MC542: Projeto MIPS

Prof. Paulo Cesar Centoducatte
PED: Raphael Zinsly

Bruno de Campos Buccolo (RA: 075984)
Douglas Nunes Alves (RA: 081181)
Richard Keller (RA: 082704)

Introdução

O processador MIPS Pipelined sem dúvida foi uma das partes mais interessantes da matéria. Primeiro por ter um paradigma totalmente diferente em relação aos outros processadores vistos. Seguramente admitimos que foi difícil entender seu funcionamento, mas agora podemos aprecisar sua complexidade.

Enfim, podemos dizer que sabemos como o processador com pipeline funciona. Vimos também a importância de detecção de hazards, forwarding e das modificações para reduzir a penalidade por errar o branch. Não fosse por isso, nós programadores, gastaríamos um bom tempo tendo que preencher nosso código com instruções que desperdiçam ciclos.

VHDL não é a área de expertise de nenhum dos integrantes do grupo, por isso tivemos algumas dificuldade ao implementar a especificação do processador. Estas que descrevemos na seção a seguir

Desenvolvimento

Para desenvolvermos a arquitetura do projeto, nos baseamos na seguinte imagem do livro de referência, desconsiderando o Hazard Unit e outras ligações necessárias para forwarding, além de adicionarmos os sinais de controle Jump e Jal, fazendo as alterações necessárias.

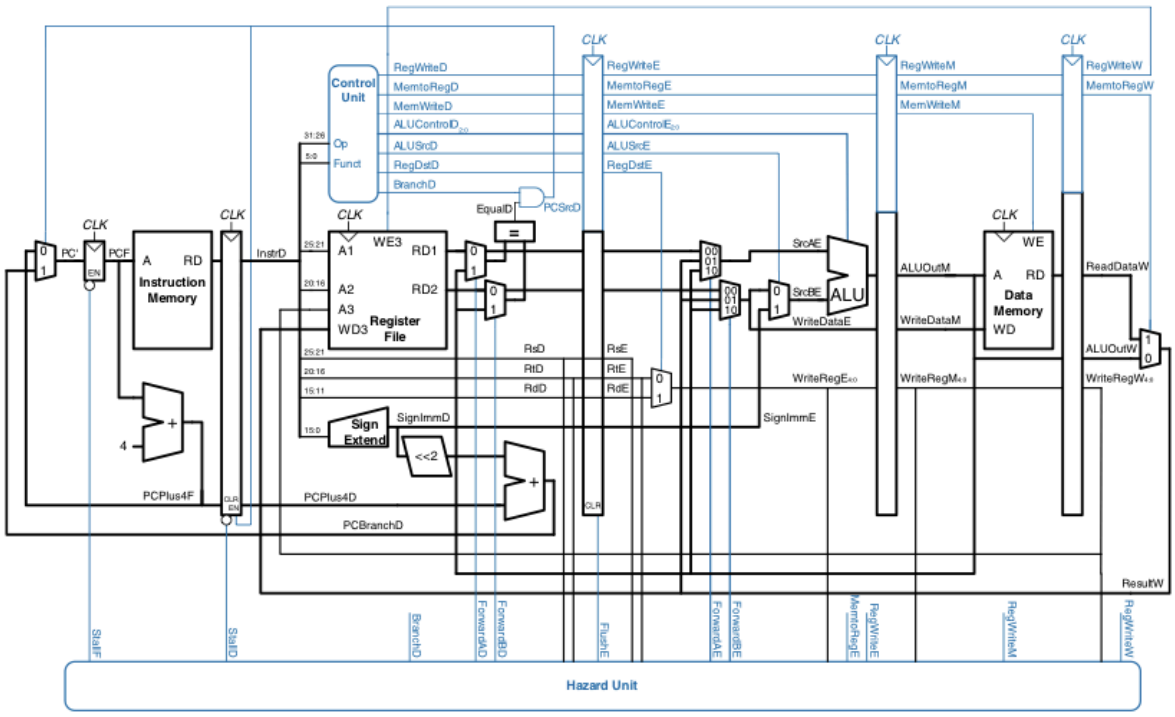


Figure 7.58 Pipelined processor with full hazard handling

Desenvolvemos cada etapa do pipeline em um módulo distinto. Primeiro temos o Fetch, que engloba o instruction fetch e a alteração do PC. Seguido dos módulos de Decode, Execution, Memory e Write Back.

Deixamos o Control Unit junto ao módulo de Decode, os sinais de controles são passados para as outras etapas do mesmo modo como representado na imagem acima. O Control Unit contém a lógica de verificar o Op e Funct e enviar os sinais de controles corretos.

Inserimos o Data como especificado no enunciado, o mesmo é síncrono, ou seja, o sinal é passado de etapa a etapa pela subida do clock, para que seja lido apenas na fase de memory.

A ULA foi extendida do lab anterior, aceitando também a operação de XOR. A operação de subi não implementamos pois não há motivo para existir visto que é o mesmo que addi com um registrador de valor oposto.

No mips.vhd fizemos a integração de todos os componentes, onde os sinais de saída de um componente são os sinais de entrada do componente seguinte. Essa transferência ocorre sempre na subida do clock, o que faz o pipeline funcionar.

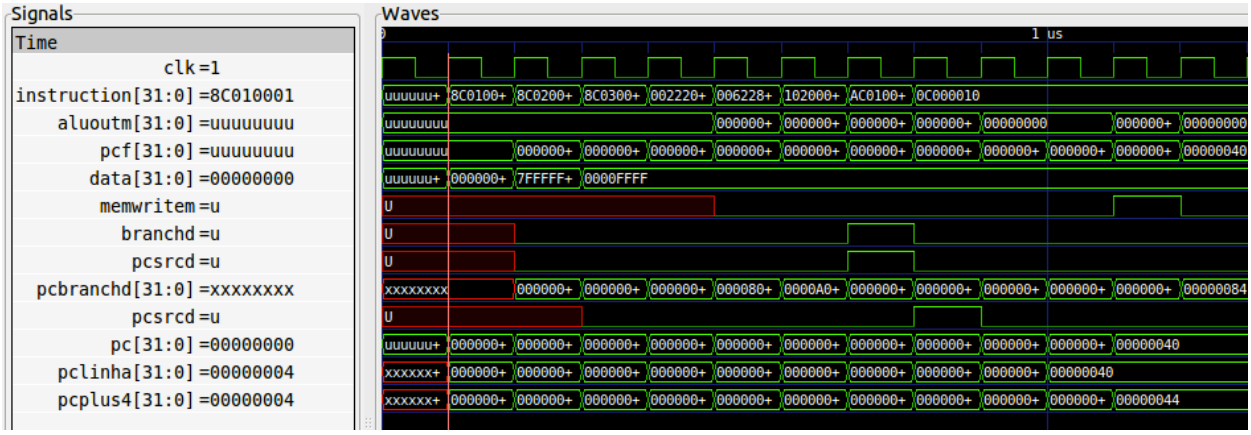
Validação

Criamos test bench parciais para cada componente a princípio para garantirmos a funcionalidade de cada módulo antes de integrarmos, essa prática foi muito útil para debug.

Testar a ALU de antemão nos trouxe segurança para testar apenas algumas instruções que a utilizam, uma vez que sabemos que o comportamento do processador é o mesmo para esse conjunto de instruções R-Type.

Ao testar, fica claro o quanto não ter uma unidade de detecção de hazards e que faça forwarding faz falta. Mesmo ao escrever o testbench, foi necessário considerar o espaço de tempo que temos que esperar até que a instrução a ser verificada chegue no estágio desejado.

O testbench pode ser encontrado no diretório indicado pela especificação do projeto. Abaixo temos uma pequena amostra dos sinais e do funcionamento do processador conforme o esperado:



Percebemos também o valor de uma ferramenta como o Gtkwave em que conseguimos observar os estados de todos os sinais de todos os componentes.