Compilation

Ezra Varady

Goal

- Compile the stack language we used in class
- Currently targets MIPS assembly
  - should be easy to add other architectures
- Requires backend written in assembly

Backend

- Specially formatted ASM
- turned into generic library file
- “Library” file is used to create internal representation of ASM syntax
- Commonalities between ISAs allows fairly easy addition of architectures

Libraries

- Name of corresponding assembly function with prototype
- Allows for more succinct description of functions
- Uses modified stack language lexer
- Format should support multiple languages

Interpretation

- All code except raw assembly is read in as lexer tokens
- Eventually converted into strings for easier file IO and internal comparison
- Library converted into gross internal representation

Conversion to assembly

- List of all functions in the environment, including primitives
- Coerce it into label code pairs
- Use this as a lookup table to write out to the file as I read through the stack language strings I pull out of the designated file
- Arguments in assembley

ASM stack language

- Stack language primitives are represented directly as assembly for easy conversion
- JMP between more complex functions
- Fixed stack length, fixed word width
  - No arrays
- Function definition

Conclusion

- Can take stack language representation in assembly languages and use it to compile the stack language to that ISA
- 3 dup mul 4 + =>

```
li a0, 3
jal PUSH
jal DUP
jal MUL
li a0 4
jal ADD
```