WebAssembly support in Swift

WebAssembly is a platform that significantly differs from hardware platforms that Swift already supports. While it's a virtual machine, there are considerations to be taken into account when targeting it:

- WebAssembly is still at an early stage, so many features you'd be expect from other platforms are not available yet, specifically:
    1. `wasm64` variant is not specified yet, only the 32-bit `wasm32` variant is supported in WebAssembly hosts such as browsers.
    2. While a preview of multi-threading and atomics is available in some browsers and stand-alone WebAssembly hosts, the corresponding proposal haven't formally reached the implementation phase yet.
    3. Dynamic linking is not formally specified and tooling for it is not available yet.
- Binary size is a high priority requirement. Since WebAssembly payloads are usually served in browsers, one wouldn't want end users to download multi-megabyte binaries.

Nevertheless, an early implementation of the WebAssembly target is available in a separate fork. Here we're describing some decisions that were made while developing the implementation.

Relative Pointers

Relative pointers are used in Swift runtime, but currently it's not feasible to use them for the WebAssembly target due to the design of WebAssembly and lack of LLVM support. If LLVM supported subtraction relocation type on WebAssembly like `R_X86_64_PC32` or `X86_64_RELOC_SUBTRACTOR`, this issue can be solved easily.

Since `R_X86_64_PC32` and `X86_64_RELOC_SUBTRACTOR` are mainly used to generate PIC but WebAssembly doesn't require PIC because it doesn't support dynamic linking. In addition, the memory space also begins at 0, so it's unnecessary to relocate at load time. All absolute addresses can be embedded in wasm binary file directly.