Guião de Laboratório
de
Arquitectura de Computadores

Simulação 4.1 – Bits de estado

1 – Objectivos

Esta simulação ilustra o funcionamento dos bits de estado com a instrução ADD. Os aspectos cobertos incluem os seguintes:

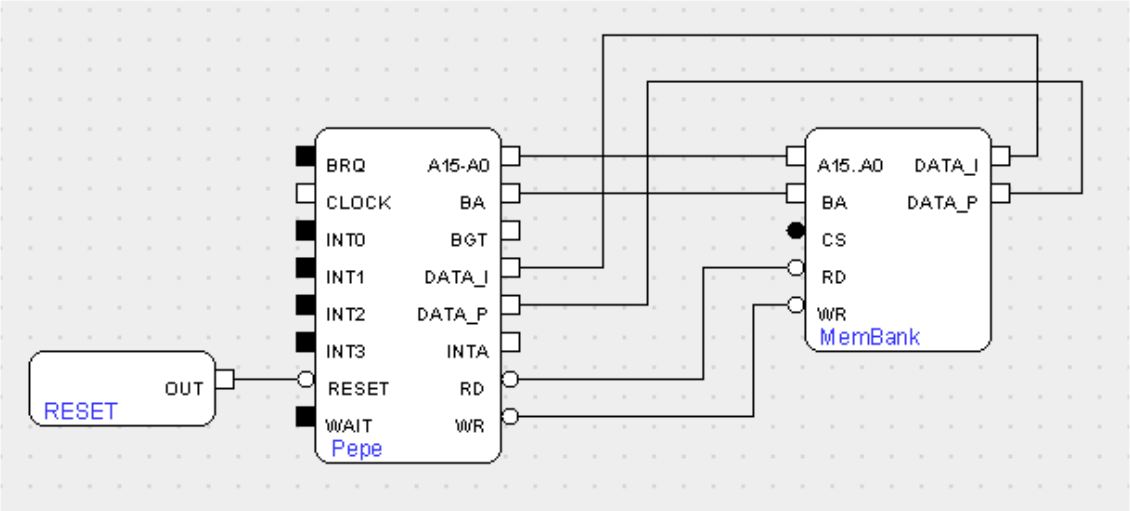
- Carregamento manual dos registos com os operandos, usando a Tabela 4.8 com base, mas agora com operandos de 16 bits;
- Verificação do funcionamento da instrução ADD e do valor dos bits de estado após a instrução;
- Repetição para vários valores de operandos.

2 – Circuito

O ficheiro "pepe.cmod" implementa o circuito da Fig. 4.7, em que o PEPE (16 bits) está ligado a um módulo de Reset, que faz a sua inicialização quando a simulação começa.

O PEPE possui um relógio interno, pelo que a sua entrada de Clock pode ficar no ar.

O banco de memória (MemBank) é constituído por duas memórias de 8 bits lado a lado, para fazerem 16 bits. DATA_I e DATA_P são os buses de dados de 8 bits cada, que ligam a cada uma dessas memórias. Uma contém as células (de um byte) com endereços pares e outra com os endereços ímpares. A memória tem o seu chip select (CS) sempre activo, uma ves que não há mais dispositivos no sistema, e os sinais WR e RD (só um deles pode estar activo a 0 de cada vez) indicam se o processador está a ler da memória ou a escrever.

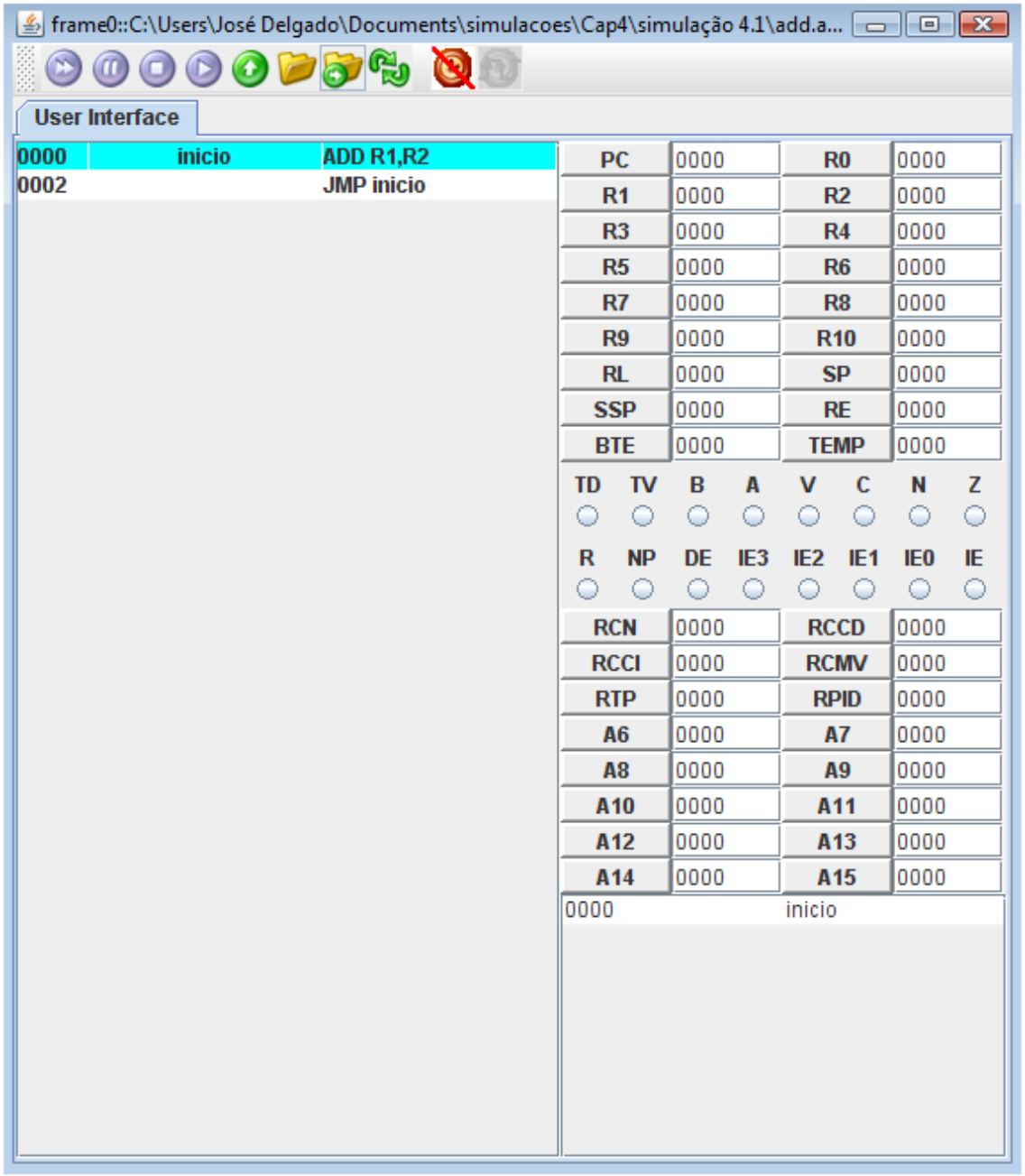


Em todas as simulações do Capítulo 4, o circuito de hardware não é relevante, pois o que está em causa é a funcionalidade das instruções do PEPE.

3 – Utilização básica do circuito

Para executar este circuito deve efectuar os seguintes passos:

1. Use o comando File->Load do simulador para carregar a arquitectura "pepe.cmod"
2. Passe para modo Simulação
3. Efectue um duplo clique sobre o PEPE para abrir o seu painel de controlo. Carregue no botão Compile&Load ( ) e escolha o ficheiro ".asm" correspondente ao programa que for executar. Possível exemplo:



4. Carregue no botão de execução passo a passo ( ) se quiser executar uma instrução de cada vez ou no botão de execução contínua ( ) na janela do "Pepe" de modo a executar o programa passo a passo.
5. Em execução passo a passo, pode ver o resultado da execução olhando para os registos do processador e para a memória (o painel de controlo desta abre-se com um duplo clique).

4 – Simulação

Compile e carregue ( ) o ficheiro “add.asm”. O aspecto do painel de controlo do PEPE deverá ficar como ilustrado em cima.

Note que todos os registos e bits de estado estão inicializados a 0. Não é isto que sucede nos processadores reais, em que os valores ficam aleatórios. No entanto, para simplificar, o simulador inicializa automaticamente todos os recursos internos do PEPE.

Note que o PC vale 0, dado que a primeira instrução que o PEPE começa a executar após o reset (inicialização) é sempre no endereço 0. Nesta altura, esta instrução, um ADD (soma) ainda não foi executada.

Carregue no botão de execução de passo a passo ( ) para executar esta instrução e parar em seguida.

Note que R1 continua a 0 (tanto R1 como R2 estavam a 0, logo a sua soma dá 0), mas nos bits de estado Z passou a estar activo, indicando que a última operação da ALU deu resultado 0.

TD	TV	B	A	V	C	N	Z
○	○	○	○	○	○	○	◉
R	**NP**	**DE**	**IE3**	**IE2**	**IE1**	**IE0**	**IE**
○	○	○	○	○	○	○	○

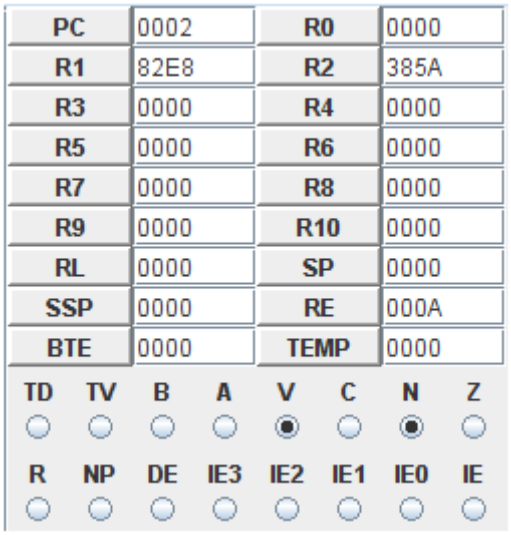
Note ainda que o PC passou para 2, onde se localiza a próxima instrução (JMP), cujo efeito é indicar ao PEPE para saltar (jump) para a instrução “inicio”. Carregue de novo em ( ) e verifique que o PC passou para 0. A barra azul vai indicando a próxima instrução a ser executada (vai executar a soma novamente)

Altere manualmente os valores dos registos R1 e R2, para a instrução ADD ter interesse. Coloque por exemplo R1=1234H e R2=385AH. IMPORTANTE: a inserção de cada número tem de terminar com Enter.

PC	0000	**R0**	0000
R1	1234	**R2**	385A
R3	0000	**R4**	0000
R5	0000	**R6**	0000
R7	0000	**R8**	0000
R9	0000	**R10**	0000
RL	0000	**SP**	0000
SSP	0000	**RE**	0001
BTE	0000	**TEMP**	0000

Carregue de novo em ( ) e verifique que R1 foi actualizado com o resultado (4A8EH) e que o bit de estado Z ficou inactivo, pois o resultado da operação já não é zero. R2 não foi alterado.

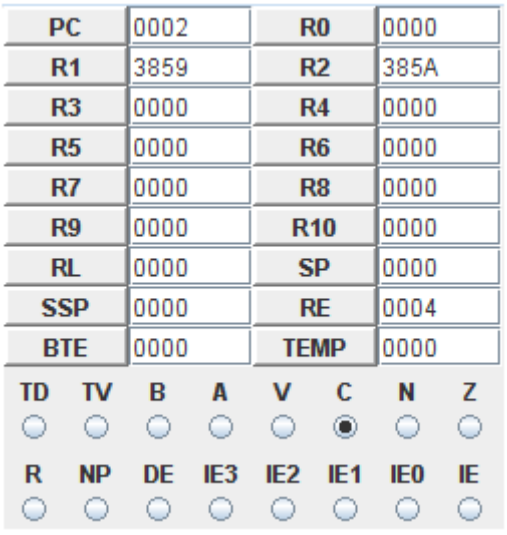
Execute de novo o JMP e o ADD, o que faz somar 4A8EH (R1) com 385AH (R2), ficando R1 com o valor 82E8H. Se fizer as contas no papel em hexadecimal, verifica que é isso que dá, mas note também que os bits N (negativo) e V (overflow, ou excesso) estão activos. Porquê?



Os números estão representados em complemento para 2 com 16 bits. Desta forma, o maior número positivo que conseguem representar é 7FFFH. O resultado excedeu assim o limite e naturalmente está incorrecto. Nesta representação, é um número negativo (bit de maior peso a 1), o que nunca poderia resultar da soma de dois números positivos.

Coloque agora manualmente o valor FFFFH (-1) no registo R1 (não se esqueça de fazer Enter) e volte a executar a instrução ADD.

Note que o resultado em R1 passou a ser 3859H, que é o valor positivo que se podia esperar da expressão (– 1) + 385AH. Note também que o bit de estado C (Carry, ou transporte) ficou a 1. Isto corresponde a uma situação normal, idêntica à do caso 4 na tabela 4.9.



Se desejar, experimente com outros números.