OSPP 2023 Challenge for #2438 Solution

中科院计算所 孙启楚 Qichu Sun, ICT CAS

Task A: Use Apple Clang compiler to build WasmEdge with the tests

参考WasmEdge的GitHub Workflow文件，结合官方文档及我本地环境实际情况，编译步骤如下：

使用MacPorts安装必要的依赖

由于我在编译WasmEdge项目前已经因其他软件需求使用MacPorts（而非官方使用的HomeBrew）提前安装了相关依赖，故该部分是我根据官方文档给出的MacPorts参考命令。

```
sudo port install boost cmake ninja llvm-13
```

配置LLVM目录及C/CPP编译器环境变量

由于MacPorts默认软件安装位置与HomeBrew不同，本处给出的是基于MacPorts环境的命令。

```
export LLVM_DIR=/opt/local/libexec/llvm-13/lib/cmake/
export CC=clang
export CXX=clang++
```

此处，根据实测，在macOS平台，不配置`CC`及`CXX`环境变量也能够正常编译。这应当是由于macOS平台默认的C/CPP编译器在很多年前就已切换到Clang/Clang++，系统现存的`gcc`工具实际也是`clang`的副本。

```
$ gcc --version
Apple clang version 14.0.0 (clang-1400.0.29.202)
Target: arm64-apple-darwin22.4.0
Thread model: posix
InstalledDir:
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolc
hain/usr/bin
```

此处使用HomeBrew或MacPorts安装`llvm`应是必要条件，WasmEdge项目编译需要使用`llvm`包提供的部分cmake配置，但不一定需要`llvm`与Apple Clang版本对齐。

编译WasmEdge

参考官方文档，使用如下命令即可。

```
cmake -Bbuild -GNinja -DWASMEDGE_BUILD_TESTS=ON .
cmake --build build
```

运行测试

参考官方文档，使用如下命令即可。

```
cd build
DYLD_LIBRARY_PATH=$(pwd)/lib/api ctest
```

此外，根据实测，在macOS平台，不配置`DYLD_LIBRARY_PATH`直接在`build`目录运行`ctest`同样可以成功。

Task B: Add new WASI host function for return a value with adding 2 input values

实现

Host function的实现位于commit `ef9e32484f2add4a941cda9f44091c239773cd4b`中，该commit的实现参考了文档对host function的描述及WasmEdge实现随机数生成的host function commit。

在文档叙述和随机数生成的commit中，host function传入的参数采用了两种不同的风格。在文档中直接给出了生成返回两数之和（`a+b`）的示例。该示例接受存储`a`和`b`的数组起始指针及结果输出位置指针，而在随机数生成的commit中，其只接受结果输出的数组指针及结果预期大小两个参数。为进一步熟悉框架并为bonus部分提供更多可能的测试角度，我选择了使用更接近后者风格的输出方式，示例接受`a`和`b`的数组起始指针，运算结果将直接存放在数组首个元素，覆盖`a`的内容，`b`的内容保持不变。

涉及文件

根据随机数生成的commit，实现一个host function需要修改如下的文件：

```
include/host/wasi/wasifunc.h # 提供host function的类基础定义及body方法原型
lib/host/wasi/wasifunc.cpp # 提供host function的body方法实现
lib/host/wasi/wasimodule.cpp # 注册host function
```

代码细节

由于用户并不总是能够传递正确形式的参数，在host function的body方法中我们需要对传入参数做合法性检查。

首先，如果用户传入的CallFrame中不包含合法的Memory，我们将无法读出预期的数据，因此需要检查从Frame中获取到的MemInst是否合法。

```
auto *MemInst = Frame.getMemoryByIndex(0);
if (MemInst == nullptr) {
  return __WASI_ERRNO_FAULT;
}
```

此外，如果用户传入的buffer指针不合法或后续并不存在充足的空间，我们也无法完成预期的运算，因而需要检查此处获取到的buffer指针及其空间大小是否符合我们要求。参考随机数生成的实现，此处同样使用了`unlikely()`来标注该分支跳转可能较小。通常而言，标注后编译器会倾向于将不跳转的分支放在与分支指令紧邻的位置及在可能时生成带方向hint的分支指令，从而允许现代超标量处理器在更为常见的分支不跳转情形且取指带宽充足时一次性取回分支指令之前基本块入口直到正确分支基本块出口的指令，提升指令供应能力，提升程序执行速度。

```
auto *const Buf = MemInst->getPointer<int32_t *>(BufPtr, WasiBufLen);
if (unlikely(Buf == nullptr)) {
  return __WASI_ERRNO_FAULT;
}
```

此外，注意到随机数生成的实现代码，其调用了`Env.randomGet()`方法完成实际的随机数生成运算，为提前熟悉OSPP中可能涉及到的复杂模块依赖关系，我仿照为`Env`实现了`Env.add2Get()`方法，在该方法中完成实际的加法运算。类似地，在`add2Get()`方法中同样可能出现错误，此处仿照随机数生成做了检查与unlikely标注。

```
if (auto Res = Env.add2Get({Buf, WasiBufLen}); unlikely(!Res)) {
  return Res.error();
}
```

在`add2Get()`方法中，为安全性考虑对buffer span做了最终的范围检查并完成了实际的加法运算。

```
auto BufferSpan = cxx20::as_writable_bytes(Buffer);
  if (unlikely(BufferSpan.size() < 2 * sizeof(int32_t))) {
    return WasiUnexpect(__WASI_ERRNO_FAULT);
  }
  auto a = reinterpret_cast<int32_t *>(&BufferSpan[0]);
  auto b = reinterpret_cast<int32_t *>(&BufferSpan[sizeof(int32_t)]);
  *a = *a + *b;
```

综上，我基于WasmEdge项目文档及类似commit实现了两数相加的host function。

Bonus: Add unit tests in the test file

Challenge本身要求将测试实现在socket的测试（`test/host/socket`）中，但注意到随机数生成的单元测试被放置于`test/host/wasi/wasi.cpp`，从逻辑性出发，我选择将加法的测试同样放置于`test/host/wasi/wasi.cpp`。考虑到不同测试用例间独立运行，在有必要时调整单元测试所在文件是一项trivial的工作。

测试覆盖了非法module、非法buffer、正数+正数、正数+负数、负数+正数及负数+负数六种情形，测试强度应当略超过本项目已有各测试。

通过提供module为nullptr的Frame或非法的buffer pointer，可以测试sum2 host function的输入合法性检查。通过正数与负数的所有可能组合检查可以有效检验有符号运算结果的正确性。

```
*MemInst.getPointer<int32_t *>(0) = 0x3210333f;
*MemInst.getPointer<int32_t *>(sizeof(int32_t)) = -0x2;
EXPECT_TRUE(WasiSum2Get.run(
    CallFrame,
    std::initializer_list<WasmEdge::ValVariant>{UINT32_C(0)},
    Errno));
EXPECT_EQ(Errno[0].get<int32_t>(), __WASI_ERRNO_SUCCESS); // verify if
exit wth success
EXPECT_EQ(*MemInst.getPointer<const int32_t *>(0), INT32_C(0x3210333d));
// verify add result
```

此处，为进一步确保检查的充分性，考虑到补码的编码规则，测试所采用的正负数均保证了32位范围内每一字节均不全0，避免被加数含0导致测试不出错误结果。

```
*MemInst.getPointer<int32_t *>(0) = 0x3210333f; // 0x3f 0x33 0x10 0x32,
all non-zero
*MemInst.getPointer<int32_t *>(sizeof(int32_t)) = -0x2; // 0xfe 0xff 0xff
0xff, all non-zero
```

为保证加法函数不会覆盖非预期位置，测试仿照随机数的检查，对加数b在测试前后取值不变性也做了检查。

```
EXPECT_EQ(*MemInst.getPointer<const int32_t *>(sizeof(int32_t)),
INT32_C(-0x2));
```

综上，我为Task B实现了对应的单元测试。

附录

在macOS平台运行WasmEdge的测试log：

```
# Version: 2023.4.29 master with sum2 test
Test project /Users/sqc/Documents/WasmEdge/build
      Start  1: wasmedgeAOTCoreTests
 1/22 Test  #1: wasmedgeAOTCoreTests .............Bus error***Exception:
0.51 sec
      Start  2: wasmedgeAOTCacheTests
 2/22 Test  #2: wasmedgeAOTCacheTests ............   Passed    0.02 sec
      Start  3: wasmedgeAOTBlake3Tests
 3/22 Test  #3: wasmedgeAOTBlake3Tests ...........   Passed    0.01 sec
      Start  4: wasmedgeMixcallTests
 4/22 Test  #4: wasmedgeMixcallTests .............Bus error***Exception:
0.35 sec
      Start  5: wasmedgeCommonTests
```

```
 5/22 Test  #5: wasmedgeCommonTests ..............   Passed    0.00 sec
      Start  6: wasmedgeLoaderFileMgrTests
 6/22 Test  #6: wasmedgeLoaderFileMgrTests .......   Passed    0.01 sec
      Start  7: wasmedgeLoaderASTTests
 7/22 Test  #7: wasmedgeLoaderASTTests ...........   Passed    0.01 sec
      Start  8: wasmedgeExecutorCoreTests
 8/22 Test  #8: wasmedgeExecutorCoreTests ........   Passed    1.31 sec
      Start  9: wasmedgeThreadTests
 9/22 Test  #9: wasmedgeThreadTests ..............***Failed    1.26 sec
      Start 10: wasmedgeAPIUnitTests
10/22 Test #10: wasmedgeAPIUnitTests .............Bus error***Exception:
0.70 sec
      Start 11: wasmedgeAPIVMCoreTests
11/22 Test #11: wasmedgeAPIVMCoreTests ...........   Passed    1.19 sec
      Start 12: wasmedgeAPIStepsCoreTests
12/22 Test #12: wasmedgeAPIStepsCoreTests ........   Passed    1.11 sec
      Start 13: wasmedgeAPIAOTCoreTests
13/22 Test #13: wasmedgeAPIAOTCoreTests ..........***Exception: SegFault
2.07 sec
      Start 14: wasmedgeExternrefTests
14/22 Test #14: wasmedgeExternrefTests ...........   Passed    0.30 sec
      Start 15: wasiSocketTests
15/22 Test #15: wasiSocketTests ..................***Failed    0.31 sec
      Start 16: wasiTests
16/22 Test #16: wasiTests ........................   Passed    3.13 sec
      Start 17: wasmedgeHostMockTests
17/22 Test #17: wasmedgeHostMockTests ............   Passed    0.36 sec
      Start 18: expectedTests
18/22 Test #18: expectedTests ....................   Passed    0.01 sec
      Start 19: spanTests
19/22 Test #19: spanTests ........................   Passed    0.00 sec
      Start 20: poTests
20/22 Test #20: poTests ..........................   Passed    0.01 sec
      Start 21: wasmedgeMemLimitTests
21/22 Test #21: wasmedgeMemLimitTests ............   Passed    0.30 sec
      Start 22: wasmedgeErrinfoTests
22/22 Test #22: wasmedgeErrinfoTests .............   Passed    0.00 sec

73% tests passed, 6 tests failed out of 22

Total Test time (real) =  12.97 sec

The following tests FAILED:
          1 - wasmedgeAOTCoreTests (Bus error)
          4 - wasmedgeMixcallTests (Bus error)
          9 - wasmedgeThreadTests (Failed)
         10 - wasmedgeAPIUnitTests (Bus error)
         13 - wasmedgeAPIAOTCoreTests (SEGFAULT)
         15 - wasiSocketTests (Failed)
Errors while running CTest
Output from these tests are in:
/Users/sqc/Documents/WasmEdge/build/Testing/Temporary/LastTest.log
Use "--rerun-failed --output-on-failure" to re-run the failed cases
verbosely.
```

此外，根据上图，时至此版本，master分支尚未完成之前邮件所提及的结构体大小不匹配的bug修复。

对应单独运行wasiTest的log，新加入的测试名为Sum2：

```
$ ./build/test/host/wasi/wasiTests
[==========] Running 11 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 11 tests from WasiTest
[ RUN      ] WasiTest.Args
[       OK ] WasiTest.Args (0 ms)
[ RUN      ] WasiTest.Envs
[       OK ] WasiTest.Envs (0 ms)
[ RUN      ] WasiTest.ClockRes
[       OK ] WasiTest.ClockRes (0 ms)
[ RUN      ] WasiTest.PollOneoffSocketV1
[       OK ] WasiTest.PollOneoffSocketV1 (1553 ms)
[ RUN      ] WasiTest.PollOneoffSocketV2
[       OK ] WasiTest.PollOneoffSocketV2 (1578 ms)
[ RUN      ] WasiTest.ClockTimeGet
[       OK ] WasiTest.ClockTimeGet (0 ms)
[ RUN      ] WasiTest.ProcExit
[       OK ] WasiTest.ProcExit (0 ms)
[ RUN      ] WasiTest.Random
[       OK ] WasiTest.Random (0 ms)
[ RUN      ] WasiTest.Sum2
[       OK ] WasiTest.Sum2 (0 ms)
[ RUN      ] WasiTest.Directory
[       OK ] WasiTest.Directory (2 ms)
[ RUN      ] WasiTest.SymbolicLink
[       OK ] WasiTest.SymbolicLink (0 ms)
[----------] 11 tests from WasiTest (3136 ms total)

[----------] Global test environment tear-down
[==========] 11 tests from 1 test suite ran. (3136 ms total)
[  PASSED  ] 11 tests.
```