das microinstruções ou mesmo inserir microinstruções que não fazem nada (tal como os NOPs em assembly).

Quando se liga o processamento com estágios, o microPEPE carrega automaticamente o microcódigo adequado.

Com o processamento em estágios activado, o esquemático muda automaticamente, passando a reflectir a Fig. 7.12 do livro e permitindo observar, em cada momento, o valor dos sinais do interior do processador. NOTA – na 1.ª edição do livro, esta figura tem duas gralhas, tal como indicado na errata no site de apoio (que contém a versão correcta da figura). Para comodidade, a figura corrgida está igualmente contida nos ficheiros desta simulação.



O programa no ficheiro "estagios.asm" permite testar o funcionamento do processamento em estágios. Este programa:

- activa a cache de código, que permite uma busca (automática, pelo hardware de suporte ao processamento em estágios) muito mais rápida do que se tivesse de fazer um acesso à memória principal (fundamental para permitir um funcionamento contínuo da cadeia de estágios, sem compassos de espera – atenção que à primeira vez que uma instrução é executada, o acesso à memória principal é inevitável. Só nas vezes seguintes que a instrução é acedida se nota a vantagem da cache);
- activa o processamento em estágios;
- executa um ciclo de instruções simples, mas que exibem dependências (indicadas nos comentários) que fazem com que este programa, tal como está, não funcione bem numa cadeia de estágios (*pipeline*).

Os registos RCCI e RCN são descritos no apêndice A do livro.

Carregue e compile o ficheiro "estagios.asm" na arquitectura "microPEPE.cmod";

- Execute o programa instrução a instrução, carregando no botão STEP () até percorrer todas as instruções e a barra azul estar na linha com a etiqueta "pipe:";
- Note que o R1 já mudou de 0 para 10H e a instrução já executou (ao contrário do que é normal sem cadeia de estágios, em que a barra azul indica a instrução que será executada). Isto deve-se ao facto de em cadeia de estados se usa escrita dos registos na primeira metade do relógio (Fig. 7.14 do livro), de forma a resolver parte do problema da dependência de dados;
- Vá executando instrução a instrução e vendo com cuidado os registos R1 e R2. Veja que os valores são correctos (R2 fica a 20H e R1 passa por 10H, depois 30H e finalmente 32H). NOTA – É importante que isto seja feito na primeira vez que o microPEPE executa estas instruções. Se necessário, faça reset ao simulador e recomece;
- Execute novamente o ciclo, instrução a instrução, e verifique que, no primeiro ADD, o registo R1 fica com 30H, como deveria ser, mas R1 fica com 10H, 30H e finalmente 12H, em vez de 32H como no exemplo anterior. Esta situação repete-se nas execuções seguintes do ciclo, se as fizer;
- Dadas as dependências declaradas no programa, em R1 e em R2, explique o seguinte:
    - Porque é que a dependência em R1 faz com que o resultado do segundo ADD seja 12H em vez de 32H, após a primeira execução?
    - Porque é que na primeira vez o R1 funciona bem e das seguintes a dependência já se faz sentir? Pista: a cache de instruções está inicialmente vazia e a busca de cada instrução demora mais do que um ciclo de relógio;
    - Porque é que no caso particular deste programa a dependência em R2 não causa nenhum problema (considere a primeira execução e as seguintes, pois as razões são diferentes nos dois casos);
- Active o relógio de um impulso de cada vez, carregando no botão  ;

- Passe para o esquemático e veja que estão três instruções em processamento na cadeia. Identifique-as no programa (tab "User Interface");
- O botão  permite executar o código micro-instrução a micro-instrução, olhando para o esquemático e vendo a evolução da cadeia de estágios. Para facilitar esta visualização existem três etiquetas, uma para cada um dos três estágios da cadeia de micro-instruções, que indicam qual a micro-instrução a ser executada em cada estágio;
- Carregando sucessivamente neste botão, verifique a evolução das microinstruções na cadeia. Note em particular o tratamento em avanço das instruções a seguir ao JMP, inútil dado que o JMP faz esvaziar a cadeia. Um estágio vazio aparece como m_NOP;

Corrija agora as dependências em R1 e R2 colocando NOPs no programa "estagios.asm" (crie uma cópia do ficheiro). Verifique que agora o R1 passa a ter o valor correcto de 32H.

Entretanto, os ficheiros "tabela7.11.pdf" e "tabela7.14.pdf" reproduzem as tabelas correspondentes do livro (a tabela 7.14 existe apenas na 2ª edição). Verifique as alterações efectuadas na tabela 7.14 em relação às instruções ADDM e SUM, devido ao processamento em estágios.

Pode experimentar o funcionamento da instrução SUM usando o ficheiro "SUM-estagios.asm", idêntico ao "nova-instrucao.asm" da simulação 7.2 mas activando o processamento em estágios. Efectue a execução com o botão STEP () até chegar à etiqueta "pipe:" e a partir daí desligue o relógio (), visualize o esquemático e execute as instruções com o botão  .

Note que, no microcódigo, a instrução SUM aparece implementada como na tabela 7.14 e não como na 7.11.

Note o desenrolar das microinstruções m_SUM1, m_SUM2, etc., e o ciclo dentro da instrução SUM.

O ficheiro "p610-2ªed.pdf" contém algumas explicações não disponíveis na 1ª edição do livro.