HEROC – překladač jazyka vycházejícího z C

Vilém Vychodil

***Abstrakt*—Cílem tohoto souboru je specifikovat požadavky na zápočet a zkoušku z předmětu KMI/PAPR v letním semestru akademického roku 2015/2016. Jazyk, který bude implementován vychází z jazyka C a blíží se jeho podmnožině, ve které se vyskytuje jediný datový typ a to (šedesátičtyřbitový) `long`. Jazyk ale není podmnožinou C, obsahuje i některé rysy, které bychom v C nenašli a v některých detailech mají některé konstrukce oproti standardnímu C jinou sémantiku.**

I. Podmínky udělení zápočtu

Nejpozději v zápočtovém týdnu bude student schopen poslat prototyp implementace překladače, který bude splňovat minimální požadavky, které zahrnují:

- funkční lexikální analyzátor,
- funkční syntaktický analyzátor,
- funkční generátor druhé interní formy programu.

Zápočet bude udělen za předvedení funkce během cvičení, odevzdávat se bude zdrojový kód prototypu.

II. Získání zkoušky

Nejpozději do konce zkouškového období pro tento semestr je třeba odevzdat úplnou implementaci překladače nebo aspoň takovou verzi, která je autorem považována na úplnou. Nutnou podmínkou pro získání zkoušky je udělený zápočet. Úplná implementace musí obsahovat:

- zdrojové kódy překladače,
- programátorskou dokumentaci v PDF, ve které je zhruba popsáno, jak a kde je v překladači co implementováno (očekávaný rozsah 4–6 stran formátu A4).

Samotná zkouška proběhne formou diskuse nad zdrojovým kódem, student musí přijít na zkoušku připraven tak, aby byl schopen demonstrovat svůj překladač v běhu. To jest, buď jej rozjede na svém účtu na učebně nebo přinese notebook s potřebnou instalací. Během zkoušky bude student prozkoušen z teoretických partií předmětu s ohledem na to, jak vypadá jeho implementace. Výsledná známka je pak souhrnem znalostí a kvality a rozsahu implementace překladače.

III. Cílová platforma

Cílová platforma je assembler pro x86–64 v notaci GAS. Vygenerovaný kód musí obsahovat návěstí `main` a musí jít přeložit a sestavit do spustitelného souboru pomocí překladače gcc nebo yasm. Pro seznámení s cílovou platformou je doporučeno přečíst si následující jemný úvod a specifikaci SYSV AMD64 ABI, zejména sekci 3.2 o zásobníkových rámcích. Ve výsledku by

```
$ gcc -m64 -o pokus kod.s herocio.c && ./pokus
```

by mělo přeložit vygenerovaný kód (soubor `kod.s`) a spolu s implementací výstupních funkcí (soubor `herocio.c`, viz dále), uložit výsledek do spustitelného souboru `pokus` a při úspěchu jej spustit.

IV. Rozhraní překladače

Výsledný překladač (program) čte standardní vstup a výsledek překladu zapisuje na standardní výstup. Všechna chybová hlášení musí být zapisována na standardní chybový výstup s rozlišením o jaký typ chyby jde; minimálně je potřeba rozlišovat:

- syntaktické chyby,
- sémantické chyby.

Styl výpisu chybových hlášení je ponechán na řešiteli, z popisu chyby by ale mělo být patrné, co ji způsobilo a o jakou chybu se jedná.

Vstupem pro překladač je textový soubor (zamýšlený zdrojový kód) zapsaný v sedmibitovém ASCII. Překladač by tedy mělo být možné použít například takto:

```
$ cat zdrojak.heroc | prekladac >kod.s 2>err.txt
```

s tím, že zdrojový kód v `heroc` je obsažen v souboru `zdrojak.heroc`, `prekladac` je spustitelný soubor překladače, `kod.s` je jméno pro výsledný výstup překladu v assembleru a `err.txt` je soubor, který bude obsahovat chybová hlášené, pokud k nějakým dojde. V případě jakékoliv chyby (syntaktické nebo sémantické) není nutné aby překladač generoval kód, zotavení z chyb není třeba implementovat.

V. Lexikální prvky jazyka

Lexikální atomy jazyka HEROC jsou zjednodušenou podmnožinou atomů jazyka C, podrobný popis lze najít například v pracovní verzi standardu jazyka.

Na lexikální úrovni je potřeba rozlišovat literály celých čísel v rozsahu 64-bitového `long`u (se znaménkem). Čísla lze psát ve standardní desítkové, osmičkové a šestnáctkové notaci. Například tedy

```
64904
0xfd88
0176610
```

jsou tři různé zápisy (téže) číselné konstanty. Sedmibitové ASCII znaky lze psat v klasické C-notaci, stejně tak řetězcové literály:

```
'r'
"Hello, World!"
```

Případě znakových i řetězcových literálů je umožněno používat únikovou sekvenci začínající „\“, která může být následována jakýmkoliv znakem, přitom „n“ má význam „znak nový řádek“ (kód znaku 10), „t“ má význam „znak

horizontální tabulátor“ (kód znaku 9) a všechny ostatní znaky reprezentují sebe sama.

Klíčová slova jazyka jsou podmnožinou klíčových slov jazyka C, konkrétně je potřeba rozlišovat následující klíčová slova:

```
break continue do else for
if long return sizeof while
```

Identifikátory mají omezenou délku maximálně 80 znaků, mohou začínat písmenem nebo podtržítkem a dál mohou pokračovat písmeny, podtržítky i číslicemi a nesmějí se shodovat s žádným klíčovým slovem.

Uvažované operátory jsou následující:

```
! != % %= & && &= * *= + ++ += - -- -= / /=
: < << <<= <= = == > >= >> >>= ? ^ ^= | |= || ~
```

Další pomocné znaky:

```
; ( ) { } [ ]
```

Komentáře jsou ve stylu C, uvozené `/*` a končící nelbližším výskytem `*/`.

VI. Syntaxe jazyka

Jazyk je ze syntaktického hlediska zjednodušenou podmnožinou jazyka C, ve které existuje jediný datový typ `long` a používá se podmnožina příkazů. Příkazy se omezují na následující:

- Příkaz přířazení;
- Podmíněný výraz `if` s nepovinnou `else` větví;
- Cyklus typu `for`;
- Cyklus typu `while`;
- Cyklus typu `do`∼`while`;
- Příkaz výskoku z vnitřního cyklu `break`;
- Příkaz pokračování další iterací cyklu `continue`;
- Příkaz ukončení volání funkce `return`;
- Příkaz bloku.

Výrazy jazyka HEROC jsou stejné jako výrazy z jazyku C, nepoužívá se v nich žádné explicitní přetypování, priority a asociativita operátorů je stejná jako v jazyku C. Při definici funkce se u parametrů neuvádějí jména typů, všechny jsou implicitně `long`. Stejně tak se neuvádí typ výsledku, který je vždy `long`. Operátor `sizeof` lze používat pouze ve tvaru `sizeof (long)`. Přiřazení lze chápat i jako výraz, takže HEROC umožňuje používat zřetězená přiřazení ve stylu `x=y=10` a používat výsledek přiřazení v podmínce `if`-příkazu a podobně.

Pro zjednodušení budeme předpokládat, že větve podmíněného příkazu a cyklů musí být vždy bloky. Například

```
if (podminka)
  x = 10;
```

nelze, je potřeba uvést příkaz přiřazení v bloku:

```
if (podminka) {
  x = 10;
}
```

Deklarace je možné psát pouze na začátku bloku, nikam jinam. Jediným typem je `long`, takže všechny deklarace se píší ve tvaru `long jmeno,...` Součástí deklarace může být i inicializace. Všechny použité proměnné musí být deklarovány.

VII. Sémantika jazyka

Opět se jedná o zjednodušení sémantiky jazyka C. Pravdivostní hodnota nepravda je reprezentována číslem 0, vše ostatní se považuje za pravdu. Volání `return` bez argumentu vrací hodnotu 0. Jazyk má pouze jedinný jmenný prostor – v jednom okamžiku nemůže dané jméno označovat funkci i proměnnou. V rámci bloků se používá standardní překrývání platnosti významu jmen. Operátor `sizeof (long)` vrací konstantní hodnotu 8.

Operátor reference lze použít se jménem libovolné proměnné nebo funkce a výsledná hodnota je ukazatel na proměnnou nebo funkci reprezentovaný hodnotou `long`. To je *nejvýznamnější rozdíl* oproti C. Jakákoliv hodnota (typu long) může být použita pomocí dereference (operátor `*`) nebo indexu do pole (operátor `[]`) jako ukazatel. Následující výrazy jsou z pohledu HEROC ekvivalentní:

```
*ptr        ≡ ptr [0]
*(ptr + 8)  ≡ ptr [1]
*(ptr + 16) ≡ ptr [2]
```

Řetězcové konstanty jsou chápány jako pole prvků typu `long`, narozdíl od C je tedy na každý znak vyhrazeno 8 bytů, přistuponat k n-tému prvku pole je možné buď pomocí `str [n]` nebo `*(str + n * sizeof (long))` a podobně.

Oproti C obsahuje HEROC rozšíření týkající se nového typu výrazu. Inicializace pole psaná ve tvaru

$\{x, y, z, \ldots\}$

je chápána jako *výraz*, který *vrací ukazatel na alokované pole* (na zásobníku) obsahující dané prvky. HEROC tedy umožňuje například napsat příklaz přiřazení jehož R-hodnota obsahuje „výraz pole“:

```
foo = 10 + ({20, 30, 40} [2]);
```

Po jeho provedení bude mít `foo` hodnotu 50. HEROC nedisponuje *operátorem čárka*, to jest, v HEROC není přípustné psát

```
foo = (bar += 2, bar + baz)
```

a podobně.