Code Generation

Exercise 1

In this exercise, we will explore a way to add basic support for exceptions in a language compiled to WebAssembly.

Part 1

The first construct will introduce is the `throw` statement. The statement takes the form `throw e`, where the expression `e` computes a non-zero 32-bit integer that represents the error code.

First, we examine what to do in the simple case when we encounter the `throw` statement within the main function. In that case:

- The exception `e` should be computed,
- The value should be stored in the global variable at index `exidx`,
- The program should then stop its normal execution,
- The function at index `handleridx` should then be called,
- The execution of the program should then stop.

We assume that the handler function at index `handleridx` will not itself throw any exceptions.

To support this feature, we decide that the main function code should be wrapped in some fashion. Come up with the wrapper code so that, once the exception value is computed and stored at global index `exidx`, going to the handler is a simple jump instruction.

Part 2

In the first part, we handled exceptions occurring in the main function. However, it did not offer a solution for exceptions occurring in other functions. Indeed, it is not possible to directly jump from within a function directly to the handler code. For this part, what we will do is look at calls to other functions occurring within the main function. We will assume that, when an exception occurred within that function, that the global variable at index exidx contains a non-zero value. In that case, the execution should directly jump to the handler.

Emit the code for function calls, assuming that you know the label to jump to in case of exceptions.

Part 3

Design a way to support try/catch blocks.