Arrows

A CASE STUDY ON LANGUAGE DESIGN

Language Specification

- The language machine has two stacks and one register
- All instructions are encoded within one pixel wide arrows
- The register value increments for every 5 pixels the arrow travels straight
- Turns within an arrow encode instructions
  - Turn up: Push the current value onto the left stack
  - Turn down: Push the current value onto the right stack
  - Turn left: Pop x from the left stack, and subtract x from the current value
  - Turn right: Pop x from the right stack, and subtract x from the current value

Language Specification

- If the back of an arrow is flush, it is normal
- If the back of an arrow has a pixel in the middle, it is where the program starts (this arrow must start going towards the right)
- If the back of an arrow has two pixels (not the one in the middle), the register value is replaced with a character from stdin if the arrow is entered from the back
- If the head of an arrow has two extra pixels the register value is printed as an ASCII character to stdout when the arrow exits
- If an arrow has two heads when it is entered from the side it will go left if the register value is not 0, right otherwise

Let's write some code

- Preferred IDE is kolourpaint
- MS Paint will also work

Implementation

- Has both an interpreter and a compiler
- Interpreter steps through program images one pixel at a time
- Compiler creates a control flow graph of the program before generating x86-64 assembly
- Output assembly is then assembled with GCC, along with a standard library, to create an executable binary

Control Flow Graphs

- A graph showing the possible paths a program can follow
- Each node is a straight chunk of code with no jumps or conditions
- Each edge is a change in control flow
- Multiple edges from a node represent conditionals
- Edges going backwards are used to create loops

Control Flow Graphs in Arrows

- Two types of nodes:
  - Statement
    - A chunk of statements that do not alter control flow
    - Has a single edge out that leads to another node
  - Conditionals
    - A single conditional statement comparing the register value with 0
    - Two edges, one for if the register is 0, one for if it is not 0

Statements in Arrows

- END
  - Ends the program, with a return code equal to the register value
- ADD
  - Adds a constant value to the register
- PUSH_L
  - Pushes the register to the left stack
- PUSH_R
  - Pushes the register to the right stack

Statements in Arrows

- POP_SUB_L
  - Pops from the left stack and subtracts the popped value from the register
- POP_SUB_R
  - Pops from the right stack and subtracts the popped value from the register
- PRINT
  - Prints the register value as an ascii value to standard out
- READ
  - Reads in a character from standard in and replaces the register value with the ascii representation of said character

Code Generation in Arrows

- Visit each node exactly once
- For each node, emit a label so other nodes can jmp to it
- For each STATEMENT node
  - Iterate over the stored list of statements and emit instructions for each statement
  - Emit a jump to the next arrow
- For each CONDITIONAL node, emit code for the comparison and code for the jumps to the if_zero and if_else exit arrows

Last Slide

- Didn't know what to do here
- Like that arrow I put there in the theme? I'm pretty proud of it