Note-se que um salto no microcódigo (como na microinstrução m_SUM7, Tabela 7.11) faz esvaziar apenas a cadeia de microinstruções, sem afectar a cadeia de instruções, uma vez que a instrução em execução se mantém.

Durante a execução das microinstruções de uma mesma instrução, a cadeia de instruções está parada e o valor do PC e do RMI mantêm-se (com excepção do MPC).

No entanto, no caso de um salto condicional no microcódigo, como em m_SUM2 da Tabela 7.11, a própria cadeia de microinstruções tem de parar até o resultado do teste ser conhecido, o que só acontece no estágio ER (Escrita do Resultado). Os *bits* que controlam as condições de salto do MPC (SA0, SAZ, etc.) ligam ao registo RSA, que memoriza a saída da ALU. Desde que se sabe que no MPC está uma microinstrução com um salto condicional até que esta chegue ao estágio ER decorrem dois ciclos de relógio, durante os quais não se pode deixar a cadeia de microinstruções (nem a de instruções) continuar. Caso contrário, se esta fosse uma das últimas microinstruções da instrução, poderiam entrar microinstruções de outras instruções, avançando o PC, RI e RMI e destruindo o contexto que se deve manter se o salto no microcódigo se der dentro de uma dada instrução (caso das instruções que contêm ciclos internos) ou se se tratar de um mapeamento condicional que salte para a instrução seguinte, como por exemplo a microinstrução m_SUM2 da Tabela 7.11, em que nesse caso o PC poderia estar já a apontar não para a instrução seguinte mas para uma posterior.

Nestas condições, o mais seguro é parar o MPC e a cadeia de estágios, quando o sinal SEL_MICRO_SALTO indica um salto condicional no microcódigo, durante dois ciclos de relógio, o que é feito pelo bloco "Controlo das cadeias" da Fig. 7.12, através do controlo do sinal de relógio durante dois ciclos de relógio do processador. Este bloco é uma máquina de estados simples, que gera ainda (no segundo e terceiro ciclos de relógio) um sinal que faz entrar zeros na parte de controlo do ROP a seguir à passagem da microinstrução de salto condicional (que serão depois propagados para o RSA). Isto garante que todos os sinais ficam inactivos até a próxima microinstrução fluir novamente, o que equivale, na prática, a um esvaziamento controlado da cadeia de microinstruções.

Note-se que os saltos incondicionais no microcódigo (incluindo o mapeamento da próxima instrução) não usam este esquema, pois o salto é feito imediatamente no MPC e fluxo de microinstruções é contínuo, sem necessidade de esperar por uma decisão.

O sinal D_OK, que indica se o acesso à *cache* de dados teve sucesso, pode também impedir o avanço das cadeias de estágio, tal como é indicado na Fig. 7.12. No entanto, neste caso também o sinal de relógio dos restantes estágios é afectado (todo o processador pára à espera que a *cache* complete o seu acesso à memória principal). Este detalhe está omitido na figura, por simplicidade.

7.3.4 EXCEPÇÕES COM PROCESSAMENTO EM ESTÁGIOS

Para além dos saltos no PC e no MPC, as excepções constituem eventos que interferem com a regularidade do funcionamento em linha das cadeias de estágios e que requerem muitos