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

Simulação 7.9 – Protecção

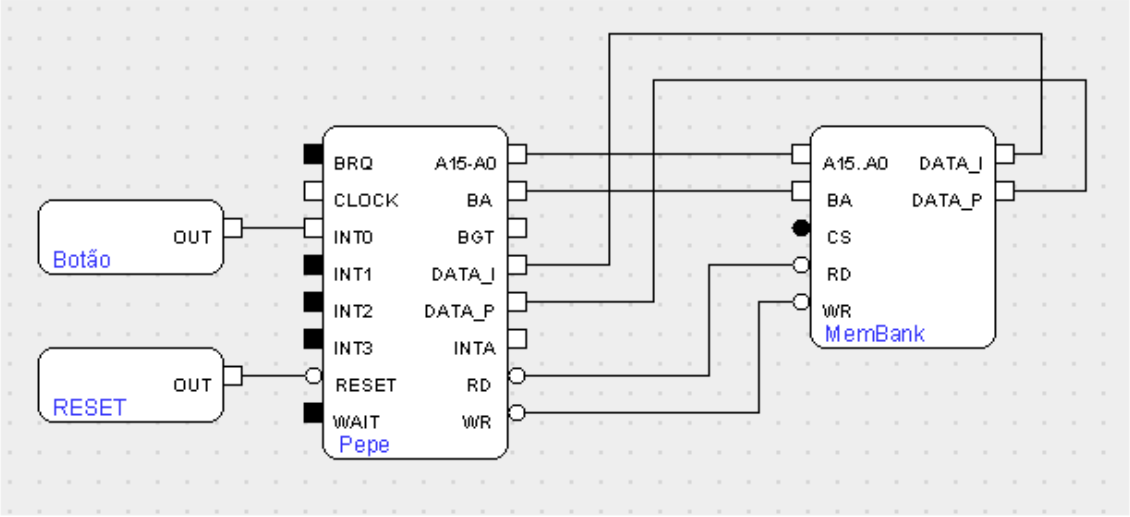
1 – Objectivos

Esta simulação exemplifica as técnicas de protecção descritas nesta secção, com pequenos exemplos que exercitam os vários aspectos referidos, nomeadamente:

- Níveis de privilégio e utilização do bit NP;
- Verificação da comutação do SP com o valor de NP;
- Geração de uma excepção SISTEMA com execução de acções não permitidas ao nível de Utilizador;
- Implementação (com microprogramação) de uma instrução de nível de Sistema;
- Verificação da protecção ao nível da memória virtual com os descritores de páginas de memória virtual.

2 – Circuito

O ficheiro "pepe.cmod" implementa o circuito básico do PEPE, apenas com o banco de memória e um botão para exercitar a interrupção 0.



3 – Simulação

Carregue este circuito no simulador e passe para Simulação. Abra o painel do PEPE e compile e carregue ( ) o ficheiro "proteccao.asm", que contém um pequeno programa para testar as características de protecção do PEPE. Execute-o passo a passo (botão).

O programa começa em modo de Sistema (NP=0), que é o modo a seguir a uma inicialização do PEPE. Verifique que a chamada à rotina X, e as suas instruções PUSH e POP, usam a pilha do Sistema e o SSP.

A instrução SET RE, 14 coloca a 1 o bit 14 do RE, ou seja, o NP. Isto faz o PEPE passar para modo utilizador.

As instruções de MOV SP, 200H e o CALL X, bem como as suas instruções PUSH e POP, usam a pilha do Utilizador e o USP.

Até aqui, tudo normal. O problema põe-se ao tentar voltar para modo Sistema, tentando colocar o bit NP a 0, usando a instrução CLR RE, 14. Não funciona. O PEPE não o permite, senão não havia protecção (qualquer programa de modo Utilizador poderia facilmente passar para modo Sistema e ter acesso a todos os recursos do computador).

Qualquer tentativa de execução de uma acção ilegal em modo Utilizador gera uma excepção SISTEMA (ver tabela A.8). A secção 7.7.5 indica os casos em que esta excepção pode ser gerada.

Verifique que, ao tentar executar a instrução CLR, o controlo vai parar à instrução com a etiqueta erro_sistema, que ignora o erro (retorna, simplesmente). Note que o NP está a zero imediatamente antes do RFE (o que acontece em qualquer geração de excepção), mas após o RFE o NP continua a 1.

A instrução seguinte, SWE, ilustra precisamente este mecanismo de passagem de modo Utilizador para modo Sistema ao atender uma excepção. O SWE em modo Utilizador é normalmente usado para chamadas ao sistema, e é isso que o programa simula. Execute a instrução e note que o controlo vai parar à rotina "util0", já com o NP=0 (modo Sistema). Todo o atendimento e utilização da pilha (e SSP) é feito em modo Sistema, como pode comprovar ao executar o PUSH e POP desta rotina.

Mas note que após a execução do RFE o NP voltou a estar a 1, o que significa que o programa voltou ao modo Utilizador, tal como seria de esperar de uma chamada ao sistema.

As instruções seguintes ilustram o mesmo mecanismo, mas agora com uma interrupção, que pode ocorrer durante a execução de um programa em modo Utilizador.

Coloque um ponto de paragem na instrução com a etiqueta "rot_int0" (clique na instrução, devendo aparecer uma barra roxa). Execute o programa em modo contínuo (botão ). O programa estará a executar a última instrução (JMP fim) indefinidamente.

Tome note do valor de USP e de SSP.

Abra o painel do botão e carregue. O programa pára (barra azul) na instrução com a PUSH R2, a primeira da rotina de atendimento da interrupção 0. Note que NP já está a 0 e que o valor de SSP diminuiu quatro unidades. Isto quer dizer que a rotina de interrupção está a ser atendida em modo de Sistema e que o PC e RE do programa interrompido foi colocado em segurança na pilha de sistema.

Execute agora o programa em passo a passo (botão ). O RFE faz o processador voltar ao programa (instrução JMP fim), de novo em modo Utilizador. Confira o valor do NP, que voltou a 1. Conclusão: as interrupções a um programa executado em modo de Utilizador são seguras.