Překladač jazyka HeroC

Specifikace jazyka

Základy specifikace jazyka jsou popsány v souboru `docs/heroc-2016.pdf`. HeroC je odvozen z jazyka C, který výrazně zjednodušuje (například uvažuje pouze jediný datový typ) a přidává vlastní specifika.

Jako pravdivostní hodnota `true` se uvažuje jakákoliv nenulová hodnota, tedy `false` je pouze 0. Jazyk obasuje pouze datový typ `long` (64 bitů). Z toho důvodu není nutné uvádět typ návratové hodnoty funkce při její deklaraci.

Zápis proměnné

Zápis proměnných je povolen pouze na záčátku bloku kódu v následujícím formátu: `long a;`, `long a = 1;`, `long a = 0xf001;`, `long a = b = 42;`, `long c = {3,1,4};`

Během syntaktické analýzy dochází k převodu čísel v osmičkové/šestnáctkové soustavě do soustavy desítkové.

Zápis funkce

```
nazev(arg1, arg2){ long a; . . . return a; }
```

Podporované binární operátory

!= % %= & && &= * *= + += - -= / /= < << <<= <= = == > >= >> >>= ^ ^= | |= ||

Podporované unární prefixové operátory

* & ++ -- ~ ! -

Podporované unární postfixové operátory

-- ++

Spuštění

Překladač jazyka je napsán v jazyce Python3 s využítím nástroje ANTLR4 pro specifikaci gramatiky jazyka a tvorby DFT. Je tedy nutné mít nainstalovaný interpret `python3` a `antlr4-python3-runtime`.

Vypsání abstraktního syntaktického stromu na standardní výstup je možné provést příkazem `cat ./cesta/k/souboru/pokus.heroc | python3 ./compiler/print_tree.py`

Pro přeložení programu zapsaného v HeroC a vypsání na standardní výstup provede příkaz `cat ./cesta/k/souboru/pokus.heroc | python3 ./compiler/compiler.py > pokus.s`, případné chyby jsou směrovány na chybový výstup, je tedy možné použít zápis `... 2> err.out`

Spustitelný soubor je možné získat spuštěním `gcc -m64 -o pokus pokus.s herocio.c`

Pokud předpokládáme že testovací soubory `examples/exampleXX.heroc` pokrývají veškerou funkčnost jazyka HeroC, pak lze tento překladač považovat za plnohodnotný. Otestování všech příkladů lze spustit příkazem `./compiler-test.sh`.

Struktura překladače

HeroC.g4

Lexikální a syntaktická pravidla gramatiky pro nástroj ANTLR4. Výchozím bodem pro její tvorbu byla referenční gramatika jazyka C pro ANTLR https://github.com/antlr/grammars-v4/blob/master/c/C.g4.

HeroCCustomVisitor.py

Visitor procházející strom vygenerovaný parserem, ze kterého vytváří abstraktní syntaktický strom.

compile.py

Překladač jazyka HeroC, na standardním vstupu očekává program v jazyce HeroC.

print_tree.py

Nástroj pro výpis AST, na standardním vstupu očekává program v jazyce HeroC.

ASM/ASM.py

Třída specifikuje zálohované registry, direktivu programu v assembleru a zjednodušuje generování instrukcí v assembleru. Obsahuje seznam zálohovaných registrů před voláním funkce.

ASM/Registers.py

Výčtový typ představující používané registry.

AST/Node.py

Základní třída představující uzel abstraktního syntaktického stromu, rodič následujících typů. Pro získání kódu uzlu v assembleru je nutné přetížit metodu `asm()`, předávání hodnoty uzlu se uvažuje skrze registr `%RAX`.

AST/Array.py

Třída představující pole. Během generování kódu vloží prvky na zásobník a vrátí adresu prvního prvku.

AST/BinaryExpression.py

Třída představující binární operaci (viz. specifikace jazyka).

AST/Block.py

Třída představující blok kódu. Obsahuje tabulku vazeb deklarovaných proměnných a jejich adres.

AST/DoWhileExpression.py

Třída představující Do-While výraz. Obsahuje testovanou podmínku a tělo cyklu. Tělo cyklu je vykonáno před testováním podmínky.

AST/Environment.py

Třída představující tabulku vazeb symbolů a jejich adres.

AST/ForExpression.py

Třída představující For cyklus. Obsahuje inicializační výraz, testovanou podmínku, tělo cyklu a výraz prováděný po vykonání těla.

AST/Function.py

Třída představující funkci. Obsahuje název funkce a její tělo.

AST/FunctionCall.py

Třída představující volání funkce. Dodržuje konvenci CDECL, tedy prvních 6 argumentů je předáno přes registry, zbylé jsou vloženy na zásobník.

AST/Identifier.py

Třída představující identifikátor. Obsahuje metodu pro výpočet adresy proměnné, v případě nenalezení v hierarchii prostředí nastane chyba.

AST/IfEvalStatement.py

Třída představující inline zápis If-Else podmínky ve tvaru `test ? pravda : nepravda`

AST/IfStatement.py

Třída představující klasický zápis If-Else podmínky. Obsahuje testovaný výraz, blok vykonaný v případě splnění testovaného výrazu a nepovinně blok vykonaný v případě nesplnění podmínky.

AST/JumpStatement.py

Třída představující výraz skoku. Může se jednat o přerušení cyklu `break`, přerušení aktuální iterace cyklu `continue` nebo ukončení a navrácení hodnoty z funkce `return`. Je-li volán `return` bez parametru, vrací implicitně `0`.

AST/Number.py

Třída představující číslo.

AST/PostfixExpression.py

Třída představující unární postfixovou operaci (viz. specifikace). Vybrání prvku z pole dle indexu je reprezentována samostatně.

AST/Program.py

Třída představující program zapsaný v jazyce HeroC. Obsahuje inicializaci globálních proměnných a seznam funkcí.

AST/SemanticException.py

Třída představující sémantickou chybu.

AST/String.py

Třída představující řetězec. Při volání metody `asm()` se řetězec vloží na zásobník a je navrácen ukazatel na první znak řetězce.

AST/SubscriptExpression.py

Třída představující výběr prvku z pole. Lze uvažovat zápis ve stylu `expr[subexpr]`, kde výraz `expr` vrací adresu na zásobníku a `subexpr` představuje index.

AST/Variable.py

Třída představující proměnnou. Obsahuje identifikátor proměnné, její typ a hodnotu. Typ může nabývat jedné z hodnot `ARRAY`, `LONG` nebo `POINTER`. Kromě mětody `asm()` obsahuje třída `Variable` metodu `asm_global` pro generování kódu globálních proměnných.

AST/WhileExpression.py

Třída představující Do-While výraz. Obsahuje testovanou podmínku a tělo cyklu.Tělo cyklu je vykonáno po testování podmínky.