RISC V Architecture

Mahesh Awati
Department of Electronics and Communication Engg.

RISC V ARCHITECTURE

UNIT 2 – Instructions: The Language of Computer

Mahesh Awati
Department of Electronics and Communication Engineering

Instructions – Language of Computer

Instructions for Making Decisions

RISC-V assembly language includes two decision-making instructions, similar to an ***if*** **statement with a *go-to*.**

PC-Relative Addressing: Use the **`immediate`** field as a 2's complement offset (signed offset) to PC Branches generally change the PC by a small amount Can specify $\pm 2^{11}$'unit' addresses from the PC

Why not use byte as a unit of offset from PC?
Instructions are of 32-bits (4-bytes) size and a offset of byte means branch into middle of instruction. Branching into middle of the instruction does not make any sense.

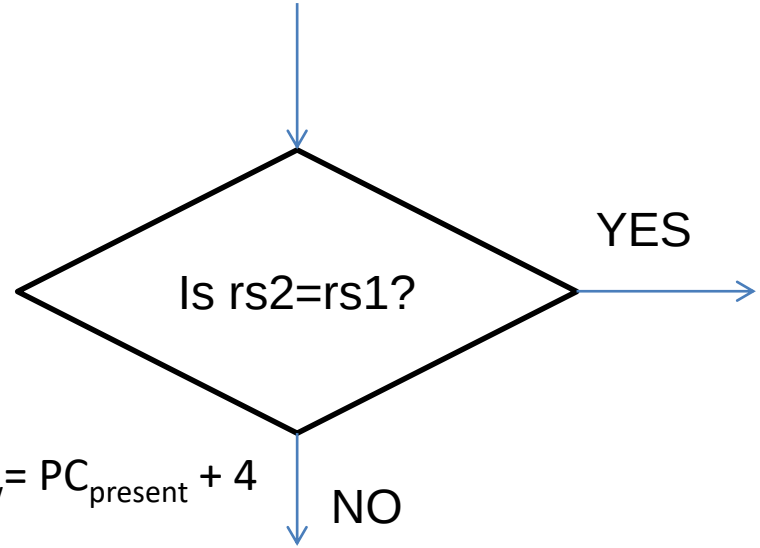
Instructions – Language of Computer

Instructions for Making Decisions

RISC-V assembly language includes two decision-making instructions, similar to an ***if* statement with a *go-to*.**

Conditional branches : An instruction that tests a value and that allows for a subsequent transfer of control to a new address in the program based on the outcome of the test.

beq rs1, rs2, L1



PC= Address of Instruction where branching need to be done. The Label L1 is replaced by the address by assembler.

$PC_{new} = PC_{present} + Signed\ offset$

where Signed offset = Immediate($\pm 2^{11}$) * 4

immediate is number of instructions to jump either forward (+) or backwards (–)

$PC_{new} = PC_{present} + 4$

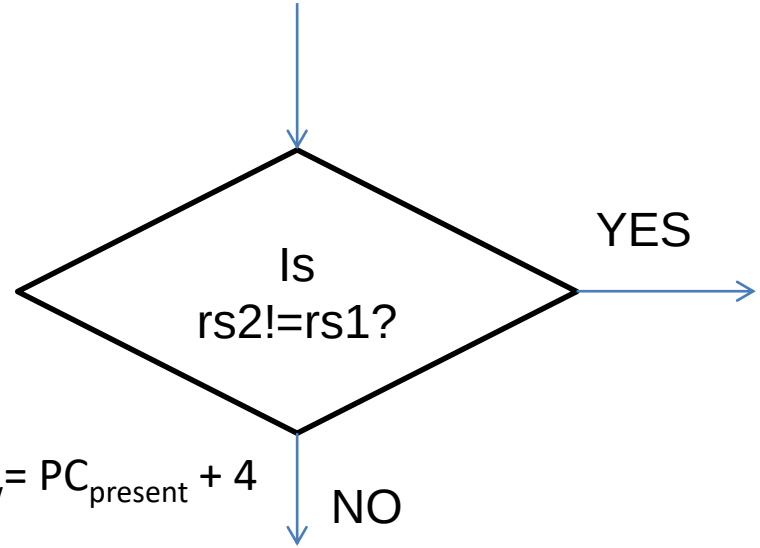
PC – NO change As a result, No change in Sequence of execution i.e., executes next Instruction after branch

Instructions for Making Decisions

RISC-V assembly language includes two decision-making instructions, similar to an ***if*** **statement with a *go-to*.**

Conditional branches : An instruction that tests a value and that allows for a subsequent transfer of control to a new address in the program based on the outcome of the test.

bne rs1, rs2, L1



PC= Address of Instruction where branching need to be done. The Label L1 is replaced by the address by assembler.

$PC_{new} = PC_{present}$ + Signed offset

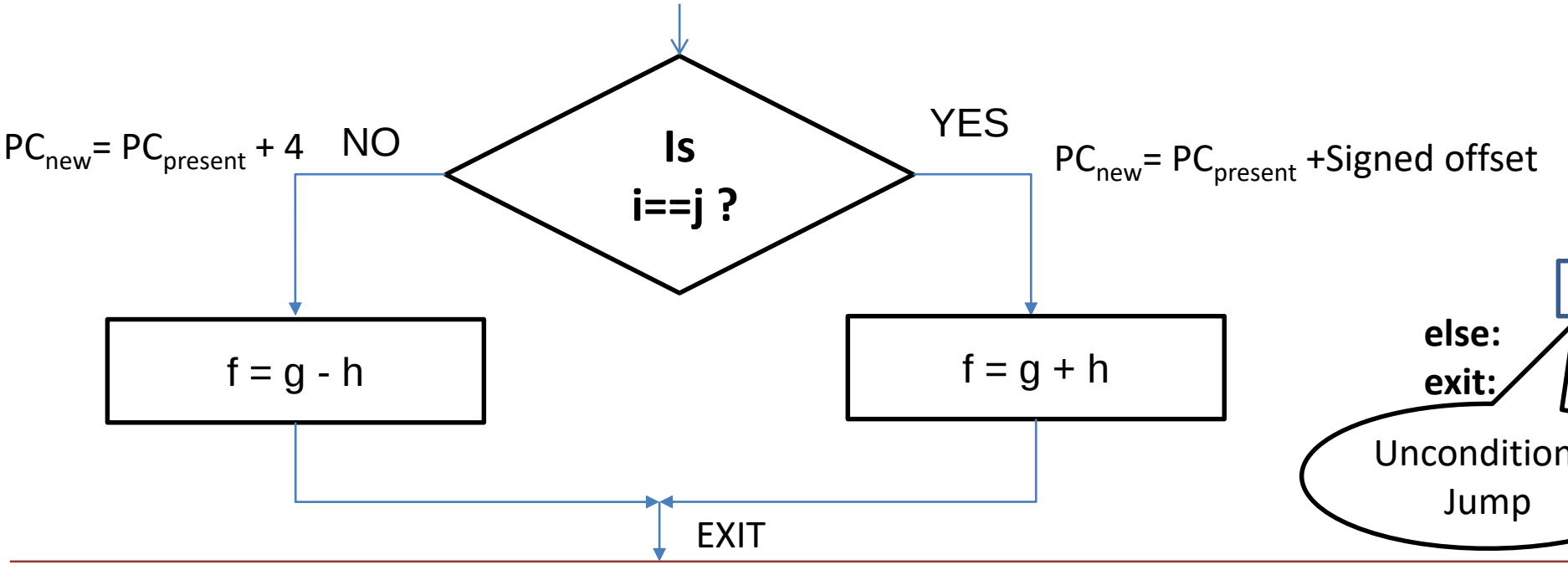
where Signed offset = Immediate($\pm 2^{11}$) * 4

PC – NO change As a result, No change in Sequence of execution i.e., executes next Instruction after branch

Compiling *if-then-else* into Conditional Branches

In the following code segment, f, g, h, i, and j are variables. If the five variables f through j correspond to the five registers x19 through x23, what is the compiled RISC-V code for this C *if* statement?
if (i == j) f = g + h; else f = g – h;

f	x19
g	x20
h	x21
i	x22
j	x23




```
    bne x22,x23,else
    add x19,x20,x21
    beq x0,x0,exit
else:
    sub x19,x20,x21
exit:
```

Unconditional Jump

Instructions – Language of Computer

Instructions for Making Decisions

Loops : for iterating a computation

Here is a traditional loop in C:
while (save[i] == k)
i += 1;
Assume that **i and k correspond to registers x22 and x24** and the **base of the array save is in x25.** What is the RISC-V assembly code corresponding to this C code?

```
loop:   slli x10, x22, 2    //   x10 = x22 * 4 = i * 4
        add  x10,x10,x25
        lw x9, 0(x10)    // x9=save[i]
        bne x9,x24,exit
        addi x22,x22,1
        beq x0,x0,loop
exit:
```

	..
	..
addr16	save[4]
addr12	save[i=3]
addr8	save[2]
addr4	save[1]
addr0	save[0]

x25 →

If i=3
Then, address of
save[i=3] = base address +i * 4;
where 4 is offset for a 32 bit data
save[i] address = x25+ i* 4 = x25+12

Instructions – Language of Computer

Instructions for Making Decisions

Loops : for iterating a computation

Basic block A sequence of instructions without branches (except possibly at the end) and without branch targets or branch labels (except possibly at the beginning).

Instructions – Language of Computer

Instructions for Making Decisions

- The full set of comparisons is **less than (<), less than or equal (≤), greater than (>), greater than or equal (≥)**, equal (=), and not equal (≠).
- Comparison of bit patterns must also deal with the dichotomy **between signed and unsigned numbers**.
  Signed Number : bit pattern with a 1 in the most significant bit represents a negative number and, of course, **is less than any positive** number, which must have a 0 in the most significant bit.
  0x8000 FFFF(Negative Number) < 0x7FFF 0000(Positive Number) in signed Number and In Unsigned Number 0x8000 FFFF > 0x7FFF 0000
- RISC-V provides instructions that handle both cases.

Instructions – Language of Computer

Instructions for Making Decisions

Instruction	Function
blt	The branch if less than (blt) instruction compares the values in registers rs1 and rs2 and takes the branch if the value in rs1 is smaller, when they are treated as two's complement numbers.
bge	Branch if greater than or equal (bge) takes the branch in the opposite case, that is, if the value in rs1 is at least the value in rs2..
bltu	Branch if less than, unsigned (bltu) takes the branch if the value in rs1 is smaller than the value in rs2 when the values are treated as unsigned numbers.
bgeu	branch if greater than or equal, unsigned (bgeu) takes the branch in the opposite case

THANK YOU

Mahesh Awati
Department of Electronics and Communication

mahesha@pes.edu

+91 9741172822