Politecnico di Torino

Microelectronic Systems

DLX Microprocessor: Design & Development
Final Project Report

Master degree in Electronics Engineering
Master degree in Computer Engineering

Referents: Prof. Mariagrazia Graziano, Giovanna Turvani

Authors: Group 52
name1, name2

October 18, 2023

Contents

CHAPTER 1

Introduction

1.1 Specifications

CHAPTER 2

Functional schema

2.1 Datapath

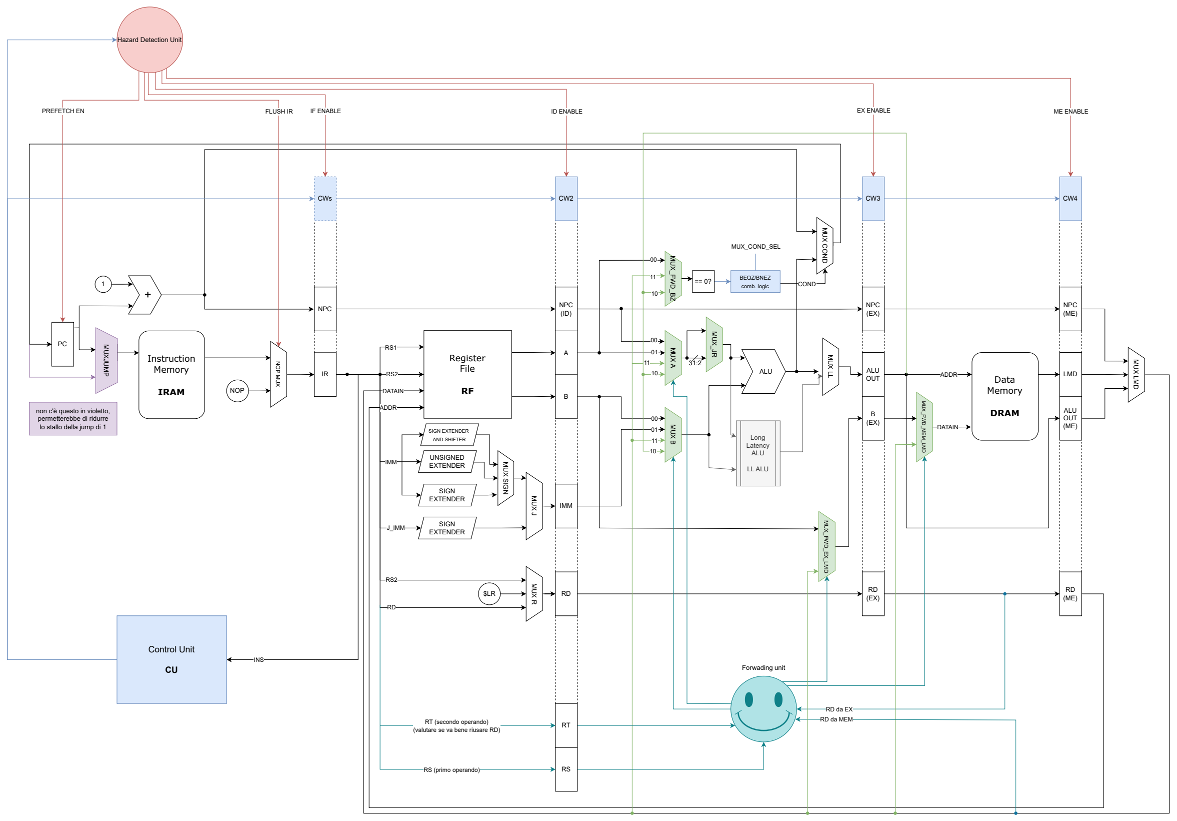

Figure 2.1: sus

2.2 Functional blocks

2.2.1 Control unit

The Control Unit is in charge of managing the datapath throughout its stages. Given an instruction from IR, it generates the corresponding control word that manages the various registers, MUXes and other control signals in the datapath. It also receives a block of *STALL* signals from the HDU (Hazard Detection Unit) that instructs the CU on when to stall the pipeline in case of data hazards.

It is implemented as an hybrid between an hardwired and a microcoded CU, with two processes that simply associate every instruction with a given control word and every ALU opcode with the corresponding operation, plus a process that manages the transition between every stage taking into consideration the *STALL* signals from the HDU.

2.2.2 Register file

The register file included in the DLX contains 32 registers, each of word size equal to 32 bits. Following the MIPS ISA, register *R0* has its value hardcoded to 0 and register *R31* is the *Link Register* used for returning from subroutines.

It has been chosen to be a Double-read/Single-write type, as in: on every clock cycle, the register file can output the values of two registers at the time by providing addresses on ports *ADD_RD1* and *ADD_RD2* and requesting reads on ports *RD1* and *RD2*; a write can be made by setting an address on *ADD_WR* and a word on port *DATAIN*. Requested registers output their value on ports *OUT1* and *OUT2*.

2.2.3 ALU

The ALU of the DLX operates on two inputs: *DATA1* and *DATA2*. It can obtain them from a multitude of sources, namely:

- from the register file, through 2 stage registers *A* and *B*;
- from the program counter, necessary for address computations in branching and jump instructions;
- from the instructions themselves, immediate values via the instruction register after the appropriate transformations;
- from the ALU itself, and in general from other stages' outputs through MUXes controlled by the forwarding unit.

The function to be computed on the operands is selected by a third input *FUNC* that receives a code representing it and the result is sent to the output *OUTALU*. A list of currently implemented functions follows. The implementation of each operation is behavioural unless specified otherwise.

Addition

$ALUOUT = DATA1 + DATA2$

A P4 adder is present inside the ALU to perform additions. Further details on its implementation together with the VHDL description are included in lab 2's zip file.

Subtraction

$ALUOUT = DATA1 - DATA2$

The P4 adder is also used for subtractions, by means of negating one of the inputs and setting the carry-in input of the adder to 1.

Multiplication

$ALUOUT = DATA1 \cdot DATA2$

Multiplication is executed on the operands fully but the result is still word size: the most significant half of the computed value is discarded.

Initially, the multiplication operation was to be delegated to a long latency ALU, with this implementation just being a placeholder. However the group decided to focus on more difficult to implement features, leaving no time for description and testing of an LL ALU.

AND

$ALUOUT = DATA1 \wedge DATA2$

OR

$ALUOUT = DATA1 \vee DATA2$

XOR

$ALUOUT = DATA1 \oplus DATA2$

Logical Shift Left

$ALUOUT = DATA1 \ll DATA2$

Logical Shift Right

$ALUOUT = DATA1 \gg DATA2$

Set equal

$if(DATA1 == DATA2)\ then\ ALUOUT = 1\ else\ 0$

Set not equal

$if(DATA1 \neq DATA2)\ then\ ALUOUT = 1\ else\ 0$

Set greater than or Equal (signed and unsigned)

$if(DATA1 \geq DATA2)\ then\ ALUOUT = 1\ else\ 0$

Set greater than (signed and unsigned)

$if(DATA1 > DATA2)\ then\ ALUOUT = 1\ else\ 0$

Set less than or Equal (signed and unsigned)

$if(DATA1 \leq DATA2)\ then\ ALUOUT = 1\ else\ 0$

Set less than (signed and unsigned)

$if(DATA1 < DATA2)\ then\ ALUOUT = 1\ else\ 0$

2.2.4 Hazard detection unit

The Hazard Detection Unit takes care of detecting data and control hazards in the pipeline and successively dispatching stage enable (or STALL, as written in the CU subsection) signals to the CU for indication on where and for how many stages to stall for. It does so by progressively checking for hazardous conditions in a process starting from IRAM and DRAM readiness, then for branches (assuming not taken), jumps and loads from memory giving precedence, in case of instructions of the same type, to instructions currently found in the execute stage. If no hazardous condition is found, a STALL_CLEAR signal is sent out.

2.2.5 Forwarding unit

The Forwarding Unit takes as input a subset of the control signals sent by the CU and of the register selection codes (not the values inside the registers themselves). Given those, it detects favourable conditions for source and destination registers between stages of the pipeline and if they match it forwards operands where they are needed by changing selection signals for the appropriate MUXes and skipping the write-back stage.