LAB # 9
Arithmetic and Branch Instruction Assembly Code

Objective:

- Get Familiar with Arithmetic and Branch Instruction Assembly Code

System Module (Hardware/Software)

- Venus Simulator

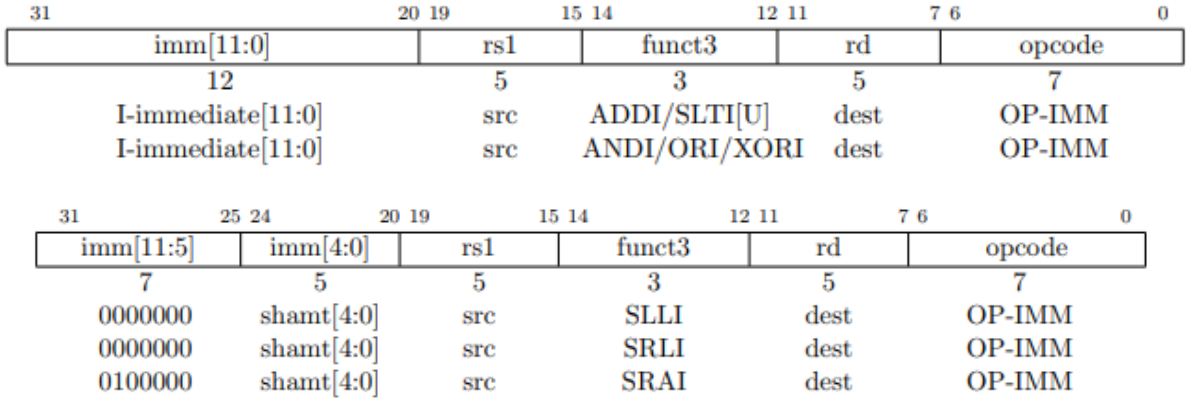
• Theoretical Background:

Introduction:

Arithmetic Instructions:

Most integer computational instructions operate on XLEN bits of values held in the integer register file. Integer computational instructions are either encoded as register-immediate operations using the I-type format or as register-register operations using the R-type format. The destination is register rd for both register-immediate and register-register instructions. No integer computational instructions cause arithmetic exceptions.

The instruction format for register-immediate instruction is shown below:



31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
I-immediate[11:0]	src	ADDI/SLTI[U]	dest	OP-IMM
I-immediate[11:0]	src	ANDI/ORI/XORI	dest	OP-IMM

31 25	24 20	19 15	14 12	11 7	6 0
imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	shamt[4:0]	src	SLLI	dest	OP-IMM
0000000	shamt[4:0]	src	SRLI	dest	OP-IMM
0100000	shamt[4:0]	src	SRAI	dest	OP-IMM

Types of Register-Immediate instruction supported in RISC-V are:

Types	Symbol
Add Immediate	ADDI
Set Less Than Immediate (signed)	SLTI
Set Less Than Immediate (unsigned)	SLTIU
And Immediate	ANDI
Or Immediate	ORI
Xor Immediate	XORI

Shift Left Logical Immediate	SLLI
Shift Right Logical Immediate	SRLI
Shift Right Arithmetic Immediate	SRAI

ADDI adds the sign-extended 12-bit immediate to register rs1, SLTI (set less than immediate) places the value 1 in register rd if register rs1 is less than the sign-extended immediate when both are treated as signed numbers, else 0 is written to rd. SLTIU is similar but compares the values as unsigned numbers (i.e., the immediate is first sign-extended to XLEN bits then treated as an unsigned number). ANDI, ORI, XORI are logical operations that perform bitwise AND, OR, and XOR on register rs1 and the sign-extended 12-bit immediate and place the result in rd. The operand to be shifted is in rs1, and the shift amount is encoded in the lower 5 bits of the I-immediate field. SLLI is a logical left shift (zeros are shifted into the lower bits); SRLI is a logical right shift (zeros are shifted into the upper bits); and SRAI is an arithmetic right shift (the original sign bit is copied into the vacated upper bits).

The assembly representation for register-immediate instructions are:

addi rd, rs1, imm
slti rd, rs1, imm
ori rd, rs1, imm

RV32I defines several arithmetic R-type operations. All operations read the rs1 and rs2 registers as source operands and write the result into register rd. The funct7 and funct3 fields select the type of operation.

The instruction format for register-register instruction is shown below:

31 25	24 20	19 15	14 12	11 7	6 0
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	src2	src1	ADD/SLT/SLTU	dest	OP
0000000	src2	src1	AND/OR/XOR	dest	OP
0000000	src2	src1	SLL/SRL	dest	OP
0100000	src2	src1	SUB/SRA	dest	OP

Types of Register-Register instruction supported in RISC-V are:

Types	**Symbol**
Addition	ADD
Subtraction	SUB
Set Less Than (signed)	SLT
Set Less Than (unsigned)	SLTU
Logical And	AND

Logical Or	OR
Logical Xor	XOR
Shift Left Logical	SLL
Shift Right Logical	SRL
Shift Right Arithmetic	SRA

ADD and SUB perform addition and subtraction respectively. Overflows are ignored and the low XLEN bits of results are written to the destination. SLT and SLTU perform signed and unsigned compares respectively, writing 1 to rd if rs1 < rs2, 0 otherwise. AND, OR, and XOR perform bitwise logical operations. SLL, SRL, and SRA perform logical left, logical right, and arithmetic right shifts on the value in register rs1 by the shift amount held in the lower 5 bits of register rs2.

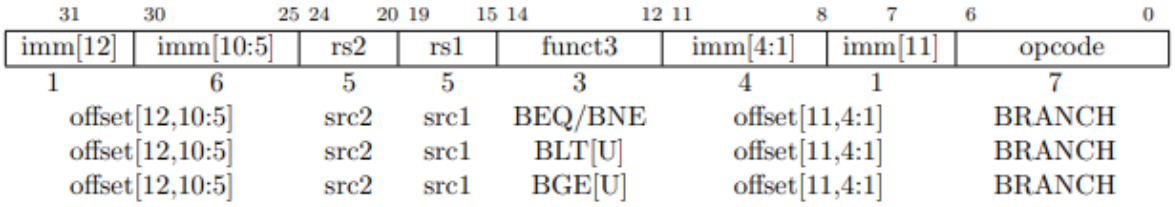
The assembly representation for register-register instructions are:

add rd, rs1, rs2
and rd, rs1, rs2
sub, rd, rs1, rs2

Branch Instructions:

All branch instructions use the B-type instruction format. The 12-bit B-immediate encodes signed offsets in multiples of 2, and is added to the current pc to give the target address. The conditional branch range is ±4 KiB.

The instruction format for branch instruction is shown below:



Types of Branch instruction supported in RISC-V are:

Types	**Symbol**
Branch if Equal	BEQ
Branch if Not Equal	BNE
Branch Less Than (signed)	BLT
Branch Less Than (unsigned)	BLTU
Branch Greater Than (signed)	BGE
Branch Greater Than (unsigned)	BGEU

Branch instructions compare two registers. BEQ and BNE take the branch if registers rs1 and rs2 are equal or unequal respectively. BLT and BLTU take the branch if rs1 is less than rs2, using signed and unsigned comparison respectively. BGE and BGEU take the branch if rs1 is greater than or equal to rs2, using signed and unsigned comparison respectively

Lab Exercise 1:
Run the below assembly code on Venus Simulator

```
addi x8, x0, 0x5
ori  x9, x0, 0x7
add x10, x8, x9
or  x5,  x9, x10
slt x6, x10, x5
```

Lab Exercise 2:
Run the below assembly code on Venus Simulator

```
addi x5, x0, 0x0
addi x6, x0, 0x6
loop:
    addi x5, x5, 0x1
    bne x5, x6, loop
addi x7, x0, 0x10
```

Procedure:

Follow the same procedure to run Venus as we did in Lab#03.

In-Lab Tasks:

1. **Write down a simple assembly program to calculate the Fibonacci series till 144. Stimulate the code on Venus.**
2. **Write down a simple assembly program to calculate the sum of even numbers between the range of 1-50. Stimulate the code on Venus.**

Conclusion:

What have you learnt from this lab?

_____

_____

_____

_____

Learning Outcomes:

Upon successful completion of the lab, students will be able to:

LO1: Run the basic assembly code on Venus Simulator.