Processador RISC de 32 bits

Implementado em VHDL, com instruções do MIPS DLX

Design de Computadores 2018.2 - Insper

Antonio Sigrist, Carlos Rosa, Felipe Buniac, Isabella Oliveira e Vitória Mattos

1. Objetivo

Este projeto tem como objetivo a implementação de um processador de 32 bits em VHDL no chip programável FPGA DE2-115 e por consequência estimular o aprendizado e a fixação dos conteúdos da disciplina Design de Computadores, além de desenvolver a habilidade de utilizar esses conhecimentos em uma aplicação real.

2. Desenvolvimento

Para o funcionamento do processador de 32 bits será necessária a implementação de uma unidade lógica aritmética (ULA) também de 32 bits com as seguintes funções:

Do tipo R: ADD (soma), SUB (subtração), AND (“e” lógico), OR (“ou” lógico) e SLT (subtração e transferência de bit, de MSB para LSB).

Do tipo I: LW (soma) , SW (subtração) e BEQ (subtração e detecção de resultado zero).

Para instruções do tipo J não será necessário utilizar a ULA.

Desta forma, a ULA completa, com verificação de overflow, carry in e carry out é a representada na **Figura 1**, sendo que a porta lógica sinalizada com o símbolo + está representada na **Figura 2**. Os pontos de controle da ULA são InverteA, InverteB e Seleção e a ordenação dos bits da palavra de controle são:

Bit 3: InverteA, bit 2: InverteB, Bit 1: bit 1 da seleção e bit 0: bit 0 da seleção.

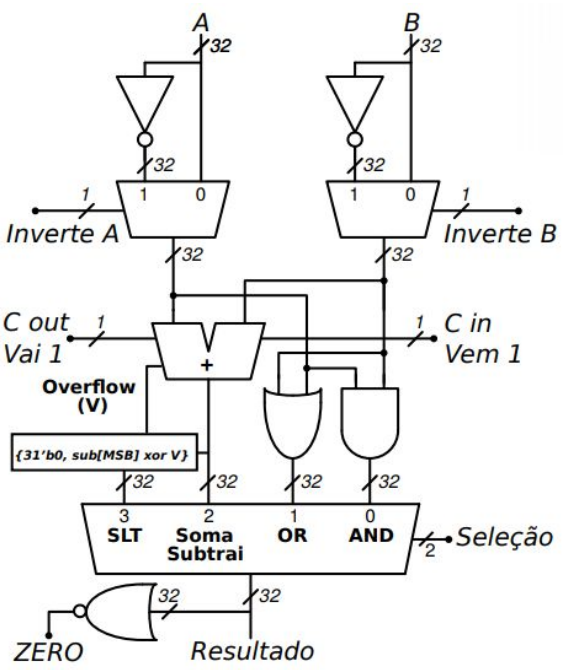

Figura 1: Implementação da ULA

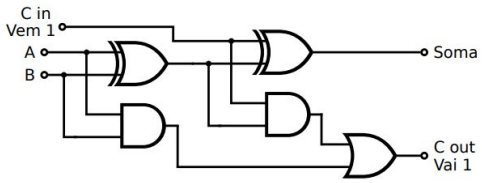

Figura 2: Implementação do carry in e carry out na ULA

Além das instruções citadas acima, que fazem parte do subconjunto “A”, foram implementadas também algumas instruções do subconjunto “B”, que adiciona as instruções a seguir:

- addi: soma com imediato
- subi: subtração com imediato (que foi implementada porém não está funcional)
- andi: E lógico com imediato
- ori: OU lógico com imediato
- slti: set if less than (comparação menor que imediato)

A alternativa escolhida para garantir que o processador tenha um bom desempenho é a utilização de um pipeline para que duas ou mais instruções possam ser executadas em paralelo ao invés de terem que esperar o ciclo de clock. Para isso, a informação criada por uma instrução em cada etapa do pipeline deve ser salva, quando necessário, para ser utilizada nas etapas seguintes. O fluxo de dados do projeto com a utilização de pipelines está representado na **Figura 3**.

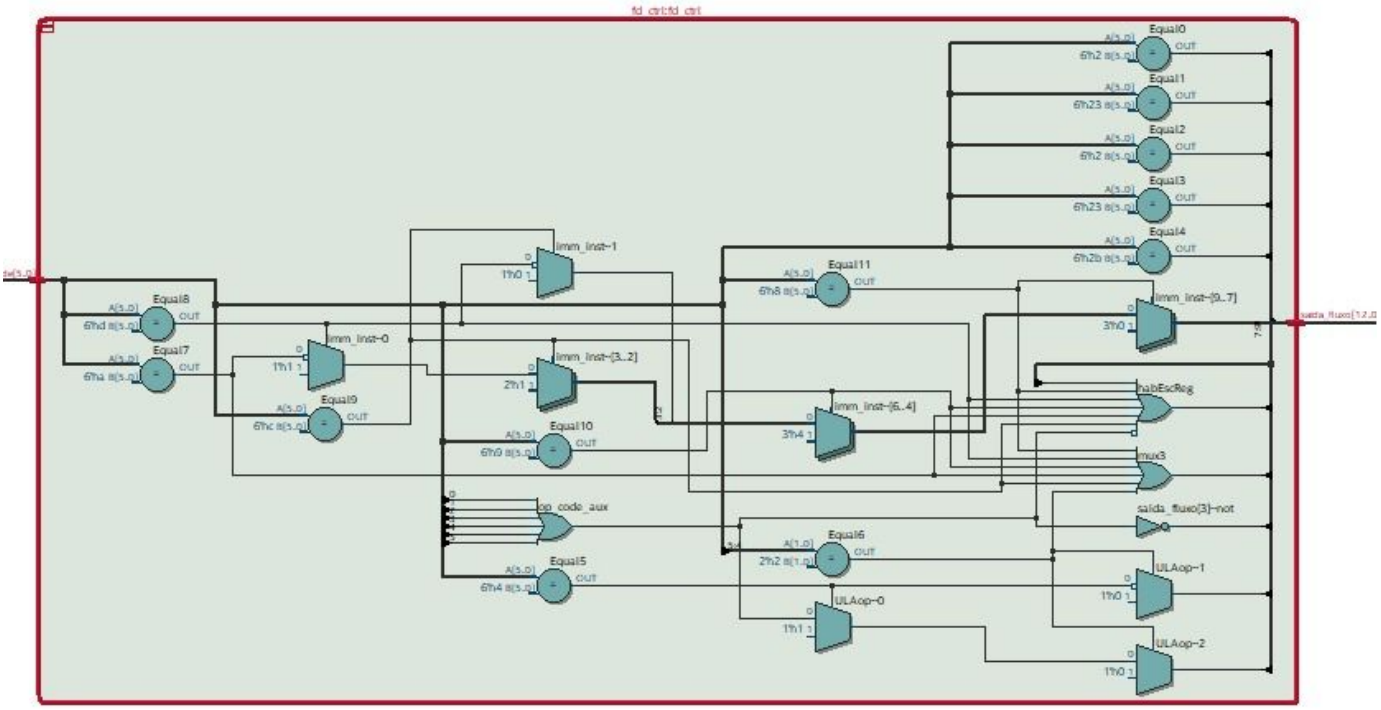

Figura 3: Fluxo de dados do processador 32 bits com pipeline.

Para que o tempo total de execução de cada instrução (latência) não se altere, a execução das instruções é sobreposta no tempo. Assim, cada etapa do pipeline executa uma instrução diferente.

O acesso à memória só ocorre com load word e store word. Para calcular o endereço da memória, a etapa de execução é utilizada. O acesso ocorre na etapa seguinte e os registradores, PC e de fronteira entre estágios são atualizados a cada ciclo de “clock”.

Para implementar as novas instruções o tamanho do segundo pipeline foi alterado para 150 pois foram adicionados 3 bits provenientes do fluxo de dados (que também aumentou de tamanho para 13), para identificar as novas funções.

3. Controle da ULA

Para verificar o tipo de instrução da função que a ULA irá executar:

- Caso o Opcode seja igual a 0x00, a instrução é do tipo R. Desta forma, o conteúdo da “funct” determinará qual será a instrução.
- Caso o Opcode seja diferente de 0x00, a instrução é do tipo I ou J. Assim, o conteúdo do Opcode determina qual é a instrução.

4 bits são necessários para fazer o controle da MUX contida na ULA, cada um representa uma instrução.

Instrução Tipo R	**Funct**	**ULA Executa**	**ULActrl**
ADD	10.0000	add	0010
SUB	10.0010	sub	0110
AND	10.0100	and	0000
OR	10.0101	or	0001
SLT	10.1010	slt	0111
Instrução Tipo I	**Opcode**	**ULA Executa**	**ULActrl**
LW	10.0011	add	0010
SW	10.1011	add	0010
BEQ	00.0100	sub	0110
J	00.0010	XXX	XXXX

4. Apresentação

O MIPS funciona de forma que todas as operações da ULA (AND, SUB, ADD e OR) mostram seus operandos e resultado, sendo que no segmento 0 da placa FPGA aparece o resultado e nos segmentos 1 e 2 aparecem os valores que serão manipulados pela operação. Em toda a operação da ULA, um clock antes que essa aconteça, é exibido nos 6 primeiros LEDs (do 0 ao 6) da placa FPGA o valor do funct correspondente, acendendo os LEDS com bit igual a 1.

Para as operações de LW e SW é exibido o valor que está sendo lido ou escrito, respectivamente, nos displays 4 e 6. Nos 6 últimos LEDs,/ (do 17 ao 12) é exibido o opcode da operação realizada.

Observação: o Nop carrega o valor 9 (sem significado).

5. Problemas e soluções

O maior problema encontrado durante o desenvolvimento do projeto foi que havia um equívoco no diagrama apresentado em sala de aula nos sinais passados para o mux MemtoReg e ele impossibilitou por muito tempo que o projeto funcionasse da forma correta. Quando os sinais foram invertidos o projeto passou a funcionar corretamente.