Liveness File Format

თითოეული ფუნქციის თითოეული ხაზისთვის წერია in და out სეტები. ხაზების დათვლა იწყება ფუნქციის ლეიბლიდან (ანუ int-list, float-list და ეგენი არ შედის ხაზებში).

Bugs or features that are not operational

არ მაქვს იმპლემენტირებული global briggs allocation. -g ფლეგის დროს ვაკეთებ იგივეს რასაც -b გააკეთებდა.

Mips Functions calling implementation

სტანდარტული მიდგომისგან განსხვავებით, $s რეგისტრებს ინახვავს caller და არა callee, რადგან ასეთი შეთანხმებით, ჩემს შემთხვევაში, უფრო მარტივი იყო intra block allocation-ს იმპლემენტაცია. დანარჩენი ისევეა როგორც სტანდარტულში.

Callee გამოყოფს სტეკში ადგილს ცვლადებისთვის და $ra რეგისტრისთვის (ინახავს $ra რეგისტრს). შემდეგ return-მდე ამოაქვს $ra რეგისტრში შენახული მნშვნელობა, თუ საჭიროა $v0 რეგისტრში წერს დასაბრუნებელ მნიშნველობას, $sp-ს ზრდის და ასრულებს jr $ra ინსტრუქციას.

Caller ინახავს $s რეგისტრებში ამოტანილ ცვლადებს ისევ მეხსიერებაში, გამოუყოფს ადგილს პარამეტრებისთვის ($a რეგისტრებს არ ვიყენებ) **Callee**-სთვის და იძახებს მას. კონტროლის დაბრუნების შემდეგ კი $sp-ს ზრდის.

Class design and general code structure

კოდში ჩახედვას არ გირჩევთ

CodeGenerator აგვარებს კოდის გენერაციას, არ არის დამოკიდებული რეგისტრის ალოკაციის ალგორითმზე. გადაეცემა **AllocatorFactory** ტიპის ობიექტი. და სხვადასხვა **AllocatorFactory**-ის გადაცემისას აგენერირებს სხვადასხვა კოდს. ანუ იდეაში ისე უნდა ყოფილიყო, რომ Allocator ინტერფეისს გავუკეთებდი სხვადასხვა იმპლემენტაციებს, codegen უკვე მზად მექნებოდა და შეხება აღარ მომიწევდა, მაგრამ აღმოჩნდა რომ არც ისე კარგად მქონდა ინტერფეისი მოფიქრებული და შეცვლა მომიწია intra block ალოკაციის დამატებისას. Global briggs ალოკაცია რომ დამეწერა მაგისთვის კიდე შეცვლა მიწევდა. სხვა მგონი საინტერესო არაფერი არ არის.

Design of data structures used to implement each register allocation scheme: naïve, block, Briggs.

Naïve: Allocator.allocate() აბრუნებს MemoryTable კლასის ობიექტს, სადაც წინასწარ განსაზღვრულია რომელი ცვლადი რა ოფსეტზე წერია, რეგისტრებში არცერთი ცვლადი არ გვაქვს.

Intra block: Allocator.allocate() აბრუნებს MemoryTable კლასის ობიექტს, სადაც წინასწარ განსაზღვრულია რომელი ცვლადი რა ოფსეტზე წერია, თუმცა თითოეული basic block-ის შემდეგ ვიძახებთ **Allocator.reallocate()** მეთოდს, რომელიც რაღაც ცვლადებს რეგისტრებში აიტანს, და

რაღაცეებს მეხსიერებაში დააბრუნებს (ოღონდ, შესაბამის mips ინსტრუქციებს CodeGenerator აგენერირებს, ეს უბრალოდ ეტყვის CodeGenerator-ს რაები ამოიტანოს წინასწარ ამ basicblock-ისთვის და რაები დააბრუნოს).