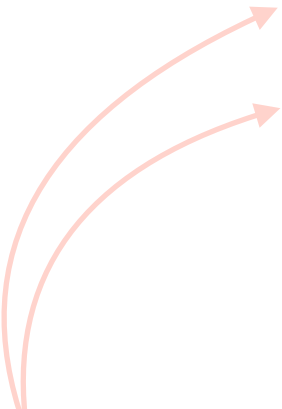




**Control-Flow Instructions**

- Control flow, at machine code level, is the order in which instructions are executed.
- All previous instructions feature implicit **trivial control flow**, that is, they simply set the program counter to the next instruction. Their main purpose is **data manipulation**.
- The following instructions have a more sophisticated effect on control flow.

# Control-Flow Instructions



beq	<code>\$rs1</code>	<code>\$rs2</code>	<code>imm</code>
jal	<code>\$rd</code>	<code>imm</code>	
jalr	<code>\$rd</code>	<code>\$rs1</code>	<code>imm</code>

- The first two instructions use a different addressing mode called pc-relative addressing at the resolution of 12 bit.
- **Branch on equal** sets the pc to  $pc + imm$  if the content of `$rs1` matches `$rs2`.
- **Jump and link** is used for procedure calls and stores the return address (address of next instruction) in `$rd`.
- **Jump and link register** is similar to `jal`, except that it uses register-relative addressing.

# Control-Flow Instructions

- Control flow, at machine code level, is the order in which instructions are executed.
- All previous instructions feature implicit **trivial control flow**, that is, they simply set the program counter to the next instruction. Their main purpose is **data manipulation**.
- The following instructions have a more sophisticated effect on control flow.