Take Home Assignment 1 – Addressing Modes

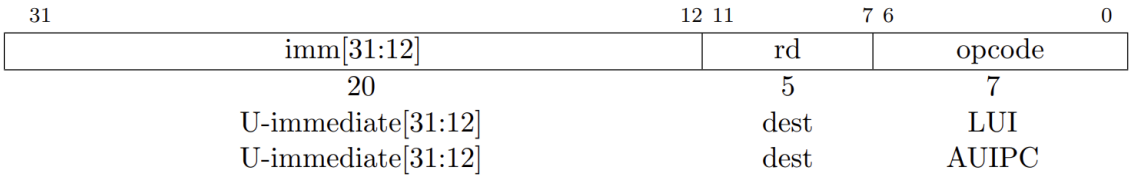
PC-relative Addressing-

A way to calculate memory addresses using PC as a reference point.

Uses – Branching and Jump instructions.

1) **auipc – calculate a 20-bit immediate value, sign-extend it to 64 bits, shift it left by 12 bits, and then add it to the current PC to form a 64-bit address.**

   Eg: auipc a0, 0x12345 # Add 0x12345 << 12 to the PC and store the result in a0.

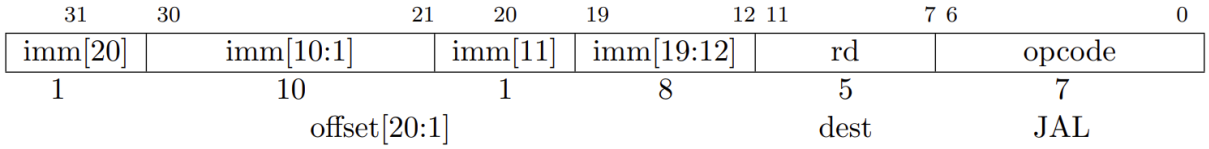


31 — 12	11 — 7	6 — 0
imm[31:12]	rd	opcode
20	5	7
U-immediate[31:12]	dest	LUI
U-immediate[31:12]	dest	AUIPC

2) **jal - perform an unconditional jump to a target address formed by adding a signed immediate**

value, which is shifted left by 1 bit, to the current PC.

Eg: jal ra, 12 #Jump to a address that is 12 bytes away and save the return address in ra.



31	30 21	20	19 12	11 7	6 0
imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	opcode
1	10	1	8	5	7
	offset[20:1]			dest	JAL

3)bne- Branches to a target address if the two specified registers are not equal.

Eg : bne x3, x4, 12 # Branch to 3 instructions ahead if x3 != x4

31	30 25	24 20	19 15	14 12	11 8	7	6 0
imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode
1	6	5	5	3	4	1	7
offset[12|10:5]		src2	src1	BEQ/BNE	offset[11|4:1]		BRANCH
offset[12|10:5]		src2	src1	BLT[U]	offset[11|4:1]		BRANCH
offset[12|10:5]		src2	src1	BGE[U]	offset[11|4:1]		BRANCH

4) blt -Branches to a target address if the value in the first specified register is less than the value in the second specified register.

Example :- blt x5, x6, 16 # Branch to 4 instructions ahead if x5 < x6

5) beg- Branches to a target address if the value in the first specified register is greater than or equal to the value in the second specified register.

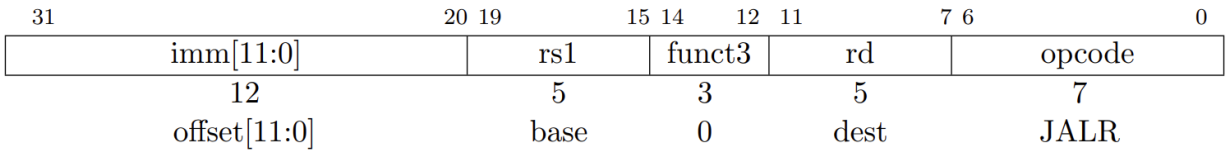
Example :- bge x7, x8, -8 # Branch to 2 instructions before if x7 >= x8

Register – Offset –

1) **jalr -The indirect jump instruction JALR uses the I-type encoding. The target address is obtained by adding the sign-extended 12-bit I-immediate to the register rs1, then setting the least-significant bit of the result to zero. The address of the instruction following the jump (pc+4) is written to register rd. Register x0 can be**

used as the destination if the result is not required.

EG : jalr x1, x2, 16 # Jump to the address (x2 + 16) and store the return address in x1



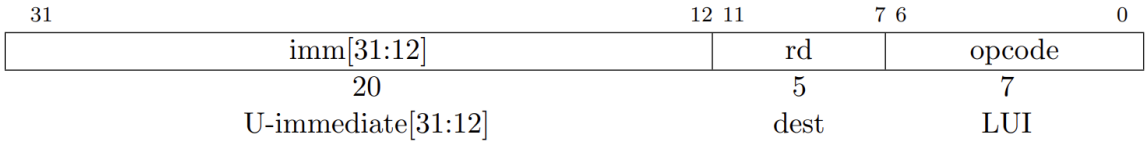
2) addi - add a signed immediate value to the value in a register and store the result in a destination register. This is a register-offset instruction because we add the immediate value to the register value.

Eg: addi x3, x4, 10 # Add immediate value 10 to the value in x4 and store the result in x3.

Absolute –

lui- load a 20-bit immediate value into the upper 20 bits of a register, effectively setting the lower 12 bits to zero.

Eg: lui x5, 0x12345 # Load immediate value 0x12345 into the upper 20 bits of x5



Summary:

1. Immediate addressing, where the operand is a constant within the instruction itself.

2. Register addressing, where the operand is a register.

3. Base or displacement addressing, where the operand is at the memory location whose address is the sum of a register and a constant in the instruction.

4. PC-relative addressing, where the branch address is the sum of the PC and a constant in the instruction.

Illustration of four RISC-V addressing modes. By RISCV pdf