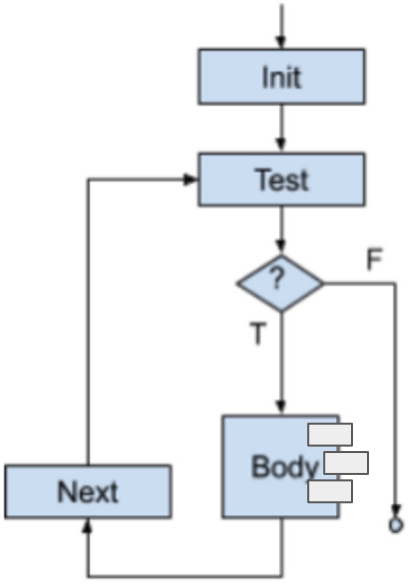
Control Flow Graph

- A graphic representation of the representation between basic blocks
- A basic block:
  - a list of instructions with
  - a single entry point (starting point)
  - a single exit point (last instruction)
- Such representations model the behavior of our code
- Recall the while loop, and other control structures

- What about subroutines calls

  (subroutine: general term for …
  methods, functions, procedures, etc.)



While Loop

Three Address Code (TAC)

- A generic assembly language in which all instructions have at most three addresses
- An address references either
  - a register location
  - a memory location
- Immediate values are stored in a location within memory

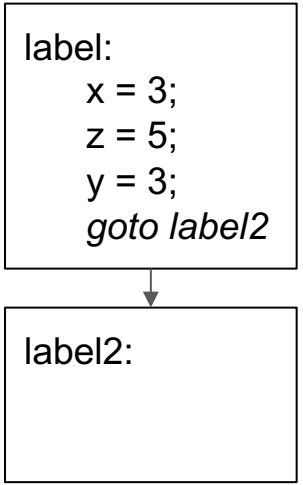
Examples:

1. a = y + x
2. a = y
3. a = x + 2
4. b = d * 2 + y
   - t0 = d * 2
   - t1 = t0 + y
   - b = t1

Assumption: the assembly language is for a register-based machine, with an infinite number of registers.

Basic Blocks

- A number of instructions in which there is
  - a single entry point (via a label), and
  - a single exit point (via a goto)
- All programs can be broken down into a set of basic blocks
- A control flow graph determines which a basic block is executed.
- Standard control flow graphs
  - if-then-else and all other variants (e.g., switch)
  - while, do-while and all other variants
  - for loop and all other variants
  - call-return



Code Flow: If-then-else

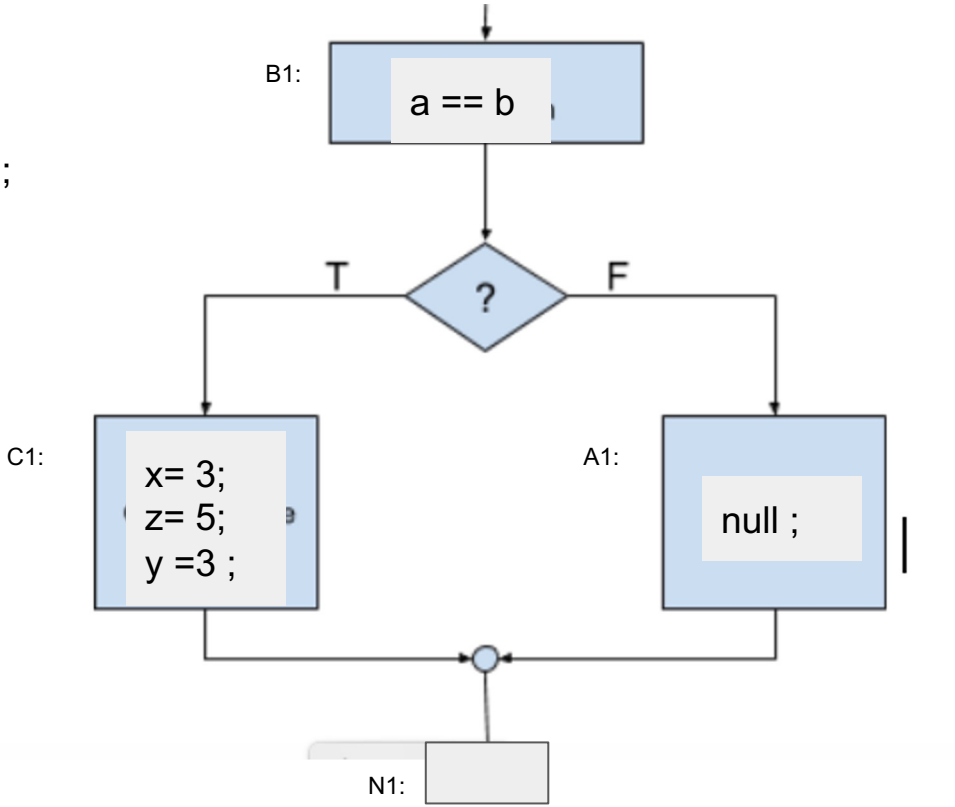
B1: a == b

if true goto C1 ;
goto A1

C1:
x = 3
z = 5
y = 3
goto N1

A1:
goto N1

N1:




```
if ( a == b ) {
    x = 3;
    z = 5;
    y = 3;
} else {
    null ;
}
```