# Analyzátor výrazů

Analyzátor výrazů (AV) je volán analyzátorem programu (AP), pokud je v zápisu programu očekáván výraz. AV tedy přebírá řízení překladu od AP a vrací mu jej až po úspěšném zpracování celého výrazu. AV kontroluje správnost zápisu výrazu pomocí precedenční syntaktické analýzy, jíž simuluje tvorbu derivačního stromu. Dále je během analýzy výrazu kontrolováno, zda není sémanticky špatně – špatná kombinace datových typů operandů, použití nedeklarovaného identifikátoru a použití funkce jako proměnné. Během činnosti jsou samozřejmě také generovány příslušné instrukce odpovídající operacím provedeným ve výraze.

Při předání řízení překladu programu AV je mu předána také informace o výsledném datovém typu. V takovém případě AV zajistí i přetypování výsledku instrukce do správného datového typu. Předaný datový typ může být také speciální hodnota NO\_DATATYPE. Toto použití AV slouží např. pro determinování datového typu proměnné auto.

Při zpracování příchozího lexému jsou převedeny všechny možné operandy (literály i proměnné) na terminál i, avšak je zachován příznak, zda se jedná o proměnnou, nebo literál. Lexémy, které nemůžou být ani operátorem ani operandem jsou převedeny na ukončující terminál $. Chyba vrácená lexikální analýzou je propagována zpět k AP. Problémem je zde ukončující závorka ‚)‘, která může být jak součást výrazu, tak ukončující terminál, proto si AV při načítání lexémů jednoduchých závorek ukládá aktuální hodnotu zanoření a podle toho považuje ukončující závorku buď za operátor, nebo za $. Návratovou hodnotou AV je lexém, jenž je ukončujícím.

Precedenční analýza je tvořena precedenční tabulkou v. příloha – tabulka pro precedenční analýzu fungující nad jednoduchým zásobníkem, jehož položka je buď terminál, nebo neterminál a v obou případech sebou nese informaci o datovém typu, jelikož neterminál se v průběhu analýzy stává operandem. Neterminál jakožto operand může být proměnná, konstanta, nebo mezivýsledek, což je pouze dočasná, abstraktní proměnná použitá během vyhodnocování výrazu. V případě proměnné (i dočasné) si sebou nese i informaci o jejím indexu, v případě konstanty onu hodnotu.

Operace mezi dvěma konstantami je pak zderivována opět na neterminál typu konstanta. Dá se na to pohlížet jako na optimalizaci, nicméně je to nutné z hlediska instrukční sady. Výraz a = 1 + 1; tedy vygeneruje jedinou instrukci, která uloží do proměnné a hodnotu 2, výraz a = 1 + 1 + a; pak pouze jedinou instrukci sečtení (2 + a, resp. a + 2), nicméně výraz a = a + 1 + 1; z povahy zásobníkového řešení vygeneruje 2 instrukce, nikoliv jednu ((a + 1) + 1).

Sémantika operací je implementována v 3 (aritmetické, relační a logické operátory) souborech \*.inc.c (sloužící jako víceřádkové makro), které jsou parametrizována konkrétním typem operace, jejím bajtkódem, dále tím zda je operace komutativní, jaký má (a jestli vůbec) neutrální a agresivní prvek a pro který prvek nemá definované chování (nula v děliteli). Tyto parametry slouží k dalším optimalizacím. Výraz a = a \* 1; tak vede na přiřazení hodnoty 1. Při dělení konstantou nulou pak nedochází ke generování instrukce DIV, ale instrukce ERROR.

Výpočet výrazu interpretem simuluje zásobníkové zpracování podobně jako instrukce instrukční sady x87 FPU, nicméně je to výhodné, protože stejným způsobem funguje analýza výrazu. Nevýhodou je nutnost před započnutím výpočtu první operand načíst do paměťového místa vyhrazeného pro výpočet výrazu první operace na každé úrovni priority (dle priorit operátorů), jelikož adresa prvního operandu je zároveň místo, kde se ukládá výsledek operace. Pro výraz (a+1)\*(b+1) tedy kromě 3 operací proběhnou i 2 načtení do dočasných proměnných. Pro výraz a+b+c+d by také proběhly 3 operace, avšak pouze jedno načtení.

|  |  |  |
| --- | --- | --- |
| a |  |  |
| a+1 |  |  |
| a+1 | b |  |
| a+1 | b+1 |  |
| a+1 | b+1 | c |
| a+1 | b+1 | c+1 |
| a+1 | (b+1)\*(c+1) | c+1 |
| (a+1)\*(b+1)\*(c+1) | (b+1)\*(c+1) | c+1 |

|  |
| --- |
| a |
| a+b |
| a+b+c |
| a+b+c+d |

Tabulky ilustrují průběh výpočtu v jednotlivých krocích pro 2 různé výrazy. Jeden řádek v tabulce je jeden krok a jeden sloupec jedna položka simulovaného zásobníku, tedy jedna dočasná proměnná. Modrá značí naplnění dočasné proměnné instrukcí LOAD, šedá nevyužité místo a oranžová výsledek, který je po dokončení výpočtu vždy na vrcholu zásobníku. Velikost zásobníku pro daný výraz je zjištěna během analýzy, při běhu tudíž tento imaginární zásobník představuje pouze vyhrazená část paměti výpočet jednoho konkrétního výrazu v rámci pro lokální proměnné prováděné funkce.