MÁQUINA DE CALCULAR BASEADA EM STACK

Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

Laboratório de Sistemas Digitais

Grupo 5, MG 2, Turma P3
Rodrigo Rosmaninho, Rita Amante
(88802) r.rosmaninho@ua.pt, (89264) rita.amante@ua.pt

2017-2018

INTRODUÇÃO

Para este projeto foi implementada uma máquina de calcular com base numa pilha (stack) com profundidade de até 16 níveis de operandos (números inteiros) representados em 8 bits, em complemento para 2, o que implica que a pilha só permite inserir operandos na gama [-128,127].

O utilizador pode introduzir operandos na pilha ou realizar as seguintes operações sobre a pilha: adição, subtração, multiplicação, divisão, remoção do elemento no topo da pilha, e troca entre o elemento no topo da pilha e o imediatamente anterior.

Para além disso, é possível visualizar o valor que se encontra em qualquer posição da pilha, e, se necessário, removê-lo da mesma.

Em conformidade com o que foi pedido no enunciado deste projeto, este foi dividido em 3 fases distintas:

- Fase 1 – Implementação da pilha com uma profundidade de apenas 2 níveis
- Fase 2 – Aumento da profundidade da pilha para 16 níveis
- Fase 3 – Adição de módulos de validação dos dados de entrada/saída

Para além destas, foi desenvolvida ainda uma quarta fase, de forma a tornar possível a obtenção de uma classificação superior a 16 valores:

- Fase 4 – Utilização do display LCD integrado do kit em vez dos *displays* de 7 segmentos, para visualização de dados, resultados, pilha, e mensagens de erro.

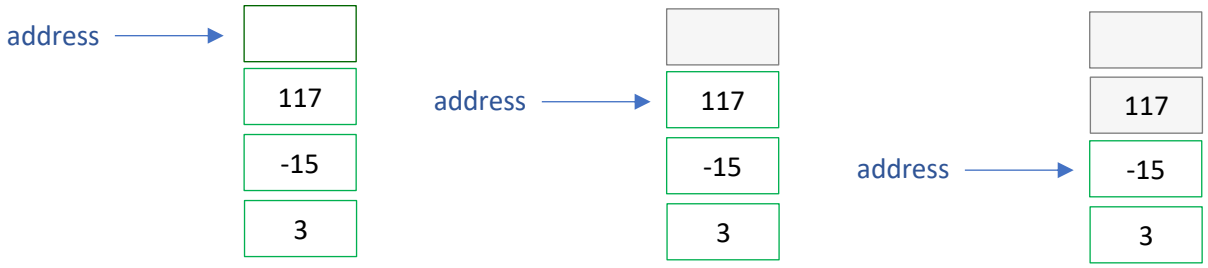
Em anexo poderá ser encontrada uma explicação da arquitetura usada no desenvolvimento desta quarta fase. Assim como um manual do utilizador para todas as fases.

IMPLEMENTAÇÃO

Implementação da Pilha

A pilha foi implementada usando uma RAM 16x8 (16 elementos com 8 bits) com 2 portos de leitura assíncrona e 2 portos de escrita síncrona. Sendo o address do primeiro espaço livre na pilha (elemento seguinte ao topo da pilha) é guardado num módulo à parte.

Como a primeira posição da pilha equivale a address = 0, o valor de address representa, também, o número de elementos na pilha num dado momento, o que constitui uma informação útil.



Para eliminar a posição no topo da pilha basta decrementar o valor de 'address' por 1 unidade. Desta maneira, como 'address' representa o primeiro espaço livre, o módulo de visualização da pilha deixa de poder aceder ao valor nessa posição, tal como o módulo de cálculo.

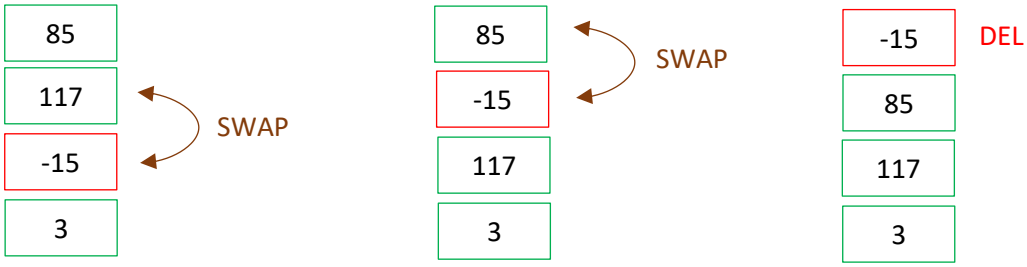
Assim que for introduzido um novo operando esta posição será reescrita e o valor anterior será perdido. Para fazer reset à pilha 'address' passa a ter o valor 0.

Dois portos de leitura facilitam a leitura de dois operandos ao mesmo tempo, para que as operações sejam realizadas. Dois portos de escrita facilitam as operações *SWAP* e DIV, em que é necessário escrever dois valores para a RAM em vez de apenas um.

Implementação da remoção de um elemento numa posição arbitrária da pilha

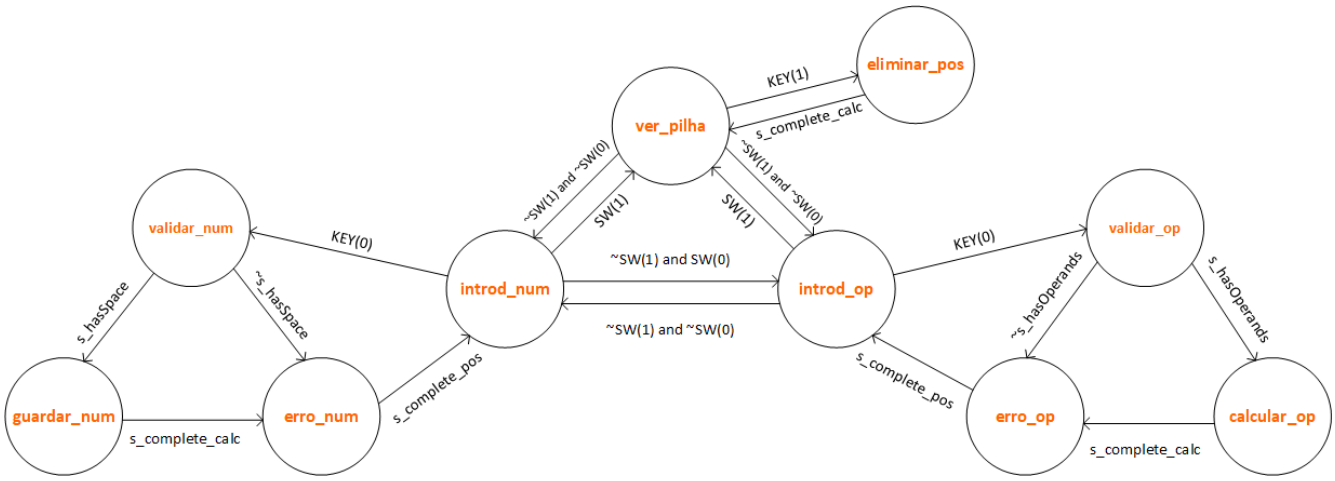
Enquanto se visualiza a pilha é permitido clicar numa KEY que deve eliminar o elemento que se está a visualizar. Não podem, no entanto, ficar posições vazias no meio da pilha.

Como o bloco de cálculo já permitia a execução de uma operação SWAP (e troca entre o elemento no topo da pilha e o imediatamente anterior) foi implementado o seguinte algoritmo:



São efetuadas operações SWAP (incrementando um sinal de address a cada operação) até que o elemento que se quer eliminar esteja no topo da pilha. Quando tal acontecer, efetua-se uma operação normal de remoção do elemento no topo da pilha (decrementando o address global).

Máquina de Estados Finitos - Controlpath



FASES 3 e 4

Estado **ver_pilha** – Ver Pilha

Ativa o bloco da visualização da pilha, através da ativação da saída en_verPilha.

Estado **introd_num** – Introdução de Operando

Ativa o bloco de introdução de operandos, através da ativação da saída en_operando.

Estado **introd_op** – Introdução de Operador

Ativa o bloco de introdução de operadores, através da ativação da saída en_operador.

Estado **validar_num** – Validação da Existência de Espaço na Pilha

Ativa o bloco da validação pré-inserção de operandos na pilha, através da ativação da saída en_valOperando.

Estado **validar_op** – Validação da Existência de Operandos Suficientes na Pilha

Ativa o bloco da validação pré-realização de operações, através da ativação da saída en_valOperador.

Estado **guardar_num** – Inserção de Operando na Pilha
Ativa o bloco da inserção de operandos na pilha, através da ativação da saída en_stackOperando.

Estado **calcular_op** – Cálculo do Resultado e Atualização da Pilha
Ativa o bloco de cálculo, através da ativação da saída en_stackOperador.

Estado **erro_num** – Exibição de Erros Relativos à Inserção de Operandos
Ativa o bloco de pós-validação através da ativação da saída en_posOperando.

Estado **erro_op** – Exibição de Erros Relativos à Inserção de Operadores e Cálculo
Ativa o bloco de pós-validação através da ativação da saída en_posOperador.

Estado **eliminar_pos** – Remoção de um Elemento numa Posição Arbitrária da Pilha
Ativa o bloco de remoção de elementos arbitrários da pilha através da ativação da saída en_eliminarPos.

FASES 1 e 2

A máquina de estados finitos do controlpath das fases 1 e 2 é bastante semelhante à apresentada anteriormente.

Mantém todos os estados menos *erro_num* e *erro_op*, que geriam a exibição de mensagens de erro. Como tal, o estado *guardar_num* passa diretamente para *introd_num* quando *s_complete_calc* = '1' e o estado *calcular_op* passa diretamente para *introd_op* quando *s_complete_calc* = '1'.

Periféricos

Foram utilizados os seguintes periféricos do kit DE2-115:

- Interruptores SW0, SW1, SW2, e SW3
- Botões KEY0, KEY1, KEY2, e KEY3
- Luzes LEDR0, LEDR1, LEDR2, LEDR3, e LEDG8
- Displays de 7 Segmentos HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, e HEX7
- Display LCD

ARQUITETURA

O ficheiro top-level do sistema integra 2 módulos essenciais: *controlpath* e *datapath*.

Para além disso inclui debouncers e registos para os periféricos de entrada do kit.

O módulo *controlpath*, que consiste numa Máquina de Estados Finitos (MEF) que gere todo o comportamento do sistema, ativando diferentes componentes do *datapath* através de várias saídas que servem de *enable* a esses componentes, dependendo do estado de alguns dos interruptores (SW) e botões (KEY) da FPGA e também de sinais provenientes do próprio *datapath*.

A arquitetura do *datapath* pode ser sumariada em 3 partes distintas:

- Introdução de dados;
- Execução de operações, leitura e escrita na memória RAM, e validação;
- Conversão e exibição de dados nos periféricos de saída como *displays* e *LCD*.
-

Um diagrama da arquitetura do *datapath* das fases 1,2,3, e 4 pode ser encontrado nas páginas de anexo deste relatório.

Consiste no circuito que gere a memória RAM que implementa a pilha, bem como a introdução de operadores e operandos, a visualização e remoção de elementos da pilha, e a validação de valores de entrada e saída.

Para introduzir um operando é ativado o bloco de introdução de números, que permite ao utilizador incrementar um número visualizável nos displays de 7 segmentos para que este seja mais tarde gravado no topo a pilha pelo bloco de cálculo.

Para introduzir um operador é ativado o bloco de introdução de operadores, que permite ao utilizador alternar entre os operadores disponíveis visualizáveis nos displays de 7 segmentos. O operador escolhido é passado ao módulo de cálculo como entrada.

Para visualizar a pilha é ativado o bloco de visualização da pilha, que permite ao utilizador visualizar a pilha nos displays de 7 segmentos um nível de cada vez, utilizando duas KEYs da FPGA para alternar entre os níveis da pilha. Para além disso, permite usar outra KEY para ativar um bloco que elimina o elemento da pilha a ser atualmente visualizado. Este bloco utiliza também o módulo de cálculo para efetuar a sua função.

O módulo *calculadora* constitui uma parte fulcral do sistema pois gere todas as operações de escrita na RAM. Qualquer operação, remoção, ou inserção é efetuada por este módulo, que também inclui um registo onde fica guardado o valor atual de 'address'. Como tal, recebe sinais de entrada provenientes de todos os módulos mencionados até agora.

Existem também módulos de conversão de palavras *signed* de 8 bits em BCD, BCD em binário utilizável pelos displays de 7 segmentos, e conversores para BCD em ASCII (para a utilização do LCD). Estes módulos recebem valores da RAM, dos módulos de inserção, ou do módulo de exibição de mensagens de erro e convertem os dados de forma a que possam ser apresentados ao utilizador de forma intuitiva (através dos displays ou LCD).

Para decidir (com base no estado atual da MEF) quais os sinais que são convertidos e exibidos ao utilizador num determinado momento, está presente alguma lógica adicional no *datapath*.

VALIDAÇÃO

Foram testados diversos blocos com ficheiros *waveform* (.vwf) de forma a confirmar o seu funcionamento correto.

CONCLUSÃO

Todas as fases foram completadas com sucesso e em conformidade com o que foi pedido no enunciado do projeto. Todas as funcionalidades foram testadas no kit e aparentam estar corretas. Os requisitos para obtenção de classificações superiores a 16 também foram completados.

Divisão do Trabalho

Rodrigo Rosmaninho (50%):

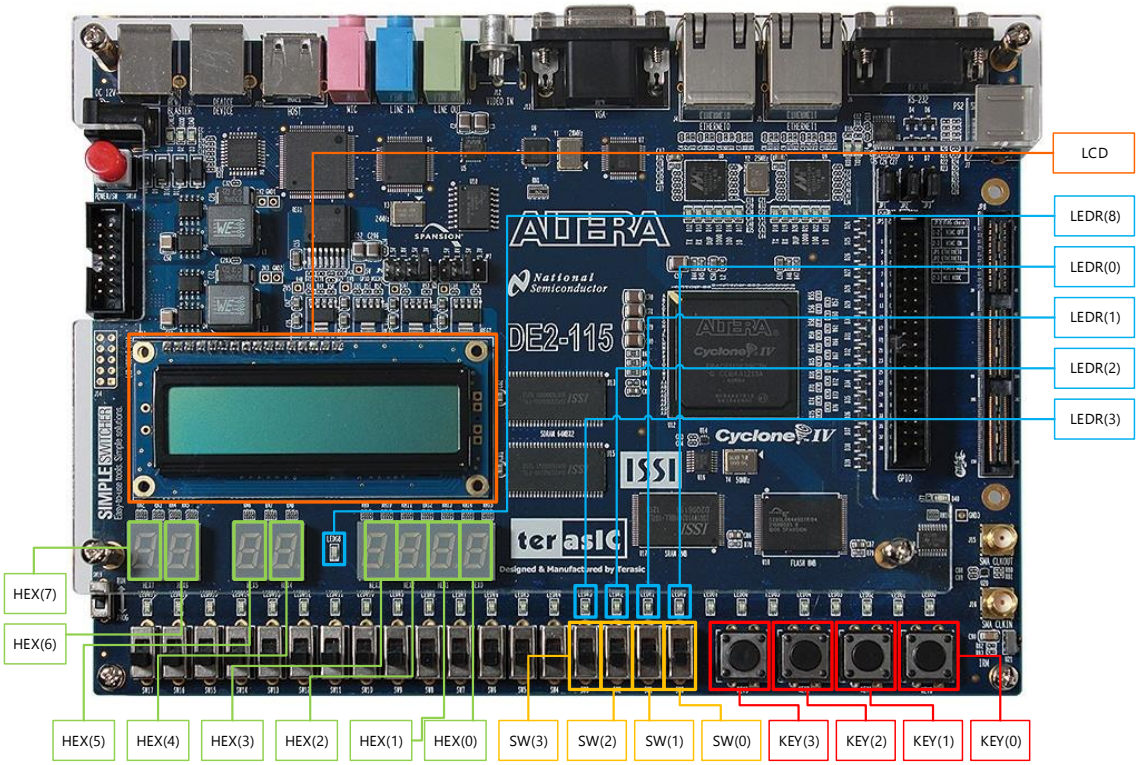
- Máquina de estados do controlpath;
- Memória RAM;
- Blocos de Introdução;
- Bloco de remoção dos elementos arbitrários da Pilha.
- Módulo de Cálculo.
- Módulos referentes à utilização do LCD

Rita Amante (50%):

- Passagem de valores de entrada por deboucers e registos;
- Testbench
- Bloco de validação;
- Bloco de apresentação de erros
- Bloco de visualização da Pilha;
- Blocos de conversão e exibição nos displays

ANEXO

Manual do utilizador



Introdução de um operando:

Para a introdução de um operando, é necessário certificar se SW(0) não está ativo. Caso se pretenda um número negativo, é preciso ativar SW(2), caso contrário, SW(2) não pode estar ativo.

Ao pressionar KEY(3), KEY(2) e KEY(1) incrementa-se ao número uma centena, uma dezena e uma unidade, respetivamente.

É de notar que só é permitido inserir operandos pertencentes à gama [-128,127], uma vez que estes são representados em 8 bits, com sinal, em complemento para 2. Se o utilizador tentar ultrapassar o intervalo, o número apresentado será zero.

Quanto à submissão do número, pressiona-se KEY(0).

Se o utilizador tentar introduzir mais que 16 elementos na pilha, será ativado o LEDR(0) que é uma indicação de espaço insuficiente na pilha.

Introdução de um operador:

O utilizador deverá ativar SW(0) para escolher uma operação.

Por *default*, a primeira operação apresentada é a adição, de seguida a subtração, multiplicação, divisão (inteira), del e, por fim, *swap.* Para escolher a operação, é necessário pressionar KEY(1) até aparecer a operação que se pretende.

Quanto à submissão do número, pressiona-se KEY(0).

É importante salientar que, quando a pilha não tem operandos suficientes para realizar a operação, o LEDR(1) ativa, quando o resultado da operação resulta em *overflow*, o LEDR(2) fica ativo e quando a operação da divisão por zero é realizada com sucesso, o LEDR(3) fica ativo.

Visualização da pilha:

O utilizador, para visualizar a pilha, necessita de ativar SW(1).

Para percorrer os elementos da pilha, pressionar KEY(1) caso se pretenda o elemento seguinte e KEY(0) para o elemento anterior.

Nos *display* de 7 Segmentos, é possível observar a posição onde se encontra o elemento na pilha, em HEX(7) e HEX(6). Em HEX(5) E HEX(4) observa-se o número de elementos que a pilha contém. Em HEX(3), HEX(2), HEX(1) e HEX(0) observa-se o elemento da pilha.

Caso o utilizador esteja na fase 4, será possível observar a posição onde se encontra o elemento na pilha e o número de elementos que a pilha contém no *display* LCD.

Reset da Máquina de Calcular:

O utilizador poderá fazer *reset* da Máquina de Calcular ativando SW(3).

Diagrama do *datapath* para as fases 1,2, e 3

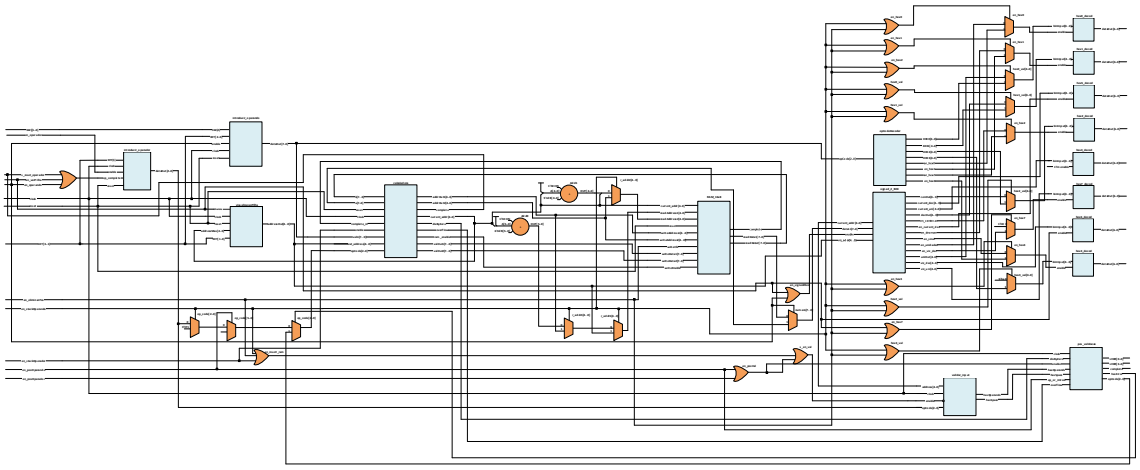


Diagrama do *datapath* para a fase 4