AULA PRÁTICA N.º 9

Objetivos

- Implementar em VHDL e testar os seguintes módulos do *datapath*: banco de registos, ALU, unidade de controlo da ALU e *multiplexers*.
- Adicionar os novos módulos à estrutura do *datapath* desenvolvida na aula anterior.

Não faça *copy/paste* do código que foi disponibilizado nas aulas teóricas. Escrever o código VHDL ajuda-o a entender a estrutura e o funcionamento do *datapath*.

Introdução

Nesta sequência de aulas práticas, está a ser implementado o *datapath single-cycle* que foi apresentado nas aulas teóricas. Nesta aula, faremos a implementação VHDL de quatro módulos, nomeadamente: banco de registos, ALU, unidade de controlo da ALU e *multiplexers*. Para a implementação destes módulos tome como referência o código apresentado nas aulas teóricas.

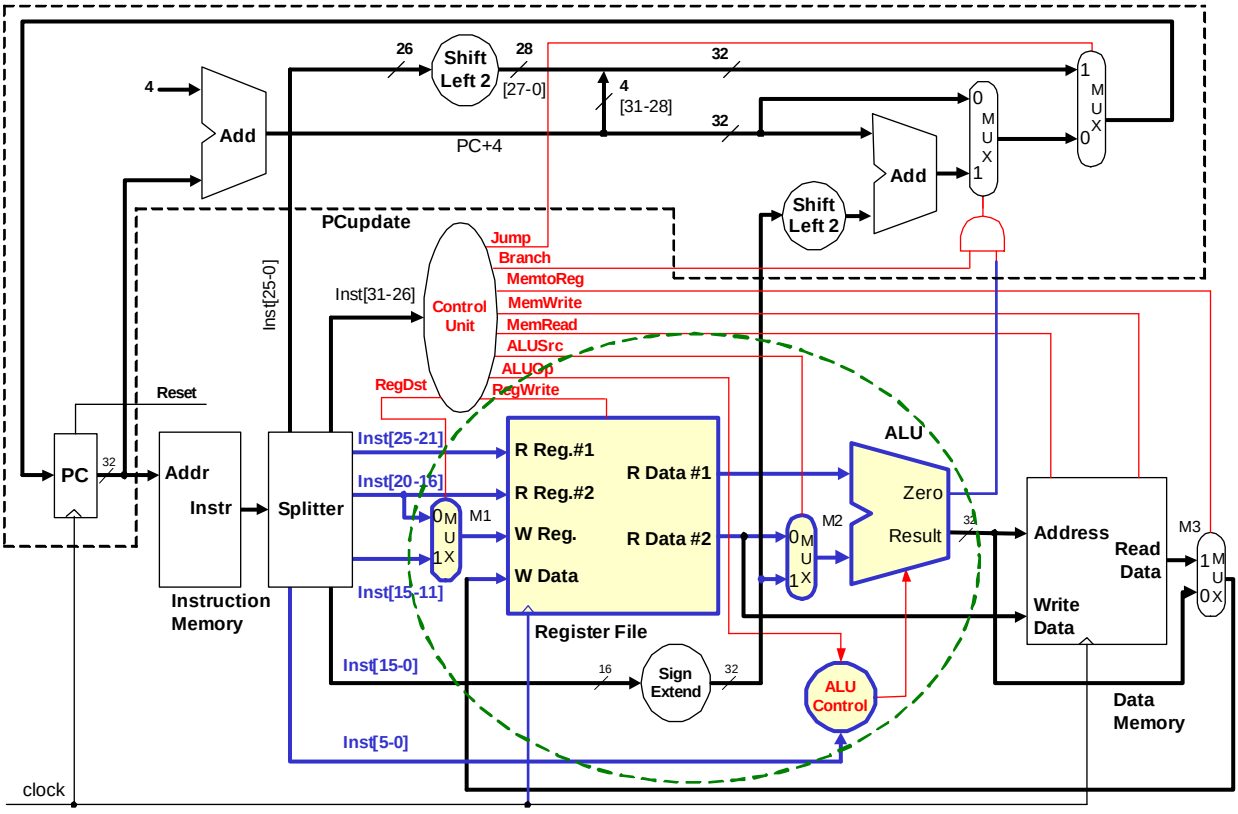

Figura 1. *Datapath single-cycle* com os módulos a implementar nesta aula desenhados com fundo colorido e as ligações a azul.

Banco de registos

O banco de registos agrupa os 32 registos internos de 32 bits do processador e é implementado como uma memória multi-porto: 2 portos de leitura assíncrona e 1 porto de escrita síncrona com sinal de controlo de escrita (*enable*). Veja a implementação apresentada nas aulas teóricas.

ALU

A unidade aritmética e lógica implementa as operações necessárias para a execução do sub-conjunto de instruções considerado na implementação do *datapath* simplificado, nomeadamente, **ADD**, **SUB**, **AND**, **OR**, **XOR**, **NOR** e **SLT** (*set if less than*). A operação a realizar é controlada pelos 3 bits da entrada "ALU control", de acordo com a Tabela 1.

Tabela 1. Operação realizada na ALU em função dos 3 bits de controlo.

ALU control	ALU operation
000	AND
001	OR
010	ADD
011	XOR
100	NOR
110	SUB
111	set if less than

A ALU deve ainda sinalizar a ocorrência de um resultado nulo, colocando a '**1**' a saída "**zero**". Note que a ALU abordada nas aulas teóricas não implementa as operações XOR e NOR.

Unidade de controlo da ALU

A unidade de controlo da ALU produz os sinais que definem a operação da ALU (ALU control, 3 bits), em função do sinal "**ALUOp**" gerado pela unidade de controlo principal e do campo "**funct**" da instrução, tal como especificado na Tabela 2.

Tabela 2. Geração dos 3 bits que definem a operação da ALU (ALU control), em função da entrada "**ALUOp**" e do campo "**funct**" da instrução.

Instruction	ALU operation	OpCode	Funct	ALUOp	ALU control
load word	add	0x23	-	00	010
store word	add	0x2B	-	00	010
addi	add	0x08	-	00	010
branch if equal	subtract	0x04	-	01	110
add	add	0x00	0x20	10	010
sub	subtract	0x00	0x22	10	110
and	and	0x00	0x24	10	000
or	or	0x00	0x25	10	001
xor	xor	0x00	0x26	10	011
nor	nor	0x00	0x27	10	100
set if less than	set if less than	0x00	0x2A	10	111
set if less than imm	set if less than	0x0A	-	11	111

Multiplexers

Os *multiplexers* fazem o encaminhamento de dados no *datapath*, de modo a adequar os recursos às necessidades. M1 permite escolher o registo destino: campo "**rd**" nas instruções tipo R, ou "**rt**" na instrução LW e nas aritméticas e lógicas imediatas. M2 seleciona o 2º operando da ALU: registo lido do banco de registos, codificado no campo "**rt**", ou *offset* da instrução extendido com sinal para 32 bits. M3 encaminha para o banco de registos o valor a escrever no registo destino: valor lido da memória ou o valor calculado pela ALU.

Guião

Parte I

1. Abra no Quartus o projeto que iniciou na aula anterior.

2. Implemente em VHDL o módulo Banco de Registos. Poderá designar este módulo por "**`RegFile.vhd`**". Selecione este ficheiro como o *top-level* do projeto.

3. Usando o simulador *University Program VWF*, simule funcionalmente o módulo "**`RegFile.vhd`**", efetuando operações de escrita em pelo menos dois registos, e operações de leitura desses registos.

4. Acrescente ao banco de registos um ponto adicional de leitura assíncrona e ligue-o, através dos sinais globais "**`DU_RFaddr`**" e "**`DU_RFdata`**", ao módulo de visualização.

5. Implemente em VHDL a ALU de 32 bits. Poderá designar este módulo por "**`ALU32.vhd`**". Selecione este ficheiro como o *top-level* do projeto.

6. Simule funcionalmente a ALU. Verifique a correção do valor presente na saída para as 7 operações implementadas, para os seguintes pares de valores: **`In1=0x12AB0134`** e **`In2=0x2C3154F0`**; **`In1=0xF347C210`** e **`In2=0x01D3E425`**; **`In1=In2`**.

7. Implemente em VHDL a unidade de controlo da ALU. Poderá designar este módulo por "**`ALUcontrol.vhd`**". Selecione este ficheiro como o *top-level* do projeto.

8. Simule funcionalmente a unidade de controlo da ALU, observando a sua saída nas seguintes situações:

`ALUop = "00"`
`ALUop = "01"`
`ALUop = "10"`, para os seguintes valores de "**`funct`**": **`0x20`**, **`0x22`**, **`0x24`**, **`0x25`**, **`0x26`**, **`0x27`**, **`0x2A`**
`ALUop = "11"`

9. Implemente em VHDL um *multiplexer* 2x1 genérico (i.e., de N bits). Poderá designar este módulo por "**`Mux2N.vhd`**". Selecione este ficheiro como o *top-level* do projeto.

10. Simule funcionalmente o *multiplexer* para valores de N de 5 e 32 bits (adeque o valor por defeito da constante genérica do módulo em função da simulação que estiver a realizar).

Parte II

1. Abra o ficheiro "**`mips_single_cycle.vhd`**" e selecione-o como *top-level* do projeto. Instancie os módulos que desenvolveu nesta aula e faça as respetivas ligações ao restante diagrama. Ligue a saída da ALU ao porto de escrita do Banco de Registos, como indicado na Figura 2 a tracejado azul (esta ligação é feita apenas para teste dos módulos já desenvolvidos e deverá ser removida no final do guião).

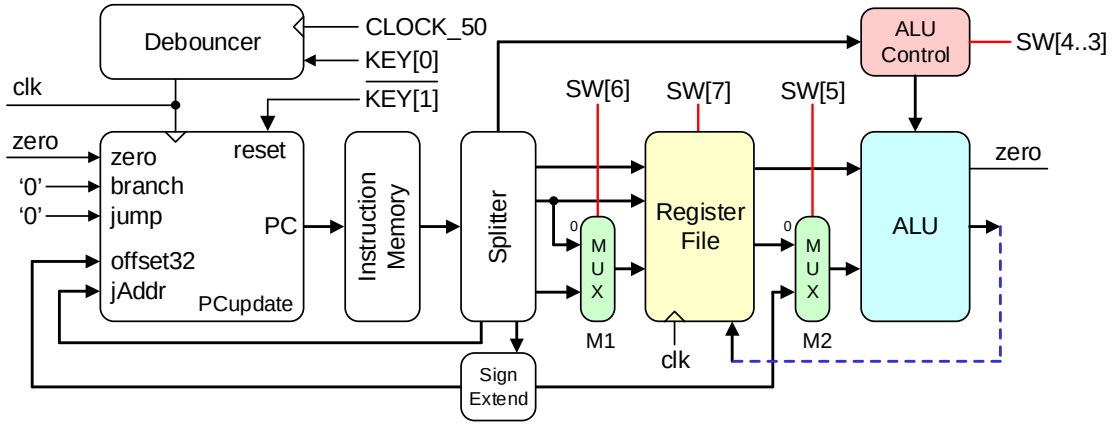

Figura 2. Diagrama de blocos simplificado com os blocos já implementados.

2. Ainda sem a unidade de controlo implementada, é possível testar o funcionamento integrado dos módulos já desenvolvidos gerando manualmente os sinais de controlo. Para isso ligue (ver Figura 2): "**`ALUOp`**" a **`SW[4..3]`**, sinal de controlo do Mux M2, "**`ALUSrc`**", a **`SW[5]`**, sinal de Controlo do Mux M1, "**`RegDst`**", a **`SW[6]`** e o sinal de controlo da escrita no Banco de registos, "**`RegWrite`**", a **`SW[7]`**.

3. Nesta fase, enquanto não é incluída a memória de dados, poderá ligar a saída da ALU ao módulo de visualização através do sinal global "**`DU_DMdata`**".

4. Efetue a síntese e a implementação do projeto e programe a FPGA.

5. Preencha a Tabela 3 com o código máquina e o endereço de memória de cada uma das instruções, sinais de controlo necessários para a sua execução e resultado produzido.

6. Execute o programa que reside na memória de instruções. Para isso, faça *reset* (não esqueça que o *reset* é síncrono) e, de seguida, defina os sinais de controlo para a primeira instrução. Avance depois para a segunda instrução premindo a tecla que está a usar como *clock*. (**`KEY[0]`**). Defina novamente os sinais de controlo (agora para a segunda instrução) e prima a tecla de *clock*. Proceda deste modo até chegar à última instrução.

Usando o módulo de visualização, tome nota, para cada uma das instruções, dos valores presentes à saída do módulo de atualização do PC, do valor à saída da ALU e dos valores armazenados no banco de registos (na conclusão da instrução). Verifique a correção dos valores obtidos, comparando-os com os que calculou no ponto 5.

Tabela 3. Instruções, sinais de controlo e resultados para o programa de teste dos módulos do *datapath single-cycle* já implementados.

Program Counter	Instruction	Instruction code	RegWrite SW[7]	RegDst SW[6]	ALUSrc SW[5]	ALUOp SW[4..3]	ALU_Out value	Destination register value
0x00000000	addi $2,$0,0x1A	0x2002001A	1	0	1	00	0x0000001A	$2 = 0x0000001A
0x00000004	addi $3,$0,-7							$3 =
	add $4,$2,$3							$4 =
	sub $5,$2,$3							$5 =
	and $6,$2,$3							$6 =
	or $7,$2,$3							$7 =
	nor $8,$2,$3							$8 =
	xor $9,$2,$3							$9 =
	slt $10,$2,$3							$10=
	slti $11,$7,-2							$11=
	slti $12,$9,-25							$12=
	nop							

PDF gerado em 05/11/2018