Projeto Nº 16

**Fase 1:**

Na fase 1 criou se todos os módulos presentes na DataPath, os módulos IMemory e DMemory foram iniciados com os dados fornecidos no projeto.

**IMemory:**

A IMemory esta pode guardar no máximo 16 instruções, contudo neste projeto apenas vamos definir 7 instruções.

As instruções são as seguintes:

**LW $0, $1, 0**

**LW $0, $2, 1**

**LW $0, $3, 2**

**XOR $1, $2, $4**

**ADDI $4, $5, 1**

**ADD $5, $3, $1**

**SW $0, $1, 0**

Existem 2 tipos de instruções, a de Tipo 1 e de Tipo 2, podemos ver o formato de cada uma no ponto 2 do projeto.

As operações do tipo 1 são **ADD**, **SUB** e **XOR** e o seu opcode é 001, a instrução

**ADD $5, $3, $1** , é de tipo 1, para converter para código de maquina devemos seguir o formato assembly mencionado no ponto 2 do projeto que para instruções do tipo 1 é o seguinte

**func, $rs,$rt $rd**

O formato em código de máquina é o seguinte:

**opcode** (3bits), **rs**(3bits),**rt**(3bits), **func**(3bits)

Sabemos que esta intrução é do tipo1, logo o opcode é 001

Sabemos que o **rs**, **rt** é **rd** são 5,3 e 1 respetivamente o que em binário dá 101,011 e 001.

Sabemos analisando a tabela 2 que a função **ADD** em binário é representada por 0000

Agora que já sabemos toda a informação basta formata-la de acordo com o formato binário pedido.

**OP.RS.RT.RD.FUNC**

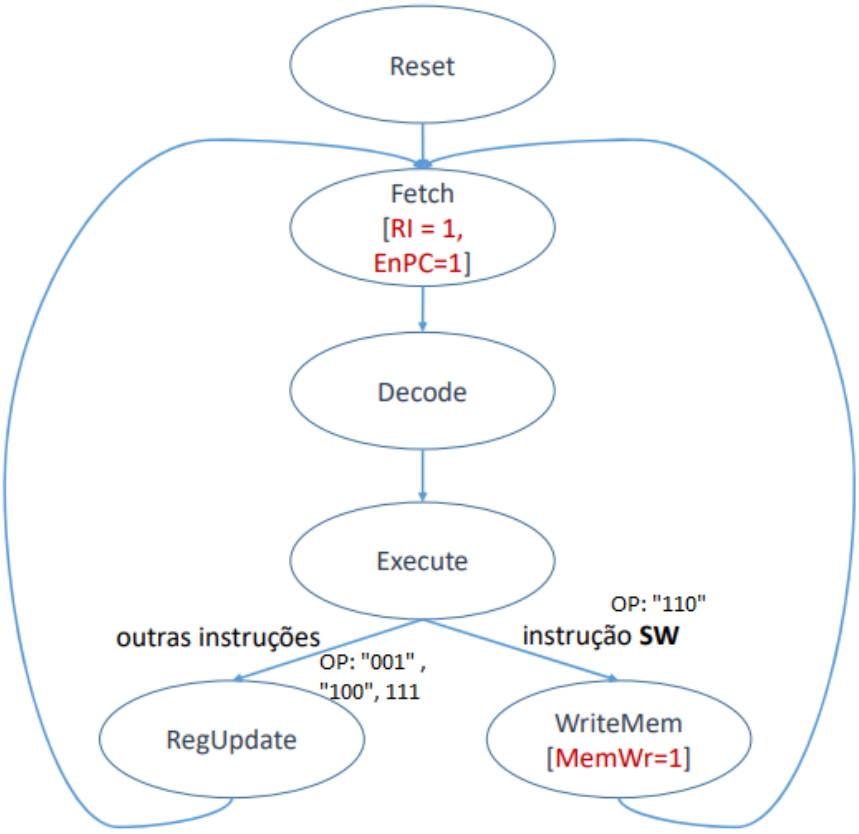
**ADD $5, $3, $1 =>** 001.101.011.001.0000 (pontos para uma melhor leitura)

**Validação:**

Para validação foram feitas várias Test Benches, para entidades triviais como é o caso dos multiplexers não foi criado nenhum test bench.

A TestBench foi então compilada na ferramenta ModelSim do Quartus onde se analisou a wave form formada por cada bloco lógico.

**Fase 2:**

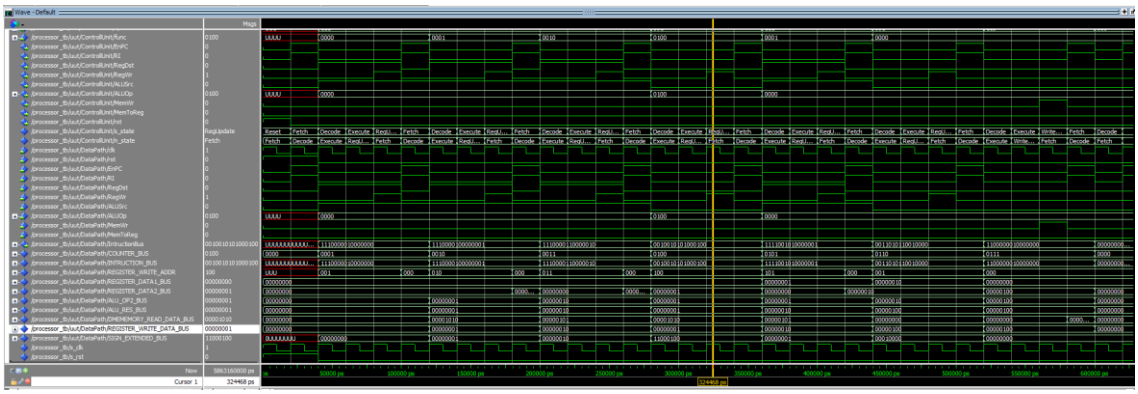



---

A control unit também foi validada com uma WaveForm, após verificação da wave concluiu se que estava a funcionar corretamente.

**Fase 3:**

Na fase3 foi feita a interligação do control unit com o data path, após validação verificou se que esta também esta a funcionar corretamente.



Na foto acima podemos ver (a foto é pequena mas dá para ampliar e ver os valores, inclui também a foto em tamanho original no projeto) a operação XOR selecionada.

A instrução é a seguinte **001**.**001**.**010**.**100**.**0100** (separado por pontos para uma leitura mais fácil) em formato assembly traduz se para **XOR $1**, **$2**, **$4**, esta instrução é uma operação que escreve no registo4 o resultado de uma operação **XOR** entre o registo 1 e 2.

Ao analisar a wave form conseguimos confirmar que o opcode é do tipo 1 “001” (não visível na imagem), e que a func é 0100 que corresponde a função **XOR** na **ALU**.

O RD1 e RD2 da Register é 0 e 1 respetivamente (valores reais representados em 8 bits) ao aplicar uma operação **XOR** o resultado esperado seria 1, o que se verifica, analisando o **ALU_RES_BUS** podemos ver que este tem o valor de **00000001**, assim podemos concluir que a operação foi feita com sucesso.

As outras operações foram verificadas usando os métodos equivalentes aos usados acima, e conclui-o se que as outras operações também foram executadas com sucesso.

**Autoavaliação: 14**

Consegui fazer as 3 partes com exceção da quarta parte, pois não tive muito tempo para a fazer devido a outros projetos/exames.

As simulações que fiz parecem me estar todas corretas, senti uma pequena dificuldade ao completar o diagrama de estados, mas consegui entender bem o fluxo do projeto.

Autor: Tiago Oliveira

Nº Mec: 99064