Control Commands

- So far, command execution has been sequential.
  - Implicitly, the PC was incremented by 1 at the end of interpreting an instruction.
- For implementing conditions and loops, we need the ability to transfer control to a non-sequential instruction, either conditionally or unconditionally.
- We do this using the instructions `JUMP` (unconditional jump) and `JUMPC` (conditional jump).
  - Like **`goto`** in C.

Unconditional and Conditional Jumps

- `JUMP` *t*
    - Jump to command at Program[*t*].
    - Implementation: PC ← *t*.

Unconditional and Conditional Jumps

- `JUMP` *t*
  - Jump to command at Program[*t*].
  - Implementation: PC ← *t*.

- `JUMPC` *t*
  - Same as JUMP, except that the non-sequential successor *t* is taken only if the value on TOS is true; otherwise, execution continues with the sequential successor.
  - In either case, the stack is popped.
  - Implementation
    - Pop TOS. Let this value be Vt.
    - If Vt is true, then PC ← *t*; else, PC ← PC+1.

Example: Find Absolute Value of TOS

```
DUP
ISPOS
JUMPC 5
PUSHIMM -1
TIMES
STOP
```

```
      DUP
      ISPOS
      JUMPC Done
      PUSHIMM -1
      TIMES
Done: STOP
```