LIR Data Structure

Sunday, April 21, 2024 11:04 PM

```
// type aliases for easier understanding
type StructId = string
type FuncId = string
type VarId = string
type FieldId = string
type BbId = string

Program
-   globals: VarId → Type
-   structs: StructId → (FieldId → Type)
-   externs: FuncId → Type
- functions: FuncId → Function

Function
-   name: FuncId
- params: vector<(VarId, Type)>
- rettyp: option<Type>
- locals: VarId → Type
-   body: BbId → BasicBlock

BasicBlock
- label: BbId
- insts: vector<LirInst>
-  term: Terminal

LirInst
|   Alloc { lhs: VarId, num: Operand }
|   Arith { lhs: VarId, aop: ArithmeticOp, left: Operand, right: Operand }
| CallExt { lhs: option<VarId>, callee: FuncId, args: vector<Operand> }
|     Cmp { lhs: VarId, aop: ComparisonOp, left: Operand, right: Operand }
|    Copy { lhs: VarId, op: Operand }
|     Gep { lhs: VarId, src: VarId, idx: Operand }
|     Gfp { lhs: VarId, src: VarId, field: string }
|    Load { lhs: VarId, src: VarId }
|   Store { dst: VarId, op: Operand }

Terminal
|       Branch { guard: Operand, tt: BbId, ff: BbId }
|   CallDirect { lhs: option<VarId>, callee: FuncId, args: vector<Operands>, next_bb: BbId }
| CallIndirect { lhs: option<VarId>, callee: VarId, args: vector<Operands>, next_bb: BbId }
|         Jump { next_bb: BbId }
|          Ret { op: option<Operand> }

Operand
| Const { num: int32_t }
|   Var { id: VarId }

ArithmeticOp
| Add
| Sub
| Mult
| Div

ComparisonOp
| Equal
| NotEq
| Lt
| Lte
```

x = $alloc op

x[42]

$gep = "get element pointer"

```
| Equal
| NotEq
| Lt
| Lte
| Gt
| Gte
```