Para entender o funcionamento do sistema de diagramas é importante conhecer os principais aspectos do processador. Por isso primeiro vou descrever um breve resumo das características

1. Resumo sobre o funcionamento do processador

O processador foi feito tendo como base a arquitetura MIPS monociclo, ou seja, a cada ciclo somente uma instrução é realizada. Apesar disso, existem algumas diferenças, como o formato de instrução ser diferente; instruções de desvios condicionais possuírem um passo intermediário; a existência de instruções de comparação sem tomada de desvios; etc. Além disso, as instruções foram agrupadas em 5 grupos: aritmética, bitwise (bit a bit), comparação, movimentação, desvio/detour/deviation.

Sobre o módulo de entrada e saída, ele foi feito sem ter como referência outro modelo. Para o seu funcionamento, o sistema possui dois clocks que são síncronos: o *basis_clock*, que é um clock mais rápido e ele determina a velocidade do módulo de entrada e saída; e o *cpu_clock* que é o clock mais lento e determina o clock da cpu. Para ser mais preciso, a cada 2 *clocks* do basis_clk, o cpu_clk realiza 1 clock. Essa determinação ocorre para que haja tempo dos dados de entrada e saída serem armazenados na cpu ou mostrados ao usuário sem delay.

Uma última informação interessante é de que a cpu interage com os dados da seguinte forma: durante a subida do clock ocorre a leitura dos dados e a realização do circuito combinacional; e durante a descida do clock ocorre a escrita dos dados. Essa definição só não se aplica ao módulo de entrada e saída que opera sempre na descida do seu *clock*.

...

Sobre o sistema

2. Requisitos

Os requisitos foram pensados com base em todas as suas funcionalidades. Essas funcionalidades podem ser ações de interação com o usuário externo (Input e output), as quais possuem limitações do tamanho de dados de input e quantidade de display para o output; as tarefas (dos 5 grupos de instruções já mencionados) que a cpu pode realizar internamente; e as definições de como o ciclo de tempo deve operar (Sincronia; quantidade de clocks; performance).

3. Casos de uso

Os diagramas de caso de uso foram pensados de forma a cobrir dois níveis de especificações. Um nível de abstração tendo como base um ator usuário. Nesse caso, o diagrama é de modo superficial, mostrando em alto nível os acontecimentos. O outro nível envolve considerar alguns componentes físicos como atores, e dessa forma, o diagrama mostra as ações que esses componentes podem realizar independentemente. Tendo isso como base, os diagramas de caso de uso desenvolvidos foram: <Interact>, <Data Flow>, <IO Module>, <Modules>, onde somente o primeiro representa alto nível de abstração e contém os diagramas de atividade.

3.1 Interact

Foi realizado pensando no uso que um usuário faria do sistema. Esse sistema seria a combinação de uma arquitetura de CPU instanciada em uma placa eletrônica. Com isso, um usuário poderia ser alguém interagindo com um programa já carregado na CPU. Essas interações podem ser: congelar o processamento através de um botão; enviar dados de input através dos triggers da placa para definir o valor binário e apertar um botão para o envio efetivo; e ligar/desligar a CPU. Neste último, o processamento da CPU inicia automaticamente, o que envolve a realização de suas atividades internas. Essas atividades podem ser manipulação de dados; realização de desvios e entrada/saída de dados.

Além disso, também é possível que um ator “engenheiro” interaja com o sistema ao escrever o código em linguagem de máquina e carregar esse código em memória.

3.2 Data Flow

Esse diagrama foi pensado com base no modo em que os dados são manipulados tendo como ator uma instrução. Ou seja, responder a pergunta: “para qualquer instrução, de que modo essa instrução pode interagir com um dado?” Como resposta, um dado pode ser requisitado, ou ele pode ser armazenado. E isso pode ocorrer principalmente em dois módulos de memória: a memória de dados e a memória de registrador. Independente do componente, é necessário obter o endereço onde o dado está ou será armazenado. Se a interação ocorrer com a memória de dados, é preciso obter um endereço que está armazenado em um registrador. Se a interação ocorrer com o banco de registrador, o endereço obtido será o endereço do registrador.

3.3 Modules

Para esse diagrama foi considerado que cada módulo é um ator diferente, e os diagramas mostram o que cada módulo é capaz de realizar. Algumas interações entre eles são demonstradas, mas em geral foram considerados casos de uso independente.

3.4 IO Module

Similar à ideia do diagrama anterior, esse arquivo mostra especificamente casos de uso com relação ao módulo de entrada e saída. As operações de entrada e saída contém bastante interação com os dois clocks do processador (que são sincronizados através do módulo Timer) e devem estar corretamente sincronizadas, por isso foi criado um arquivo separado para esse caso.

4. Atividades

Os diagramas de atividade estão dentro do diagrama de caso de uso <Interact>, e dentro do sistema <CPU>. Foram feitos 3 diagramas:<Data manipulation>, <Deviation>, <Input/Output>.

4.1 Data Manipulation

Representa as instruções que não envolvem desvio ou módulo de entrada e saída. Resumidamente envolve leitura de dados, processamento e escrita de dados. Basicamente seria decodificar uma instrução; obter os dados de cada módulo; selecionar quais dados serão usados através de sinais de controle enviados pela unidade de controle; a ULA checar qual operação realizará; escrita de dados com base nas flags; fim do ciclo.

4.2 Deviation

Representa as instruções de desvios. As primeiras ações são similares ao caso anterior: obter a instrução, decodificar e sinalizar flags de controle. Além disso, é preciso considerar 3 possíveis endereços de desvios: Jump (O endereço vem do registrador), branch (Endereço vem de um sinal imediato) e normal (endereço é o próximo sequencial, PC = PC + 1). E os desvios podem ser condicionais ou não. Para isso é preciso checar se uma *flag* de comparação está ativa ou não. Ao final, o endereço escolhido será o próximo a ser processado.

4.3 Input/Output

Representa as instruções de entrada/saída. Como já dito anteriormente, existem dois clocks ocorrendo e que são síncronos. Quando o clock da cpu está em alto, a leitura de dados de operação ocorre, e é calculado o endereço do dado que será enviado para o output ou o endereço da memória onde será armazenado o input. Além disso, na descida do clock do IO, é verificado se o usuário já enviou o dado de envio, caso não, o clock da cpu é congelado (flag await = 1) e voltará ao normal somente após receber o input. No caso de output, não precisa ocorrer o congelamento da cpu. Por fim, o ciclo se repete.