Processor Architecture Project

Roger Garcia and Andreu Gironés

January 24, 2022

1 Instructions

1.1 Load

size	imm	rs1	func3	rd	opcode
Word	imm[11:0]	rs1	010	rd	0000011
Byte	imm[11:0]	rs1	000	rd	0000011

1.1.1 Fetch

Before the fetch step ends, the instruction is decoded in control. Where the *opcode*, *func3*, and *func7* are analyzed, and the following control bits are defined.

Size	RegWrite	MemWrite	Load	Branch	Jump	Byte	ALUSrc	MemtoReg	ALUOp
Word	1	0	1	0	0	0	1	1	00
Byte	1	0	1	0	0	1	1	1	00

1.1.2 Decode

The instruction is decoded in the datapath module.
The immediate is extracted in the *signextD* module.
The register source 1 is wired from the instruction register directly to the register file module. The value read it from the register file would be wired into the ALU through *areg*.
The destination register is wired to to the next register (areg).

1.1.3 ALU

In this step, the ALU sums the immediate and the register source, and determines the target address. After it passes the result to Memory stage through *alureg*.

1.1.4 Memory

The load tries to find the targeted data in the cache, if it misses, the pipeline is stalled during the 10 cycles.

1.1.5 Write back

Finally, the data read from memory is stored into the destination register in the register file. If it is a load byte operation the byte is extended before storing it.

1.2 Store

size	imm	rs2	rs1	func3	imm	opcode
Word	imm[11:5]	rs2	rs1	010	imm[4:0]	0100011
Byte	imm[11:5]	rs2	rs1	000	imm[4:0]	0100011

1.2.1 Fetch

Before the fetch step ends, the instruction is decoded in control. Where the *opcode*, *func3*, and *func7* are analyzed, and the following control bits are defined.

Size	RegWrite	MemWrite	Load	Branch	Jump	Byte	ALUSrc	MemtoReg	ALUOp
Word	0	1	0	0	0	0	1	x	00
Byte	0	1	0	0	0	1	1	x	00

1.2.2 Decode

The instruction is decoded in the datapath module.
The immediate is extracted in the *signextD* module.
The register source 1, the one that defines the target address, is wired from the instruction register directly to the register file module. The value read it from the register file would be wired into the ALU through *areg*.
The register source 2 indicates in which register is stored the data that must be written in memory. It is wired to the register file, and the read value is wired to the next register (areg). However, if it is a store byte operation the last byte from the read word is truncated, then extended, before wiring it to *areg*.

1.2.3 ALU

In this step, the ALU sums the immediate and the register source, and determines the target address.

1.2.4 Memory

The store tries to find the targeted address in the cache, if it misses, the pipeline is stalled during the 10 cycles.

1.2.5 Write back

Nothing is done here.

1.3 Branch/Jump

Operation	imm	rs2	rs1	funct3	imm	opcode
BEQ	imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011

Operation	imm	rd	opcode
JALR	imm[20|10:1|11|19:12]	rd	1100011

1.3.1 Fetch

Before the fetch step ends, the instruction is decoded in control. Where the *opcode*, *func3*, and *func7* are analyzed, and the following control bits are defined.

Operation	RegWrite	MemWrite	Load	Branch	Jump	Byte	ALUSrc	MemtoReg	ALUOp
BEQ	0	0	0	1	0	0	0	0	00
JALR	0	0	0	0	1	0	1	0	00

1.3.2 Decode

The instruction is decoded in the datapath module.
The immediate is extracted in the *signextD* module.
In case of a branch, registers source 1 and 2 are read from the register file and wired to the next step (areg).

1.3.3 ALU

In this step, the ALU substracts the two values read from the register file and if it results in 0 the branch is taken, otherwise not.

1.3.4 Memory

Nothing is done here.

1.3.5 Write back

Nothing is done here.

1.4 Arithmetic

funct7	rs1	func3	rd	imm	opcode
funct7	rs2	rs1	000	rd	0110011

1.4.1 Fetch

Before the fetch step ends, the instruction is decoded in control. Where the *opcode*, *func3*, and *func7* are analyzed, and the following control bits are defined.

RegWrite	MemWrite	Load	Branch	Jump	Byte	ALUSrc	MemtoReg	ALUOp
1	0	x	0	0	0	0	0	10

Depending of the field *funct7* the ALU would perform the correspondent operation (aludec):

ADD	SUB	MUL
0000000	0100000	0000001

1.4.2 Decode

The instruction is decoded in the datapath module.
Registers source 1 and 2 are read from the register file and wired to the next step (areg).
Also, it is read the destination register, that would be needed in the *Write back* step.

1.4.3 ALU

In this step, the ALU performs the operation to the two values read from the register file.
In case of a multiplication, the pipeline is stalled during the 5 cycles that the operation lasts.

1.4.4 Memory

Nothing is done here.

1.4.5 Write back

Finally, the result calculated in the ALU is stored into the destination register in the register file.

2 Bypasses

The foolowing bypasses are done when the forwarding unit detects a RaW dependency between instructions:

From	To
Memory	ALU
Write back	ALU