b

School of Sciences and Engineering

Milestone 2

By:

Dalia Elnagar ID: 900191234

Kirolos Mikhail ID: 900191250

Kareem A. Mohammed Talaat ID: 900192903

**CSCE 3301-02 Computer Architecture**

Fall 2021

Milestone Description:

GitHub link:

Schematic:

**Modules Description**

**PC**

Inputs and outputs:

rst: a reset button to reset the value of the program counter to 0

pc\_en: an enabler which enables the program count register to get incremented or decremented or obtain its current value in case we do not want to load the next instruction.

clk: the system’s global clock.

PC\_input: the program counter input which is the value we desire the program counter to be next (for example, for a normal instruction (no jumps or anything) should be the old PC value + 4).

Inst\_read\_address: the output of the program counter which is the instruction address to be used to fetch the instruction 32 bits.

Purpose:

The program counter component is used to store the program counter values, and depending on the current program counter value, the instruction address is produced as an output.

**Instruction Memory**

Inputs and outputs:

Inst\_read\_address[7:2]: the 6 bits input which the component uses to fetch the instruction from the memory.

inst: the 32 bit instruction which is the fetched instruction from the memory produced as an output.

Purpose:

The instruction memory holds all the instructions in the memory, and depending on the input address, an instruction is fetched and produced as an output

**Control Unit**

Inputs and outputs:

rst: a reset button to reset all the values to 0

inst: the 32 bit instruction to be used an input.

pc\_en: an enabler which enables the program count register to get incremented or decremented or obtain its current value in case we do not want to load the next instruction produced as an output.

Branch: a boolen enabled when the instruction being currently executed is a branching instruction.

Jump: a boolen which is enabled when the instruction being currently executed is a jumping instruction (jal, jalr).

Mem\_read: a boolen which is enabled when the instruction being currently executed requires a memory read (lw for example)

Mem\_to\_reg: a boolen which is enabled when the instruction being currently executed requires saving a value from the memory to the register file (add for example).

Mem\_write: a boolen which is enabled when the instruction being executed requires writing into the memory.

ALU\_Src: a select which is used to select the input value into the ALU.

Reg\_write: a boolen which is enabled when writing to the register file is requires.

Signed\_inst: a boolen which is enabled when the instruction being executed is a signed instruction.

AU\_inst\_sel: 2 bit selection used to select the type of the loading or storing instruction (LH, LB, LBU, SW, etc..)

ALUOp: 2 bit selection used to select the type of operation the ALU is going to be executed.

RF\_MUX\_sel: a 2 bit selection used to select the value to be used as the 32 bits value to be written into the register file (write\_data) based on the instruction at hand.

Purpose:

The control unit is using the instruction bits to identify the instruction at hand, and therefore produce controlling outputs (like selects, branch, jump, etc) which regulate the other components depending on the instruction at hand.

**Register File**

Inputs and outputs:

rst: a reset button to reset the values of the registers to 0

clk: the system’s global clock.

read\_reg1: the read register 1 address from the instruction (5 bits from bit 15 to bit 19 in the instruction)

read\_reg2: the read register 2 address from the instruction (5 bits from bit 20 to bit 24 in the instruction)

write\_reg: the destination register address from the instruction (5 bits from bit 7 to bit 11 in the instruction)

write\_data: the 32 bits register value to write into register file in the destination register address.

reg\_write: the 1 bit select which enables/disables writing in the register file, specifically in the destination register.

read\_data1: the 32 bits register value produced as an output from the register file which represents the first source register in the instruction.

read\_data2: the 32 bits register value produced as an output from the register file which represents the second source register in the instruction.

Purpose:

The register file is used to store the registers values, and there can be multiple operations executed on it:

We can read registers values based on the input address.

We can modify the register value (the write\_address is used to identify that register), and write\_data is the value to be written into the register, and all of this is enabled by the reg\_write, so if it is 1 then we can change that register’s value, otherwise we cannot edit it.

**Immediate generator**

Inputs and outputs:

IR (inst): the 32 bit instruction as an input

Gen\_out (immediate): the immediate value produced depending on the instruction.

Purpose:

The immediate generator is used to generate the immediate value based on the instruction to be used later in other components depending on the instruction at hand, for example, in a beq instruction, the immediate generator is used to generate the immediate value from the instruction bits, then shift it to the left (multiply by 2) to be the offset to be added to the current program counter to represent the new program counter value.

Test Cases