Contexts For Non-Sequential Control Flow

- Non-sequential unidirectional transfer of control can happen in several contexts.

1. `if (x < y) {...} else {...}`

2. `while (x < y) {...}`

3. `break;`

Contexts For Non-Sequential Control Flow

- Non-sequential unidirectional transfer of control can happen in several contexts.

```
1. if (x < y) {...} else {...}

2. while (x < y) {...}

3. break;
```

- In cases #1 and #2, we have a conditional expression and some number of statement blocks as sub-expressions.
- In case #3, we must be inside a loop, i.e., case #2.
- Note that case #3 can happen inside an instance of case #1 that is in turn inside an instance of case #2.

Code Generation for If-Else

- *codegen*(S, L_{NEXT}, L_{BREAK}), where S = **if**(E, S1, S2).
  - Let L_{new1} = *newlabel*() and L_{new2} = *newlabel*().
  - CE = *codegen*(E, *newlabel*(), *newlabel*()).
  - CS1 = *codegen*(S1, L_{NEXT}, L_{BREAK}).
  - CS2 = *codegen*(S2, L_{NEXT}, L_{BREAK}).
  - **return** CE + "`JUMPC` L_{new1}\n" + "L_{new2}:\n" + CS2 + "`JUMP` L_{NEXT}\n" + "L_{new1}:\n" + CS1.

Code Generation for While

- *codegen*(S, L_{NEXT}, L_{BREAK}), where S = **while**(E, S1).
  - Let L_{head} = *newlabel*().
  - CE = *codegen*(E, *newlabel*(), *newlabel*()).
  - CS1 = *codegen*(S1, L_{head}, L_{NEXT}).
  - **return** "L_{head}:\n" + CE + "`ISNIL`\n" + "`JUMPC` L_{NEXT}\n" + CS1 + "`JUMP` L_{head}\n".

Code Generation for While

- *codegen*(S, L_{NEXT}, L_{BREAK}), where S = **while**(E, S1).
  - Let L_{head} = *newlabel*().
  - CE = *codegen*(E, *newlabel*(), *newlabel*()).
  - CS1 = *codegen*(S1, L_{head}, L_{NEXT}).
  - **return** "L_{head}:\n" + CE + "`ISNIL`\n" + "`JUMPC` L_{NEXT}\n" + CS1 + "`JUMP` L_{head}\n".
- Alternate code generation template.
  - Let L_{mid} = *newlabel*() and L_{top} = *newlabel*().
  - CE = *codegen*(E, *newlabel*(), *newlabel*()).
  - CS1 = *codegen*(S1, L_{head}, L_{NEXT}).
  - return "`JUMP` L_{mid}\n" + "L_{top}:\n" + CS1 + "L_{mid}:\n" + CE + "`JUMPC` L_{top}\n".

Code Generation for Break

- *codegen*(S, L_{NEXT}, L_{BREAK}), where S = **break**.
  - **return** "`JUMP` L_{BREAK}\n".