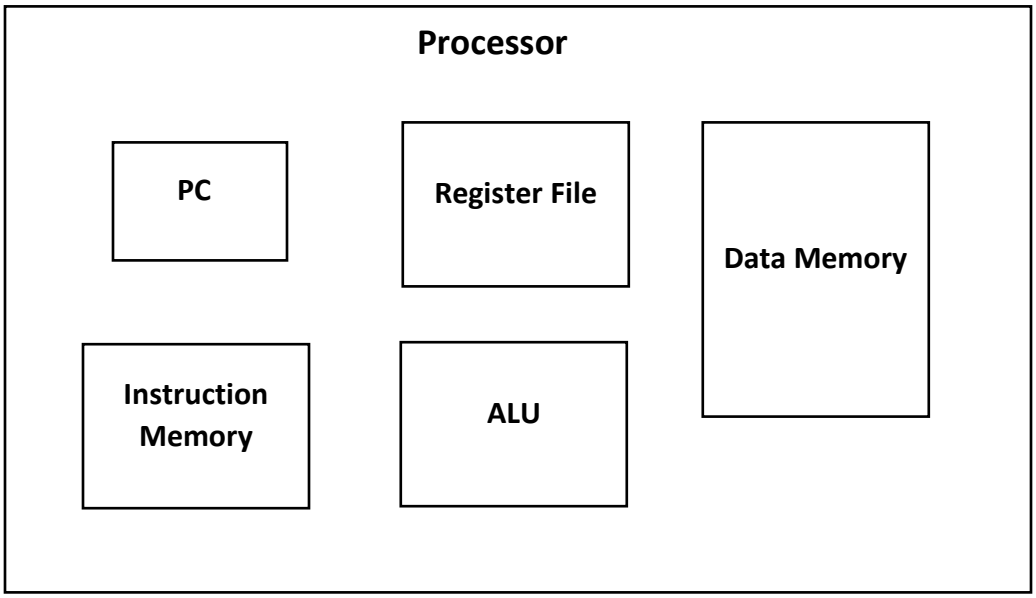
UNC CHARLOTTE

Fall 2021-ECGR-3183-Computer Organization
Dr. Fareena Saqib

Project

In this project you will be designing and simulating a 32-bit processor using VHDL. The Instruction Set and the Architecture design parameters are provided.

Architecture Design:



The main components in the Architecture are: PC, Register File , Instruction Memory , Data Memory and the ALU. Additional Combinational Logic blocks such MUX , Adders and gates can be added to the design if required for the instruction set provided. The Register file consists of 32 registers which are 32-bit wide.

Instruction Format:

Name		**Fields**					**Comments**
Filed size	**6 to 11 bits**	**5 to 10 bits**	**5 or 4 bits**	**2 bits**	**5 bits**	**5 bits**	**32 bit long**
R- Format	opcode	Rm	shamt		Rn	Rd	Arithmetic instruction format
I-Format	opcode	Immediate			Rn	Rd	Immediate format
D-Format	opcode	address		Op2	Rn	Rt	Data transfer format
B-Format	opcode	address					Unconditional branch
CB	opcode	address				Rt	Conditional branch

Instruction Set:

Name	**Instruction**	**Format**	**Operation**
Add	ADD	R	R[Rd]= R[Rn]+R[Rm]
Subtract	SUB	R	R[Rd]= R[Rn]-R[Rm]
Logical Left Shift	LSL	R	R[Rd]= R[Rn]<< shamt
Logical Right Shift	LSR	R	R[Rd]= R[Rn]>> shamt
Add Immediate	ADDI	I	R[Rd]= R[Rn]+Imm
Subtract Immediate	SUBI	I	R[Rd]= R[Rn]- Imm
And	AND	R	R[Rd]= R[Rn]&R[Rm]
Or	OR	R	R[Rd]= R[Rn]+R[Rm]
Load	LDUR	D	R[Rt]= M[R[Rn]+DTAddr]
Store	STUR	D	M[R[Rn]+DTAddr]= R[Rt]
Move	MOV	D	R[Rd]= R[Rn]
Unconditional Branch	B	B	PC=PC+ BranchAddr
Branch to Register	BR	B	PC=R[Rt]
Compare	CMP	CB	Flags= R[Rn]- R[Rm]
Compare and Branch on 0	CBZ	CB	if (R[Rt]==0 PC=PC+CondBranchAddr

Instruction Encoding:

Instruction	Opcode	Opcode size	11- bit opcode value
ADD	10001011000	11	1112
SUB	11001011000	11	1624
LSL	11010011011	11	1691
LSR	11010011010	11	1690
ADDI	1001000100	10	1160
SUBI	1101000100	10	1672
AND	10001010000	11	1104
OR	10101010000	11	1360
LDUR	11111000010	11	1986
STUR	11111000000	11	1984
MOV	110100101	9	1684
B	000101	6	160
BR	11010110000	11	1712
CMP	10110101	8	1448
CBZ	10110100	8	1440

Example:

LDUR X9, [X10,#240]
ADD X9,X21,X9

Instruction	32 – bit Binary equivalent				
LDUR	11111000010	011110000	00	01010	01001
ADD	10001011000	01001	000000	10101	01001

You can refer to the LEGv8 instructions and encoding for more details.

Part 1:

Single-Cycle Implementation:

- First you will design the datapath for the given instruction set with the control unit. Tabulate the Control signals and the ALU Control for the given Instruction set.
- Write the VHDL code for the whole design and the Testbench code to simulate the implemented design.
- Run sample VHDL code for simulation to show the implemented Single-Cycle Processor.

Part 2:

5-stage Pipelining Implementation:

- Design the datapath for the given instruction set with the control unit for a 5- stage Pipelining.
- Assume there is no hazard's in the designed 5-stage pipeline.
- Write the VHDL code for the whole design and the Testbench code to simulate the implemented design.
- Run a sample VHDL code for simulation to show the implemented 5-stage Pipelined Processor.

Report:

The following is provided as a general outline for your final report

1. Introduction
2. ISA (instruction set, instruction formats)
3. Architecture
    - Datapath
    - Controller
4. VHDL Description (do not include code, upload your entire project zipped up via Canvas, this should describe the model)
5. Simulation Results
6. Conclusion

Bonus Points:

1. Design a Floating-Point Processor to run the IEEE single-precision floating point numbers for add instruction. Write a VHDL code and simulate the design to show the implemented instruction.
2. For the designed 5-stage Pipelining in Part 2 , assume data hazard's and implement the Forwarding unit . Write a VHDL code and simulate the design to show the implemented module.

Test Programs:

1. Use the following ASM test program for simulation results.

    a=X9, b=X10, c=X11, d=X12 f=X13 , g=X14, h=X15, j= X16 , i=X19, n= X1

```
MOV X19,XZR
CMP X19, X1
```

```
        B.GE EXIT
        ADD X13, X9,X10
        SUB X14, X10, X11
        LSL X15, X9,#2
        LSR X16, X10, #4
        SUB X21, X15, X16
        CBZ X21, ELSE
        AND X12, X13, X14
        B EXIT2
ELSE :  OR X12, X13, X14
EXIT2   ADDI X19, X19, #1
EXIT
```

2. Use the following C-Code for simulation results. First convert the C-code to assembly and write the VHDL Testbench for it to get the simulation results.

```
int fib(int n)
{
if (n==0)
        return 0;
else if (n == 1)
        return 1;
else
        return fib(n−1) + fib(n−2);
}
```