# Interpret

Interpret přímo provádí přeložený kód bez nutnosti kompilace. Je volán s těmito parametry: ukazatel do tabulky symbolů (funkce main při prvním zavolání), počet (celková velikost) rámce pro lokální proměnné (+ dočasné proměnné pro výrazy i návratové hodnoty funkcí) a odkaz do paměti pro navrácení návratové hodnoty. Lokální proměnné jsou tedy alokovány dynamicky na zásobníku a tato struktura umožňuje i rekurzivní volání funkcí, jelikož instrukce CALL je rekurzivní volání procedury interpretu s novými parametry. Tento rámec je implementačně pole obecného typu DOUBLEWORD (tedy pole dvouslov; int32\_t prakticky), což je výhodnější než typ unie z pohledu paměťové náročnosti, protože proměnná typu int/bool bude zabírat pouze 4 B. Zároveň s tímto polem je ještě vygenerováno další pole bajtů stejné velikosti pro příznaky inicializace těchto proměnných.

Odkaz do tabulky symbolů je použit pro zjištění odkazu na první instrukci volané funkce, kterou je vždy instrukce INIT, která se ihned začne vykonávat bez dekódování. Operand instrukce obsahuje velikost předávaných parametrů, které jsou poté kopírovány z adresy pro návratovou hodnotu, kde volající funkce uložila předávané argumenty a je u nich nastaven příznak inicializace. U všech ostatních je naopak příznak inicializace nastaven na 0. Ve skutečnosti je tato instrukce relikt z raných verzí interpretu a nyní již není potřeba, jelikož se předává odkaz do tabulku symbolů.

Volání vestavěných funkcí je jednodušší a řešeno namapováním těchto funkcí na celé číslo a to je operandem instrukce CALLBIN. Zejména zde není nutné přesouvat proměnné mezi rámci funkcí.

Protože ARL (aritmetické, relační, logické) instrukce jsou implementovány zásobníkem, není nutné nikdy kontrolovat u prvního operandu, zda je inicializovaný, ani tento příznak nastavovat, neboť to již zajišťuje instrukce LOAD.

Jelikož je operační znak uložen na 8 B je možné výhodně dekódování instrukce implementovat jako LUT tabulku, což by se mělo pozitivně projevit na rychlosti a spolu s alokacemi lokálních proměnných na zásobníku, a typovostí instrukcí zajistit rychlé vykonávání instrukcí. Protože je však množství instrukcí značné, je pro ulehčení práce i zmenšení rizika chyby vystavěn interpret z větší části pomocí maker[[1]](#footnote-1). Část ARL instrukcí pak zcela.

## Práce s řetězci

Datový typ string je implementován jako struktura Tstring, který de facto string Pascalovského typu, protože během všech operací se stringy známe jejich velikost. To umožňuje také používat funkce rodiny mem\*() namísto str\*() napříč projektem a opět by mělo vést k efektivnějšímu provádění programů během interpretace. Ve struktuře je také uložen počet referencí na daný řetězec a přiřazení mezi řetězců je řešeno inkrementaci/dekrementací počtu ukazatelů. Pokud je aktuální hodnota 1 a má se dekrementovat, dochází k uvolnění alokované paměti řetězcem. Interpret tedy neobsahuje žádnou centrální správu dynamické paměti a uvolňuje ji hned jak je to možné. Pro literály je tato hodnota 0 a inkrementace/dekrementace se u nich neprovádějí.

**typedef** struct **{** size\_t len**;** unsigned refs**;** char str**[];  
}** Tstring**;**

1. Pravým programátorem interpretu je tedy preprocesor jazyka C ☺ [↑](#footnote-ref-1)