Contexts For Alternate Choices

- A conditional expression can occur in various possible contexts.

```
bool b; b = (x < y);
```

Contexts For Alternate Choices

- A conditional expression can occur in various possible contexts.

```
bool b; b = (x < y);

int z; z = (x < y) ? x : y;
```

Contexts For Alternate Choices

- A conditional expression can occur in various possible contexts.

```
bool b; b = (x < y);

int z; z = (x < y) ? x : y;

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

Contexts For Alternate Choices

- A conditional expression can occur in various possible contexts.

```
bool b; b = (x < y);

int z; z = (x < y) ? x : y;

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

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

Contexts For Alternate Choices

- A conditional expression can occur in various possible contexts.

```
bool b; b = (x < y);

int z; z = (x < y) ? x : y;

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

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

- In all cases, evaluating the conditional expression results in executing one of two other pieces of code identified by the symbolic labels L_{TRUE} and L_{FALSE}.

Conditionals: Short-Circuit Evaluation

- Evaluating E = (E1 & E2).
  - Evaluate E1.
  - If E1 evaluates to false, go to L_{FALSE}; otherwise,
    - Evaluate E2.
    - If E2 evaluates to true, go to L_{TRUE}; otherwise, go to L_{FALSE}.
- Evaluating E = (E1 | E2).
  - Evaluate E1.
  - If E1 evaluates to true, go to L_{TRUE}; otherwise,
    - Evaluate E2.
    - If E2 evaluates to true, go to L_{TRUE}; otherwise, go to L_{FALSE}.
- Evaluating E = (! E1).
  - Evaluate E1.
  - If E1 evaluates to true, go to L_{FALSE}; otherwise, go to L_{TRUE}.

- The “Evaluate E1” and “Evaluate E2” steps may themselves be recursive.

Conditionals: Short-Circuit Code Generation

- *codegen*(E, L_{TRUE}, L_{FALSE}), where E = (E1 & E2).
  - Let L_{new} = *newlabel*() and L_{end} = *newlabel*().
  - C1 = *codegen*(E1, L_{new}, L_{FALSE}).
  - C2 = *codegen*(E1, L_{TRUE}, L_{FALSE}).
  - **return** C1 + "L_{new}:\n" + C2 + "L_{TRUE}:\n" + "`PUSHIMM 1`\n" + "`JUMP` L_{end}\n" + "L_{FALSE}:\n" + "`PUSHIMM 0`\n" + "L_{end}:\n".

Conditionals: Short-Circuit Code Generation

- *codegen*(E, L_{TRUE}, L_{FALSE}), where E = (E1 & E2).
  - Let L_{new} = *newlabel*() and L_{end} = *newlabel*().
  - C1 = *codegen*(E1, L_{new}, L_{FALSE}).
  - C2 = *codegen*(E1, L_{TRUE}, L_{FALSE}).
  - **return** C1 + "L_{new}:\n" + C2 + "L_{TRUE}:\n" + "`PUSHIMM 1`\n" + "`JUMP` L_{end}\n" + "L_{FALSE}:\n" + "`PUSHIMM 0`\n" + "L_{end}:\n".
- *codegen*(E, L_{TRUE}, L_{FALSE}), where E = (E1 | E2).
  - C1 = *codegen*(E1, L_{TRUE}, L_{new}), where L_{new} = *newlabel*().
  - C2 = *codegen*(E1, L_{TRUE}, L_{FALSE}).
  - **return** C1 + "L_{new}:\n" + C2 + "L_{TRUE}:\n" + "`PUSHIMM 1`\n" + "`JUMP` L_{end}\n" + "L_{FALSE}:\n" + "`PUSHIMM 0`\n" + "L_{end}:\n".

Conditionals: Short-Circuit Code Generation

- *codegen*(E, L_{TRUE}, L_{FALSE}), where E = (E1 & E2).
  - Let L_{new} = *newlabel*() and L_{end} = *newlabel*().
  - C1 = *codegen*(E1, L_{new}, L_{FALSE}).
  - C2 = *codegen*(E1, L_{TRUE}, L_{FALSE}).
  - **return** C1 + "L_{new}:\n" + C2 + "L_{TRUE}:\n" + "`PUSHIMM 1`\n" + "`JUMP` L_{end}\n" + "L_{FALSE}:\n" + "`PUSHIMM 0`\n" + "L_{end}:\n".
- *codegen*(E, L_{TRUE}, L_{FALSE}), where E = (E1 | E2).
  - C1 = *codegen*(E1, L_{TRUE}, L_{new}), where L_{new} = *newlabel*().
  - C2 = *codegen*(E1, L_{TRUE}, L_{FALSE}).
  - **return** C1 + "L_{new}:\n" + C2 + "L_{TRUE}:\n" + "`PUSHIMM 1`\n" + "`JUMP` L_{end}\n" + "L_{FALSE}:\n" + "`PUSHIMM 0`\n" + "L_{end}:\n".
- *codegen*(E, L_{TRUE}, L_{FALSE}), where E = (! E1).
  - **return** *codegen*(E1, L_{FALSE}, L_{TRUE}).

Example of Short-Circuit Code Generation

codegen(((a<b)||((c<d)&(e<f))),Ltrue,Lfalse)