LLVM IR instructions:

- Syntax for instructions that produce a value:
  `%<name> = <opcode> <operands>`
- Syntax for other instructions:
  `<opcode> <operands>`

LLVM IR registers:

- Syntax: `%<name>`

LLVM IR data types:

- Integers: `i<number>`
- Floating-point values:
  `double`, `float`
- Arrays: `[<number> x <type>]`
- Structs: `{ <type>, … }`
- Vectors: `< <number> x <type> >`
- Pointers: `<type>*`
- Labels (i.e., basic blocks): `label`

LLVM IR opcodes:

Type or operation		Example(s)
Data movement	Stack allocation	`alloca`
	Memory read	`load`
	Memory write	`store`
	Type conversion	`bitcast`, `ptrtoint`
Arithmetic and logic	Integer arithmetic	`add`, `sub`, `mul`, `div`, `shl`, `shr`
	Floating-point arithmetic	`fadd`, `fmul`
	Binary logic	`and`, `or`, `xor`, `not`
	Boolean logic	`icmp`
	Address calculation	`getelementptr`
Control flow	Unconditional jump	`br <location>`
	Conditional jump	`br <condition>, <true>, <false>`
	Subroutines	`call`, `ret`
	Maintaining SSA form	`phi`

`%i = phi <type> [<val0>, <label0>], [<val1>, <label1>], …`

If the predecessor block is `<labelj>` set `%i` to `<valj>`