Relatório do Projeto Final de Laboratório de Circuitos Digitais - Fase 2 Fábio Miguel Denda Pacheco

Grupo 14: Fábio Miguel e Luís Gustavo

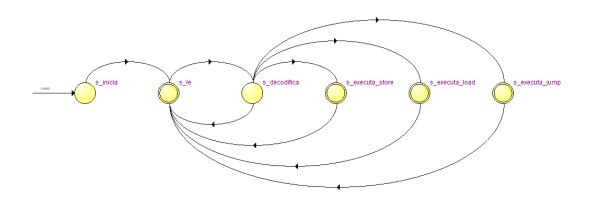


Diagrama de estados:

- O diagrama mantém a estrutura apresentada na Fase 1, mas agora inclui o estado s_executa_store para implementar a instrução STORE. A sequência é:
- s_inicia ponto de partida após o reset; nenhum registrador muda, apenas alinha a MEF.
- s_le coloca o PC no barramento, lê a memória e prepara a carga do IR e o incremento do PC. s_decodifica examina o opcode: se for 01h (STORE) vai para s_executa_store; se for 02h (LOAD) vai para s_executa_load; se for 03h (JUMP) vai para s_executa_jump (JUMP foi feito pelo Luís, peguei o projeto depois que ele tinha implementado o JUMP); caso contrário retorna a s le.
- s_executa_store coloca IR(7..0) no barramento de endereço, apresenta o conteúdo do AC no barramento de dados, habilita MW = '1' por um ciclo e grava o dado na RAM, depois volta a s_le.
- s_executa_load e s_executa_jump permanecem iguais aos da fase anterior.

Com isso, a FSM completa o ciclo fetch \rightarrow decode \rightarrow execute e agora também grava dados na memória conforme solicitado na especificação da Fase 2.

Edit:/nanoproc/reset 0	0													
Edit:/nanoproc/dock														
	0000	0000	1234											
	0000	0220			0130						0221			
<u>+</u> → sim:/nanoproc/PC 0	00	01			02						03			
im:/nanoproc/addr im:/nanoproc/addr	00	20	01		02		30		02		03		21	
<u>∓</u> → sim:/nanoproc/data 0	0220	1234		0130		0221		1234		0221		0131		ABCD
sim:/nanoproc/esta s	s_le	s executa	s le		s decodifica		s executa st	ore	s le		s decodifica		s executa lo	d
sim:/nanoproc/proxi s	s_decodifica	s le	s decodifica		s executa st	ore	s le		s decodifica		s executa lo	d	s le	

Simulação:

Linha 1 (AC): $0000 \rightarrow 1234$ (permanece) \rightarrow ABCD \rightarrow 5678...

Linha 2 (IR): 0000, 0220, 0130, 0221, 0131, 0222, 0132...

Linha 3 (PC): 00, 01, 02, 03, 04, 05...

Linha 4 (estado_atual):

s_inicia, s_le, s_decodifica, s_executa_load, s_le, s_decodifica, s_executa_store, s_le...

Linha 5 (addrBus): segue o PC nos fetchs; assume 20 no LOAD, 30 no STORE, 21/31 no par seguinte, 22/32 no terceiro...

Linha 6 (MW): permanece '0' em todos os ciclos exceto um pulso alto em cada s_executa_store.

As transições ocorrem exatamente como definidas no VHDL:

PC = 00, s_le - lê 0220 (LOAD 20), grava no IR, incrementa PC para 01.

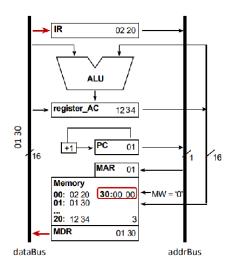
s_decodifica identifica opcode 02 e passa a s_executa_load.

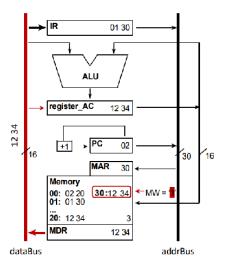
s executa load – lê 1234 do endereço 20 e carrega no AC; retorna a s le.

PC = 01, s le – lê 0130 (STORE 30), grava no IR, incrementa PC para 02.

s_decodifica detecta opcode 01 e vai a s_executa_store.

s_executa_store – coloca endereço 30 no barramento, apresenta 1234 no dataBus, ativa MW = '1' por um ciclo e grava 1234 em 30; volta a s le.





Fluxogramas da instrução STORE:

A Figura X (à esquerda) mostra o ciclo de busca do comando STO 30 (0130).

O barramento de endereço (preto) carrega PC = 01; a RAM devolve a palavra 0130, que percorre o dataBus até o MDR e é colocada no IR (setas vermelhas). O acumulador já contém 1234, resultado do LOAD anterior, enquanto MW permanece '0' e a posição 30 da memória ainda guarda 0000.

A Figura Y (à direita) retrata o ciclo de execução em s executa store.

Agora o dataBus inteiro é destacado em vermelho, pois transporta o dado 1234 que sai do AC, passa pelo MDR e entra na RAM. Simultaneamente, o addrBus exibe o operando 30 (obtido de IR(7..0)), e o controle MW = 1 fica alto por um único ciclo, habilitando a escrita. Como resultado, o conteúdo da posição 30 muda de 0000 para 1234, completando a operação STORE antes de a FSM voltar para o estado de busca.