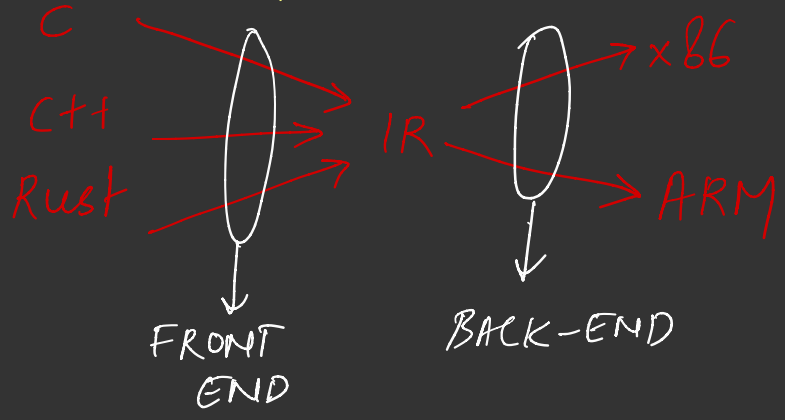
WEBASSEMBLY

C++
EMSCRIPTEN
} → Much more than that.

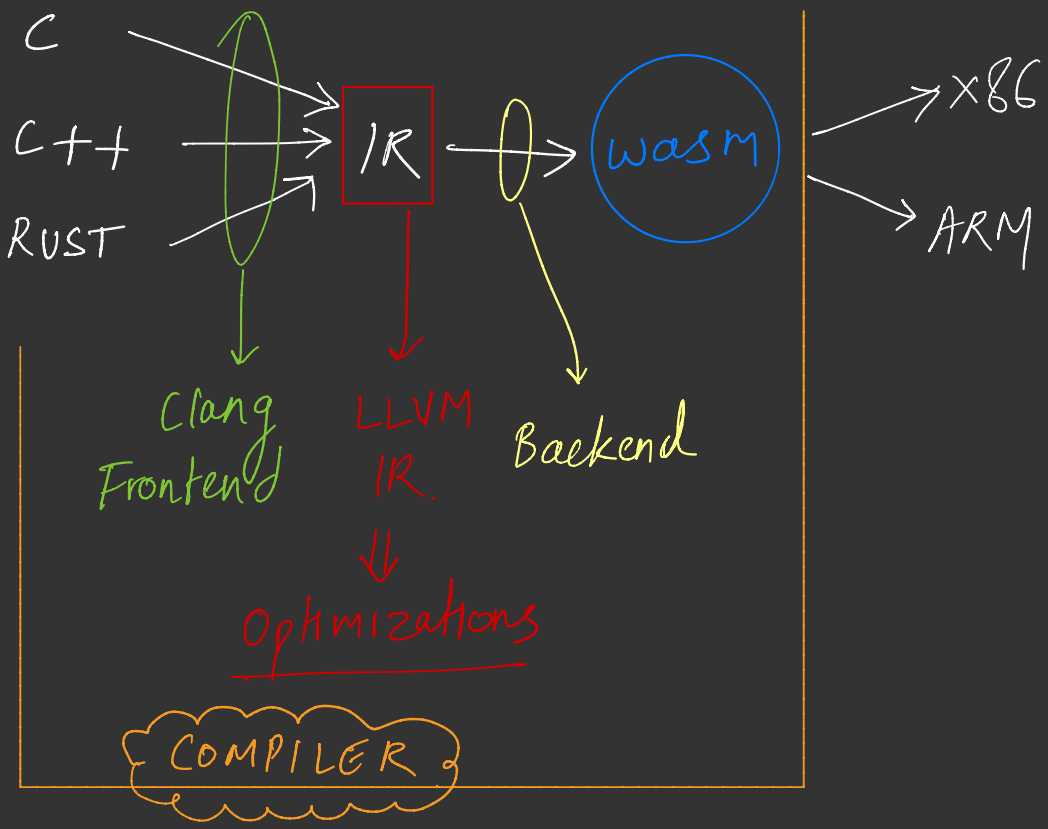
Stack based Virtual Machine → prioritize "Portability"

COMPILATION



★ WebAssembly → Virtual instructions for a conceptual machine.

- Much more direct mapping to machine code than JS source code
- Not direct mapping to specific hardware.



TOOL → EMSCRIPTEN

Includes extensive C/C++ support + C/C++ standard Library

+

OTHER LIBRARIES :
zlib + SDL + openGL

+

FILE SYSTEM & FETCH API

+

PROFILING OPTIONS .....

EMCC → Compiler to replace gcc.

EMSDK → SDK Maintenance

JAVASCRIPT Vs WEBASSEMBLY

Parsing
+
Compiling + Optimisation
+
Re-optimizing
+
Execution
+
Garbage Collection.

Interleaved.

JS source code is parsed to Abstract Syntax tree and then to IR for JS ENGINE

Fetching
+
Decode

[More compact]

WebAssembly already in IR and hence only decode is required

Compiling + Optimization

(Already done in LLVM compilation)

No need for dynamic typing

+

Execution

Depending on code and structure, 10% - 800% faster

GARBAGE COLLECTION — Not required

IMPROVING PERFORMANCE

ISSUE SOLVED !!
+
INTERFACE TYPE PROPOSAL
WEB IDL INTEGRATION

① Faster Function calls

* TRAMPOLINING → Calling WebAssembly function in JS slower because JIT doesn't know how to directly deal with WebAssembly.

100x slower than if JIT directly knew what to do

* Not noticeable if a single large task passed

② Faster Load Time

* Parallelize compilation with execution

* 2-compiler system

FIREFOX

Faster compiler running ahead

Full optimization in the background

OTHER IMPROVEMENTS

EXCEPTION HANDLING → CURRENT IMPLEMENTATION SLOW (EMSCRIPTEN)

SHARED MEMORY CONCURRENCY → INTER-THREAD COMMUNICATION

SIMD → GAME DEVELOPMENT