PCS-3225 Sistemas Digitais II — Abril de 2024 - v3

Calculadora Programável

EP 02 — Bruno Albertini, Edson Gomi e Ricardo Lera

Neste EP vamos exercitar o que foi aprendido até o momento para construir uma calculadora programável.Para começar, precisamos de uma ULA (Unidade Lógica e Aritmética, ou do inglês *Arithmetic Logic Unit*), mostrada da Figura 1.

```
module alu
    #( parameter W = 32)
     ( input  [3   :0] ALUctl,
       input  [W-1:0] A, B,
       output [W-1:0] ALUout,
       output Overflow, Zero
     );
```

ALUctl	Operação
0000	A and B
0001	A or B
0010	A + B
0110	A - B
0111	A menor que B
1100	A nor B

Figura 1: Módulo ULA

O parâmetro da ULA é o tamanho das palavras a serem operadas, em bits (o padrão é 32b). As entradas de dados são `A` e `B`, e a saída de dados é `ALUOut`. A saída `Zero` é alta se e somente se o resultado na saída de dados é zero. A entrada de controle `ALUctl` determina qual operação a ULA deve fazer, mostrada na tabela da Figura 1. As entradas são consideradas em complemento de base quando aplicável e a operação $A < B$ produz o equivalente ao número 1 se $A < B$ e o equivalente ao número zero caso contrário.

A saída `Zero` é verdadeira se e somente se o resultado em `ALUout` é zero, e a saída `Overflow` é verdadeira se a operação sendo realizada resulta em *overflow*. No caso de overflow o resultado ainda precisa ser produzido. O *overflow* só é aplicável se a operação for aritmética, caso contrário o valor na saída não tem significado e pode ser qualquer um.

O segundo módulo que precisamos é de um registrador, ou mais precisamente um banco de registradores RF (do inglês *Register File*), que está mostrado na Figura 2.

O parâmetro do banco de registradores é o tamanho das palavras que ele armazena. O banco é composto por 32 registradores (note que isso não tem relação com o tamanho da palavra) e permite uma escrita e duas leituras simultâneas. As entradas `Read1` e `Read2` determinam a leitura assíncrona do registrador indicado (do registrador 0 ao registrador 31), cuja saída estará na saída `Data1` ou `Data2` respectivamente. Exemplo: se `Read1=1010`, o conteúdo do registrador 10 estará na porta `Data1`. A escrita é síncrona e acontece na borda de subida do *clock*, se e somente se o sinal `RegWrite` estiver alto. A escolha de qual registrador será escrito é feita pela porta `WriteReg` e o valor a ser escrito é o presente na entrada `WriteData`. Exemplo:

```
module registerfile
    #( parameter W = 32)
     ( input  [4   :0] Read1, Read2, WriteReg,
       input  [W-1:0] WriteData,
       input  RegWrite,
       output [W-1:0] Data1, Data2,
       input  clock
     );
```

Figura 2: Módulo Banco de Registradores

se `WriteReg=11111`, o valor de `WriteData` será escrito no registrador 31 na próxima borda de subida do *clock* onde o `RegWrite` for alto.

Uma exceção é o registrador 0 (zero), que sempre retorna zero quando lido e não aceita escrita (a escrita neste registrador é sempre ignorada).

Ainda precisamos de uma memória para armazenar as operações que a calculadora irá realizar. Como já estudamos memórias, a memória ROM já carregada com algumas operações foi fornecida. Estude os arquivos fornecidos e entenda como a ROM funciona.

Por último, montaremos a calculadora. A nossa calculadora programável é um dispositivo que lê da ROM as operações que deve fazer, começando do endereço `0x00` e terminando no endereço `0xFF`. A cada ciclo, ela executa a operação que está naquele endereço.

As operações são determinadas pelo conteúdo da memória naquela posição. Cada posição da memória possui uma palavra de 32b, que indica a operação a ser realizada, e pode ou não carregar números. Os bits da palavra de memória são numerados de 31 (mais significativo) até 0 (menos significativo). Os bits de 11 a 7 da operação sempre indicam o registrador de destino, ou seja, onde será gravado o resultado no banco de registradores. Os bits de 19 a 15 sempre indicam o registrador que será lido para a entrada A da ULA, ou seja o operando da entrada A. Já o segundo operando pode ter duas origens possíveis. Se o bit 5 for um, a operação acontecerá com o registrador indicado pelos bits de 24 a 20, cujo conteúdo será colocado na entrada B da ULA. Caso o bit 5 for zero, a operação acontecerá entre o registrador de origem A (conforme explicado) e um número embutido na própria operação, indicado pelos bits de 31 a 20 (12 bits), que deve ser interpretado como um inteiro em complemento de base.

A operação a ser realizada é indicada pelos bits 14 a 12 e adicionalmente pelo bit 30. A tabela da Figura 3 mostra as operações possíveis.

Exemplo 1: a primeira palavra da memória, na posição `0x00`, é `00100013`. O bit 5, que nesta palavra é 0, indica que a operação carrega um número. Este número sempre estará na posição de 31 a 20, formando um número de 12b que deve ser interpretado como um inteiro em complemento de base. Neste exemplo, o número é 1. A operação a ser realizada é fornecida pelos bits de 14 a 12 (três bits), que no exemplo são `000`, portanto devemos observar o bit 30, que é `0`, indicando uma soma. A operação a ser realizada então é A + número. A entrada A da ULA é sempre fornecida pelos bits 19 a 25, que no exemplo é `00000`, portanto a entrada A será o conteúdo do registrador 0. Faremos então R0 + 1. O resultado deve ser guardado no registrador indicado pelos bits de 11 a 7, que também são `00000`, indicando que o resultado também deve ser guardado no registrador 0. A operação completa então é R0 = R0 + 1.

Exemplo 2: a última palavra da memória, na posição `0xFF`, é `40750533`. O bit 5, que nesta palavra é 1, indica que a operação não carrega um número, portanto será entre registradores. A operação a ser realizada é fornecida pelos bits de 14 a 12 (três bits), que no exemplo são `000`, portanto devemos observar o bit 30, que é `1`, indicando uma subtração. A operação a ser realizada então é A - B. A entrada A da ULA é sempre fornecida pelos bits 19 a 25, que no exemplo é `01010`, portanto a entrada A será o conteúdo do registrador 10. A entrada B, como esta é uma operação entre registradores, é o conteúdo do registrador 7, indicado pelos bits de 24 a 20, que são `00111`. Faremos então R10 - R7. O resultado deve ser guardado no registrador indicado pelos bits de 11 a 7, que também são `01010`, indicando que o resultado também deve ser guardado no registrador 10. A operação completa então é R10 = R10 - R7.

```
module calculadora #(
    parameter W = 32
) (
    input clock, reset, opera,
    input [4:0] read,
    output [W-1:0] data
);
```

bit 5	bits 14-12	bit 30	Operação
1	000	0	A + B
1	000	1	A - B
1	110	-	A or B
1	111	-	A and B
1	010	-	A menor que B
0	000	-	A + número
0	100	-	A or número
0	111	-	A and número
0	010	-	A menor que número

Figura 3: Módulo Calculadora

O módulo calculadora tem um parâmetro, que indica o tamanho da palavra da ULA e o tamanho dos registradores do banco (o número de registradores é sempre 32 e a ROM sempre tem 16 palavras de 32b). Quando o sinal de `opera` é alto, a calculadora lê um endereço de memória, executa a operação que está naquele endereço e vai para o próximo, com as operações começando na borda de subida do *clock*. Quando uma operação é iniciada, ela sempre termina. Quando o sinal de operação está desativado antes da borda de subida do *clock*, a calculadora não realiza nenhuma operação. Quando o sinal volta a ser alto, a calculadora continua de onde parou (a partir da última operação ainda não executada, inclusive). Enquanto o sinal de operação está desativado, a calculadora coloca o valor do registrador apontado pelo sinal `read` na porta `data`. Exemplo: se o sinal `opera` é zero e `read=01010`, o valor do registrador 10 pode ser visto na porta `data`. Os sinais `read` e `data` não tem significado enquanto `opera` está ativo.

Para este EP, você deve fazer a ULA, o banco de registradores e a calculadora, seguindo as assinaturas dos módulos como indicado. Envie para o juiz somente um arquivo com o seu Verilog contendo toda sua solução, para cada uma das atividades. O juiz não tem nenhum módulo além da ROM e do arquivo de carga `rom.hex`. Não envie para o juiz o módulo ROM ou o arquivo de carga em nenhuma submissão.