UDK 001

Jakbarov Odiljon Otamirzayevich, prorektor

Majidov Avarxon Mahmudxon o'g'li, talaba

Namangan muhandislik-qurilish instituti

LOW LEVEL VIRTUAL MACHINE ARXITEKTURASI

Anotatsiya: Ushbu maqolada low level virtual machinening kelib chiqish tarixi, Intermediate Representation, frontend va backend bo'yicha so'z boradi.

Kalit so'zlar: Kompilyator, LLVM, dasturlash, versiya, machine, frontend, backend, assembly, IR.

LOW LEVEL VIRTUAL MACHINE ARCHITECTURE

Anotation: this article covers the history of the origin of low level virtual machine, Intermediate Representation, frontend, and backend.

Keywords: compiler, LLVM, programming, version, machine, frontend, backend, assembly, IR.

LLVM — kompilyator va toolchain texnologiyalari bo'lib, har qanday dasturlash tili uchun frontend va har qanday instruction set arxitekturasi uchun backendni ishlab chiqish uchun ishlatilishi mumkin. LLVM tildan mustaqil Intermediate Representation (IR) atrofida ishlab chiqilgan bo'lib, u portativ, high-level assembly language bo'lib xizmat qiladi, uni multiple passe orqali turli xil o'zgarishlar bilan optimallashtirish mumkin.

LLVM C++ tilida yozilgan va compile-time, link-time, run-time va idle-time optimallashtirish uchun mo'ljallangan. Dastlab C va C++ uchun tatbiq etilgan LLVMning language-agnostic dizayni o'shandan beri turli xil frontendlarni yaratdi: LLVM dan foydalanadigan (yoki bevosita LLVM dan foydalanmaydigan, lekin kompilyatsiya qilingan dasturlarni LLVM IR sifatida yarata oladigan) kompilyatorli tillar ActionScript, Ada, C#, Common Lisp, PicoLisp, Crystal, CUDA, D, Delphi, Dylan, Forth, Fortran, Free Basic, Free Pascal, Graphical G,

Halide, Haskell, Java bytecode, Julia, Kotlin, Lua, Objective-C, OpenCL, PostgreSQL SQL va PLpgSQL, Ruby, Rust, Scala, Swift, XC, Xojo va Zig.

LLVM loyihasi 2000 yilda Urbana–Champaigndagi Illinois universitetida Vikram Adve va Chris Lattner rahbarligida boshlangan. LLVM dastlab statik va dinamik dasturlash tillari uchun dinamik kompilyatsiya usullarini o'rganish uchun tadqiqot infratuzilmasi sifatida ishlab chiqilgan. LLVM Illinois/NCSA Open Source License ruxsat beruvchi bepul dasturiy ta'minot litsenziyasi ostida chiqarilgan.

2005-yilda Apple Inc. Lattnerni yolladi va Apple ishlab chiqarish tizimlarida turli maqsadlarda foydalanish uchun LLVM tizimida ishlash uchun jamoa tuzdi. LLVM Xcode 4-dan beri macOS va iOS uchun Apple Xcode ishlab chiqish vositalarining ajralmas qismi bo'lib kelgan.

2006 yilda Lattner Clang nomli yangi loyiha ustida ishlay boshladi. Clang front-end va LLVM back-end kombinatsiyasi Clang/LLVM yoki oddiygina Clang deb ataladi. LLVM nomi dastlab Low Level Virtual Machine initialism edi. Bu initialismni "chalkash" va "nomaqbul" qildi va 2011 yildan beri LLVM "rasmiy ravishda endi qisqartma emas", balki loyiha endi an'anaviy virtual mashinalarga qaratilmaydi. Loyiha LLVM intermediate representation (IR), LLVM debuggerini, C++ standart kutubxonasining LLVM dasturini (C++11 va C++14 to'liq qo'llab-quvvatlashi bilan) va boshqalarni o'z ichiga oladi.

LLVM LLVM Foundation tomonidan boshqariladi. Kompilyator muhandisi Tanya Lattner 2014-yilda uning prezidenti bo'ldi va hozirgacha o'z lavozimida.

"LLVMni loyihalash va implement qilish uchun" Association for Computing Machinery assotsiatsiyasi Vikram Adve, Chris Lattner, va Evan Chengga 2012 yil ACM dasturiy ta'minot tizimi mukofotini taqdim etdi. Loyiha dastlab UIUC litsenziyasi ostida mavjud edi. 2019-yilda chiqarilgan v9.0.0 dan keyin, LLVM LLVM istisnolari bilan Apache 2.0 litsenziyasiga qayta litsenziyalandi.

LLVM kompilyatordan intermediate representation (IR) kodini olib, optimallashtirilgan IR-ni chiqaradigan to'liq kompilyator tizimining o'rta qatlamlarini taqdim etishi mumkin. Ushbu yangi IR keyinchalik maqsadli platforma uchun mashinaga bog'liq assembly language kodiga aylantirilishi va bog'lanishi mumkin. LLVM GNU Compiler Collection (GCC) toolchaindan IR ni qabul qilishi mumkin, bu esa uni ushbu loyiha uchun yozilgan mavjud kompilyator frontendining keng doirasi bilan ishlatishga imkon beradi. LLVM language-independent instruction va type systemni qo'llab-quvvatlaydi. Har bir instruction static single assignment(SSA) shaklida mavjud, ya'ni har bir o'zgaruvchi (yozilgan registr deb ataladi) bir marta tayinlanadi va keyin muzlatiladi.Bu o'zgaruvchilar orasidagi bog'liqliklarni tahlil qilishni soddalashtirishga yordam beradi. LLVM kodni an'anaviy GCC tizimida bo'lgani kabi statik ravishda kompilyatsiya qilish imkonini beradi yoki Java'ga o'xshash IR-dan mashina kodiga just-in-time compilation (JIT) orqali kech kompilyatsiya qilish uchun qoldiriladi.

Type systemi butun yoki floating-point raqamlar kabi asosiy turlardan va beshta hosila turlaridan iborat: pointerlar, arraylar, vektorlar, structuralar va funksiyalar.

LLVM JIT kompilyatori runtimeda dasturdan keraksiz statik tarmoqlarni optimallashtirishi mumkin va shuning uchun dasturda ko'plab variantlar mavjud bo'lgan hollarda qisman baholash uchun foydali bo'ladi, ularning aksariyati ma'lum bir muhitda keraksizligini osongina aniqlash mumkin. Bu xususiyat Mac OS X

Front end. LLVM dastlab GCC stekidagi mavjud kod generatorini almashtirish uchun yozilgan edi va GCCning ko'pgina frontend qismlari u bilan ishlash uchun o'zgartirildi, natijada endi LLVM-GCC to'plami tugatildi. O'zgartirishlar odatda GIMPLE-to-LLVM IR bosqichini o'z ichiga oladi, shuning uchun GCC ning GIMPLE tizimi o'rniga LLVM optimallashtiruvchilari va

kodegen ishlatilishi mumkin. Apple Xcode 4.x (2013) orqali LLVM-GCC-ning muhim foydalanuvchisi edi. GCC frontendidan bunday foydalanish asosan vaqtinchalik chora hisoblangan, ammo Clang-ning paydo bo'lishi va LLVM va Clang-ning zamonaviy va modulli kod bazasining afzalliklari (shuningdek, kompilyatsiya tezligi) asosan eskirgan.

LLVM hozirda Ada, C, C++, D, Delphi, Fortran, Haskell, Julia, Objective-C, Rust, va Swift-ni turli xil frontend qismlarida foydalangan holda kompilyatsiya qilishni qo'llab-quvvatlaydi. LLVM ga bo'lgan keng qiziqish turli tillar uchun yangi frontendlarni ishlab chiqish bo'yicha bir qancha sa'y-harakatlarga olib keldi. Eng ko'p e'tiborni tortgan Clang, C, C++ va Objective-C ni qo'llab-quvvatlaydigan yangi kompilyator edi. Asosan Apple tomonidan qo'llab-quvvatlanadigan Clang GCC tizimidagi C/Objective-C kompilyatorini integratsiyalashgan ishlab chiqish muhitlari (IDE) bilan osonroq integratsiyalashgan va ko'p ish zarralarini kengroq qo'llab-quvvatlaydigan tizim bilan almashtirishga qaratilgan. OpenMP directivelarini qo'llab-quvvatlash Clang-ga 3.8 versiyasidan qo'shilgan.

Utrecht Haskell kompilyatori LLVM uchun kod yaratishi mumkin. Generator rivojlanishning dastlabki bosqichida bo'lsa-da, ko'p hollarda u C kod generatoriga qaraganda samaraliroq bo'lgan. LLVM-dan foydalangan holda Glasgow Haskell

Compiler (GHC) backend mavjud bo'lib, u GHC yoki C kodini yaratish orqali kompilyatsiya qilish va undan keyin kompilyatsiya qilish orqali mahalliy kodga nisbatan kompilyatsiya qilingan kodni 30% ga tezlashtiradi.

Intermediate Representation – IR. LLVM ning yadrosi intermediate representation (IR), assembleyerga o'xshash low-leveldagi dasturlash tilidir. IR - bu reduced instruction set computer (RISC) ko'p tafsilotlarini qisqartiradigan kuchli terilgan reduced instruction set computer instructionlari to'plami. Masalan, chaqiruv konventsiyasi aniq argumentlar bilan call va ret instructionlari orqali mavhumlashtiriladi. Shuningdek, belgilangan registrlar to'plami o'rniga IR %0, %1

ko’rinishdagi cheksiz vaqtincha to’plamdan foydalanadi. LLVM IR uchta ekvivalent shaklini qo’llab-quvvatlaydi: odam o’qiy oladigan assembly format(human-readable assembly format), frontendlar uchun mos xotira formati va ketma-ketlashtirish uchun zich bitkod formati. Oddiy "Hello, world!" IR formatidagi dastur:

```
@.str = internal constant [14 x i8] c"hello, world\0A\00"
declare i32 @printf(ptr, ...)
define i32 @main(i32 %argc, ptr %argv) nounwind {
entry:
  %tmp1 = getelementptr [14 x i8], ptr @.str, i32 0, i32 0
  %tmp2 = call i32 (ptr, ...) @printf( ptr %tmp1 ) nounwind
  ret i32 0}
```

Amaldagi ko’plab turli konventsiyalar va turli maqsadlar tomonidan taqdim etilgan xususiyatlar LLVM haqiqatan ham maqsadli mustaqil IR ishlab chiqara olmasligini va ba’zi belgilangan qoidalarni buzmasdan uni qayta yo’naltira olmasligini anglatadi. Texnik hujjatlarda aniq aytib o’tilganidan tashqari target dependence misollarini 2011 yilda onlayn distribution uchun mo’ljallangan LLVM IR ning to’liq target-independent varianti bo’lgan "wordcode" bo’yicha taklifda topish mumkin. LLVM loyihasi shuningdek, Dialect nomli plagin arxitekturasidan foydalangan holda qayta foydalanish mumkin va kengaytiriladigan kompilyator infratuzilmasini yaratishga yordam beruvchi MLIR nomli intermediate representationning yana bir turini taqdim etadi.

Backend. 13-versiyada LLVM IA-32, x86-64, ARM, Qualcomm Hexagon, MIPS, Nvidia Parallel Thread Execution (PTX; LLVM hujjatlarida NVPTX deb ataladi), PowerPC, AMD TeraScale, ko’pchilik AMD kabi instructionlar to’plamlarini qo’llab-quvvatlaydi. Ayrim funksiyalar ayrim platformalarda mavjud

emas. Aksariyat xususiyatlar IA-32, x86-64, z/Architecture, ARM va PowerPCuchun mavjud. RISC-V 7-versiyadan boshlab qo'llab-quvvatlanadi.

Ilgari LLVM boshqa backendlarni ham to'liq yoki qisman qo'llab-quvvatlagan, jumladan C backend, Cell SPU, mblaze (MicroBlaze),AMD R600, DEC/Compaq Alpha (Alpha AXP) va Nios2, lekin bu hardware asosan eskirgan va LLVM ishlab chiquvchilari qo'llab-quvvatlash va texnik xizmat ko'rsatish xarajatlari endi oqlanmasligiga qaror qilishdi. LLVM machine code (MC) kichik loyihasi matnli shakllar va mashina kodi o'rtasida machine instructionlarini tarjima qilish uchun LLVM asosidir. Ilgari, LLVM assemblerni mashina kodiga tarjima qilish uchun tizim assembleriga yoki toolchaini tomonidan taqdim etilganiga tayangan. LLVM MC ning integratsiyalashgan assembleri IA-32, x86-64, ARM va ARM64 kabi ko'pgina LLVM maqsadlarini qo'llab-quvvatlaydi. Ba'zi maqsadlar uchun, jumladan, turli MIPS instructionlar to'plamlari uchun, integratsiyalashgan assembly yordami foydalanish mumkin, lekin hali ham beta bosqichida.

Adabiyotlar

1. Бруно Карлос Лопес Рафаэль Аулер. LLVM: инфраструктура для разработки компиляторов Москва, 2015

2. https://foundation.llvm.org/

3. https://llvm.org/docs/