BIRZEIT UNIVERSITY

Department of Electrical and Computer Engineering
Spring Semester, 2023/2024
Project No. 2
ENCS4370| Computer Architecture
Deadline: January 22, 2024 at 23:59

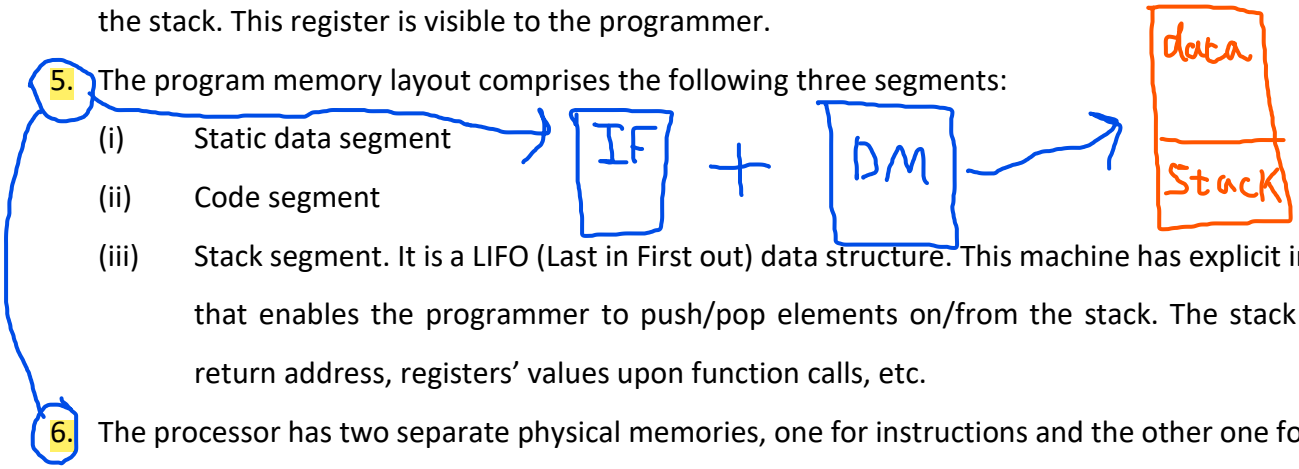
---

1. Objectives:

To design and verify a simple pipelined RISC processor in Verilog

2. Processor Specifications

1. The instruction size and the words size is 32 bits
2. 16 32-bit general-purpose registers: from R0 to R15.
3. 32-bit special purpose register for the program counter (PC)
4. 32-bit special purpose register for the stack pointer (SP), which points to the topmost empty element of the stack. This register is visible to the programmer.
5. The program memory layout comprises the following three segments:
   (i) Static data segment
   (ii) Code segment
   (iii) Stack segment. It is a LIFO (Last in First out) data structure. This machine has explicit instructions that enables the programmer to push/pop elements on/from the stack. The stack stores the return address, registers' values upon function calls, etc.



6. The processor has two separate physical memories, one for instructions and the other one for data. The data memory stores both the static data segment and the stack segment.
7. Four instruction types (R-type, I-type, J-type, and S-type).
8. Separate data and instructions memories
9. **Word-addressable memory**
10. You need to generate the required signals from the ALU to calculate the condition branch outcome (taken/ not taken). These signals might include zero, carry, overflow, etc.

3. Instruction Types and Formats

As mentioned above, this ISA has four instruction formats, namely, R-type, I-type, J-type, and S-type. These four types have a common **6-bit opcode** field, which determines the specific operation of the instruction

R-Type (Register Type)

Opcode^{6}	Rd^{4}	Rs1^{4}	Rs2^{4}	Unused^{14}

- **4-bit Rd:** destination register
- **4-bit Rs1:** first source register
- **4-bit Rs2:** second source register
- **14-bit unused**

I-Type (Immediate Type)

Opcode^{6}	Rd^{4}	Rs1^{4}	Immediate^{16}	Mode^{2}

- **4-bit Rd:** destination register
- **4-bit Rs1:** first source register
- **16-bit immediate:** unsigned for logic instructions, and signed otherwise.
- **2-bit mode:** this is used with load/store instructions only, such that
  00: no increment/decrement of the base register
  01: post increment the base register
  10-11: unused

  For example:
  `lw Rd, imm (Rs1)` `# Reg[Rd] = Memory [Reg[Rs1] + sign_extend(imm)]`

Mode Value	Action
00	**`lw Rd, imm (Rs1)`** `# No inc/dec of the base register`
01	**`LW.POI Rd, imm (Rs1)`** `# load word post-increment` After the address is sent to the memory, the base register is incremented **`Reg[Rs1] = Reg[Rs1] + 1`**
10	**Unused**
11	**Unused**

J-Type (Jump Type)

This type includes the following instruction formats. The opcode is used to distinguish each instruction

```
jmp L  # Unconditional jump to the target L.
call F # Call the function F.  F is a label.
       # The return address is pushed on the stack
```

The target address is calculated by concatenating the most significant 6-bit of the current PC with the 26-bit offset

Opcode[6]	Jump Offset[26]

```
ret    # return from a function.
       # the next PC will be the top element of the stack
```

Opcode[6]	Unused[26]

S-Type (Stack)

- **4-bit Rd**

Opcode[6]	Rd[4]	Unused[22]

```
push Rd # Push the value of Rd on the top of the stack

pop Rd # Pop the stack and store the topmost element in Rd
```

4. Instructions' Encoding

For simplicity, you are required to implement a subset only of this processor's ISA. The table below shows the different instructions you are required to implement. It shows their type, the opcode value, and their meaning in RTN (Register Transfer Notation). Although the instruction set is reduced, it is still rich enough to write useful programs.

No.	Instr	Meaning	Opcode Value
R-Type Instructions			
1	AND	Reg(Rd) = Reg(Rs1) & Reg(Rs2)	000000
2	ADD	Reg(Rd) = Reg(Rs1) + Reg(Rs2)	000001
3	SUB	Reg(Rd) = Reg(Rs1) - Reg(Rs2)	000010
I-Type Instructions			
4	ANDI	Reg(Rd) = Reg(Rs1) & Imm^{16}	000011
5	ADDI	Reg(Rd) = Reg(Rs1) + Imm^{16}	000100
6	LW	Reg(Rd) = Mem(Reg(Rs1) + Imm^{16})	000101
7	LW.POI	Reg(Rd) = Mem(Reg(Rs1) + Imm^{16}) Reg[Rs1] = Reg[Rs1] + 1	000110
8	SW	Mem(Reg(Rs1) + Imm^{16}) = Reg(Rd)	000111
9	BGT	if (Reg(Rd) > Reg(Rs1)) Next PC = PC + sign_extended (Imm^{16}) else PC = PC + 1	001000
10	BLT	if (Reg(Rd) < Reg(Rs1)) Next PC = PC + sign_extended (Imm^{16}) else PC = PC + 1	001001
11	BEQ	if (Reg(Rd) == Reg(Rs1)) Next PC = PC + sign_extended (Imm^{16}) else PC = PC + 1	001010
12	BNE	if (Reg(Rd) != Reg(Rs1)) Next PC = PC + sign_extended (Imm^{16}) else PC = PC + 1	001011
J-Type Instructions			
13	JMP	Next PC = {PC[31:26], $Immediate^{26}$ }	001100
14	CALL	Next PC = {PC[31:26], $Immediate^{26}$ } PC + 1 is pushed on the stack	001101
15	RET	Next PC = top of the stack	001110
S-Type Instructions			
16	PUSH	Rd is pushed on the top of the stack	001111
17	POP	The top element of the stack is popped, and it is stored in the Rd register	010000

5. RTL Design

You are required to design a **5-stage pipelined** processor (fetch, decode, ALU, memory access, and write back).

The design should include a datapath and control path that support all of the aforementioned instructions.

6. Verification

To verify the RTL design, write a testbench around it. Moreover, you need to write multiple code sequences in the given ISA and show how the processor you designed executes these code sequences.

7. Project Report

The report document must contain sections highlighting the following:

1 – Design and Implementation

1. Detailed description of the datapath, its components, and the assembly of these components.
2. A complete description of the control signals, description, truth table, and Boolean equations
3. The implementation details, and the design choices you made with justification
4. Datapath and control path block diagrams.
5. A list of sources for any parts of your design that are not entirely yours (if any).
6. Carry out the design and implementation with the following aspects in mind:
   - Correctness of the individual components
   - Correctness of the overall design when connecting these components together
   - Completeness: all instructions were implemented properly.

2 – Simulation and Testing

1. Carry out the simulation of the processor developed
2. Describe the test programs that you used to test your design with enough comments describing the program, its inputs, and its expected output. List all the instructions that were tested and work correctly. List all the instructions that do not run properly.
3. Also, provide snapshots of the simulator window with your test program loaded and showing the simulation output results.

3 – Teamwork

1. Work in groups of up to three students.
2. Group members are required to coordinate the work equally among themselves so that everyone is involved in all the following activities:
   - Design and implementation
   - Simulation and testing
   - Report writing
   - Project demonstration and discussion
3. Clearly show the work done by each group member.

8. Submission Guidelines

Attach one zip file containing all the design circuits, the programs source code and binary instruction files that you have used to test your design, their test data, as well as the report document.

9. Grading Criteria

Note: each of the following items is a prerequisite to the next one, e.g., we cannot consider the RTL code if there is no design, and we cannot consider the verification part if there is no RTL code.

Item	**Weight**
Control signals generation (truth tables and Boolean equations)	10
Processor Microarchitecture Design (complete datapath and control path) that supports all instructions	20
Complete **modular** RTL design of the above microarchitecture that supports all instructions	20
Verification environment (testbench) around the RTL design such that you can write code sequences in the ISA, store them in the instruction memory, execute them and show results using waveform diagrams.	20
Code organization and documentation	5
Detailed report that includes control signals truth tables, Boolean equations, detailed and clear microarchitecture design schematics, test cases, simulation screenshots, etc.	15
Discussion	10
Total	**100**