Instituto Superior Técnico – Universidade de Lisboa
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Ano Lectivo 2014 / 2015
Lisboa, 19 de Outubro de 2014
Projecto de Sistemas Digitais

PROJECTO DE SISTEMAS DIGITAIS

Projecto 1

Unidade Lógico-Aritmética Simples

Grupo n.º 23
3ª das 17h00 – 18h30, LSD2

Maria Margarida Dias dos Reis, n.º 73099

João André Catarino Pereira, n.º 73527

Índice

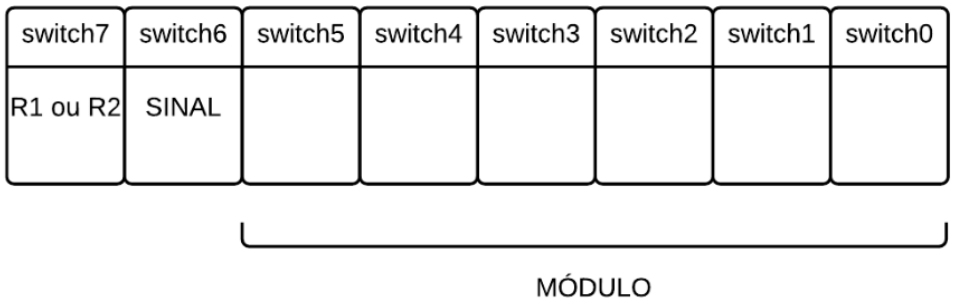
Funcionamento da Unidade Lógico-Aritmética

Para a realização da calculadora lógico-aritmética simples recorreu-se aos 4 botões da placa, os 8 interruptores e os *displays* de 7 segmentos.

As operações da calculadora são realizadas entre os valores armazenados em dois registos diferentes, R1 e R2. Para fazer *load* de um determinado número para os registos recorre-se primeiramente aos interruptores. O sinal que irá ser guardado nos registos é o sinal ENT e é um número no intervalo [-63; +63] em formato sinal-módulo. Assim, nos 6 interruptores da direita selecciona-se o valor do módulo de ENT (em binário) e no segundo interruptor a contar da esquerda seleciona-se o sinal – quando o interruptor está ligado corresponde a sinal negativo e se desligado então o sinal ENT é positivo.

Para se poder visualizar nos *displays* qual o valor actualmente armazenado no registo R1 ou no registo R2 recorre-se ao último interruptor a contar da esquerda. Quando o interruptor está inactivo então é visualizado o valor de R1 e se activo então é visualizado o valor de R2.

A selecção do sinal ENT está esquematizada na figura abaixo.



Uma vez seleccionado o sinal ENT usa-se o botão 1 para fazer *load* desse valor para o registo R1 e o botão 2 para fazer *load* para o registo 2.

Para fazer uma operação utiliza-se 5 interruptores da seguinte maneira:

switch7	switch6	switch5	switch4	switch3	switch2	switch1	switch0
R1 ou R2			SRA	NOR	MULS	SUBS	ADDS

Uma vez seleccionada com os interruptores a operação pretendida pressiona-se o botão 3 para que esta seja realizada.

Para se fazer *reset* geral do sistema, o que inclui o *clear* dos registos, pressiona-se o botão 0.

Máquina de Estados

A máquina de estados da calculadora é apresentada em baixo.

Diagrama de Blocos

Segue em Anexo 1 um diagrama de blocos geral do nosso circuito e em Anexo 2 um diagrama de blocos detalhado.

Arquitectura da Calculadora

A calculadora é constituída por uma unidade de controlo (*control.vhd*), uma unidade de dados – registos e ALU (*datapath.vhd*) e uma unidade de interface aos *displays* de 7 segmentos (*disp7.vhd*).

A unidade de controlo é responsável pela máquina de estados. Parte-se do estado inicial – "initial" – e se for pressionado algum botão avança-se para o estado correspondente, caso contrário permanece-se no estado inicial. Quando se muda de estado muda-se o valor de alguns sinais de estado, estes sinais servem para a unidade de controlo comunicar com a *datapath*.

Sinais de estado	Descrição
"oper"	Indica se nesse estado se realiza uma operação aritmética ("oper" = 1) ou *load* de um dos registos ("oper" = 0).

“sel_mux”	Sinal que controla o *multiplexer* responsável pela selecção do que escrever no registo R2 – ENT (“sel_mux” = ‘0’) ou o resultado de uma operação aritmética (“sel_mux” = ‘1’).
“en_r1”	Sinal que permite escrita no registo R1 (“en_r1” = ‘1’).
“en_r2”	Sinal que permite escrita no registo R2 (“en_r2” = ‘1’).

Os estados de *load* e o estado de operação têm sempre como estado seguinte o estado final – “end”. Neste estado colocam-se todos os sinais de estado a zero e verifica-se que se não estiver a ser pressionado nenhum botão a máquina retoma ao estado inicial.

Por opção, quando se mantém um botão pressionado durante algum tempo (tempo que seria suficiente para a realização consecutiva da operação correspondente a esse botão) apenas se realiza uma vez o procedimento desejado e depois disso o resultado permanece inalterado. Esta opção foi tomada porque se quis fazer da calculadora projectada uma calculadora o mais próximo possível da realidade.

Relativamente à *datapath* esta é responsável por converter o sinal ENT do formato sinal-módulo para o formato de complemento para dois, pois esta é a representação numérica interna dos registos R1 e R2. Esta conversão é feita verificando qual o valor do interruptor 6 (bit de sinal) e se for um sinal positivo então o valor em complemento para dois permanece igual, porém se for um sinal negativo o valor em complemento para dois requere que se negue os bits do módulo (negar os valores dos interruptores 5 a 0) e de seguida se some 1. Aquando da conversão para complemento para dois faz-se também a concatenação de bits de sinal (‘0’ para um sinal positivo e ‘1’ para um sinal negativo), ficando o sinal “ent” da *datapath* um sinal de 13 bits.

A selecção da operação aritmética é feita com um *multiplexer* em que os bits de controlo são os dos interruptores 4 a 0. O resultado da operação aritmética é então escrito no registo R2 quando “sel_mux” = 1, caso contrário é escrito o valor de ENT.

As operações de soma, subtracção e multiplicação são feitas com recurso a operadores aritméticos e o NOR é feiro com recurso a um operador lógico.

O *shift right arithmetic* corresponde a um *shift right* mas a posição vazia do bit mais significativo é preenchida com uma cópia do bit mais significativo original. O número de vezes que nesta calculadora é realizado um SRA depende do resto da divisão por 8 do valor armazenado no registo R1. O resto da divisão de um número binário por 8 corresponde aos 3 bits menos significativos desse número, ou seja, no máximo é possível

fazer 7 vezes um SRA. Neste caso o SRA é feito para todos os casos possíveis, ou seja, desde apenas 1 *shift right arithmetic* a 7 *shift right arithmetic* e depois é usado um *multiplexer* para seleccionar qual o SRA pretendido em que os bits de selecção são os 3 bits menos significativos de R1.

Uma vez que R2 é um registo de 13 bits com representação numérica interna de complemento para dois apenas permite armazenar número entre o intervalo [-4095; +4095] e como tal é necessário verificar possíveis casos de *overflow*. Esta verificação é feita apenas para a soma, a subtracção e a multiplicação, uma vez que com um NOR e um SRA não é possível ultrapassarem-se os 13 bits definidos.

Para o caso da soma sabe-se que ocorre *overflow* se os dois operandos têm o mesmo sinal e o resultado tem sinal contrário. Para o caso da subtracção ocorre *overflow* se o operando 1 tiver sinal positivo, o operando 2 tiver sinal negativo e o resultado tiver sinal negativo ou se o operando 1 tiver sinal negativo, o operando 2 tiver sinal positivo e o resultado tiver sinal positivo.

A multiplicação de um sinal de 13 bits por um sinal de 7 bits requere que o resultado seja guardado num sinal de pelo menos 20 bits, sendo que depois só se guarda no registo R2 os 13 bits menos significativos. Assim, para apurar se há ou não *overflow* recorre-se aos 7 bits que foram descartados, sabendo que se esses bits forem todos iguais entre si, ou seja, todos a ‘1’ ou todos a ‘0’ então não houve *overflow*. Sempre que é verificado um caso de *overflow* é feito *clear* ao registo R2.

Em relação à unidade de interface aos *displays* de 7 segmentos, cada *display* corresponde à representação em hexadecimal de um conjunto de 5 bits. Para a representação hexadecimal são necessários apenas 4 bits mas existe um quinto bit extra para que se possa representar o sinal de menos como apenas um traço para ser mais intuitivo ao utilizador.

Simulação

Nesta simulação foram executados três *loads* e cinco operações. O sinal “instr” (botões) representa se a operação é um dos *loads* ou uma operação aritmética. O sinal "data_in" (valores dos interruptores) representa os valores dos loads e os identificadores das operações. O resultado final está no sinal "saída" em complemento para dois.

Após o sinal "rst" ficar a 1 começa-se a executar as instruções do *testbench*. Em primeiro lugar fez-se *load* do valor 5 no registo R1 ("instr" = 001) e depois fez-se *load* do valor -5 no registo R2 ("instr" = 010). De seguida, procedeu-se ao teste de cada operação ("instr" = 100) começando pela multiplicação dos registos. O resultado foi -25 e com este valor, primeiro somou-se o registo R1 e depois subtraiu-se o mesmo, resultando primeiro -20 e depois voltando ao valor -25. A seguir realizou-se uma NOR entre -25 e R1 (com valor 5), e obteve-se o resultado certo. Por fim, para executar o SRA fez-se de novo *load* para o R1, desta vez com o valor 2 e realizou-se dois *shift rights arithmetic*.

Em baixo está uma tabela que apresenta os valores armazenados em R1 e em R2 ao longo da simulação no formato de complemento para dois e em decimal.

Operações	Registo R1	Registo R2
load R1	Complemento para dois: 0000101 Decimal: 5	Complemento para dois: 0000000000000 Decimal: 0
load R2	Complemento para dois: 0000101 Decimal: 5	Complemento para dois: 1111111111011 Decimal: -5
Multiplicação	Complemento para dois: 0000101 Decimal: 5	Complemento para dois: 1111111100111 Decimal: -25
Soma	Complemento para dois: 0000101 Decimal: 5	Complemento para dois: 1111111101100 Decimal: -20
Subtracção	Complemento para dois: 0000101 Decimal: 5	Complemento para dois: 1111111100111 Decimal: -25
NOR	Complemento para dois: 0000101 Decimal: 5	Complemento para dois: 0000000011000 Decimal: 24
load R1	Complemento para dois: 0000010	Complemento para dois: 0000000011000

	Decimal: 2	Decimal: 24
SRA	Complemento para dois: 0000010 Decimal: 2	Complemento para dois: 0000000000110 Decimal: 6