Stage-5实验报告

吴垒 2020010916

###### 实验内容

明显比stage-4简单，可能因为完成stage-4需要对整个框架做修改，改完就完全理解了。

Step11：

在frontend/lexer/lex.py中添加L/RBracket用于匹配数组的中括号。

在frontend/ast/tree.py中添加IndexExpr作为数组节点，子属性base存储其Identifier，index同时完成对数组声明（各维度）的存储与对单个数组元素下标的存储。

在frontend/parser/ply\_parser.py修改p\_declaration添加可生成empty的非终结符arrayindex用于匹配数组定义的维度部分。添加p\_postfix\_index用于匹配表达式中的数组元素下标部分。同时将p\_binary\_expression的assignment产生式使其左值可匹配数组元素。

在frontend/type/array.py中为ArrayType添加getdim(k)方法获取其第k维度的深度，并直接修改multidim的第二个参数为list类型用于直接通过intliteral的list建立type。

在frontend/typecheck/namer.py中添加visitIndexExpr用于对表达式中数组的元素访问进行类型检查并匹配符号栈中的符号。在各个visit函数中将每个表达式的类型用setattr设置为其type属性，用于进行类型检查。修改visitDeclaration，依次判断声明的是数组还是变量、是否为全局变量、是否有初始值并进行访问，设置type、symbol。修改visitAssignment判断左值类型，并设置type。

在utils/tac/tacinstr.py中添加Alloc类表示为数组分配空间的tac指令。

在utils/tac/funcvisitor.py中添加visitAlloc用于根据数组size生成分配空间的tac指令。

在frontend/tacgen/tacgen.py中修改visitDeclaration，添加判断声明对象是否为数组并调用visitAlloc为数组分配空间。添加visitIndexExpr用于生成表达式中的数组元素对应的tac码；先根据symbol的dim用add、mul计算其基址偏移量，之后同时生成两个寄存器，一个用于存储其地址，用setattr设置为addr属性，一个用于存储其值，用setattr设置为val属性，分别针对该元素出现在左值及右值的情况，避免再添加特判。修改visitAssignment，分别判断其是左值数组元素还是标识符、是否为全局变量并设置其值。

在utils/riscv.py中添加Alloc的Riscv指令，用于传递tac码中分配地址的信息。

在backend/riscv/riscvasmemitter.py的generateGlobal函数中将原本bss段固定的保留内存长度4改为symbol.type.size，用于同时为标识符、数组分配内存空间。在riscvasmemitter.riscvinstrselector中添加visitAlloc将tac指令Alloc直接翻译为Riscv指令，由于涉及到栈空间的分配，此步暂时不翻译为实际RV指令，交由emitNative处理。在RiscvSubroutineEmitter中添加emitAllocStack方法用于在栈上分配数组，只需将当前offset与sp相加作为数组基址、并将nextLocalOffset加上数组空间即可。

在backend/reg/bruteregalloc.py的allocForLoc中添加对loc指令的判断，若指令类型为Riscv.Alloc，调用subemitter.emitAllocStack分配空间，否则直接emitNative即可。

Step2：

在frontend/parser/ply\_parser.py中修改p\_paramlist、p\_paramlist\_single在尾部添加非终结符paramindex表示可能存在的数组下标，用于匹配数组传参，并添加对paramindex的语法匹配；修改p\_declaraction为固定匹配未初始化的Identifier，p\_declaraction\_array匹配未初始化的数组声明，添加p\_declaration\_array\_init匹配初始化的数组声明，并对尾部非终结符arrayInit、其包含的integerList进行相应语法匹配。

在frontend/symbol/varsymbol.py中修改setInitValue使数组value可初始化为list。

在frontend/typecheck/namer.py中修改visitParameter完成对形参类型的设置，修改visitDeclaration完成对数组的初始化，需要判断是否为全局变量类型。对函数形参若为数组类型且其第一维为空，则将其第一维初始化为-1。

在frontend/typecheck/typer.py的visitCall中添加对函数参数列表的类型检查。

修改frontend/type/array.py中Arraytype的\_\_eq\_\_方法，当某一维为-1时说明为通配属性，可判定为该维相等。

在frontend/tacgen/tacgen.py中修改transform中对每个function的param的访问，判定其若为数组则应设置baseTemp属性而不是Temp。修改visitIdentifier判断其是否为Call的数组传参，若为数组则设置其val为首地址。修改visitDeclaration添加对数组的初始化tac码，用visitStoreTemp依次将初始化值装载。

在backend/riscv/riscvasmemitter.py的riscvasmemitter的generateGlobal方法中添加将完成初始化的全局数组放到data段，并将值依次用.word填入。

###### 思考题

Step11：可变长度数组

假设当建立动态数组时设该数组size存在t0，则需要添加Riscv的sw, t1, sp存储当前栈顶位置作为数组基址并添加 add sp, sp, -t0将数组空间压栈，这样就导致了sp在一个函数内的变化；然而在变化期间仍需要通过sp+offset访问局部变量，故需要进入某一函数后固定fp、将所有变量的访问方式改为通过fp + nextLocalOffset的方式访问，更换计算nextLocalOffset的方式，这将是一项比较麻烦的工程，因为stage9的函数栈空间分配我就想了很久没有想明白。

Step12：数组第一维为空的设计

由于数组任一元素的地址计算不依赖于该数组第一维的元素的值（如访问a[n1][n2]的元素a[i][j]偏移为i+n2\*j），因此不需要知道第一维即可实现数组内容的访问；可理解为第一维指向若干固定间隔的指针，而指针间隔的计算是依赖于之后的维度的；因此访问数组参数时第一维会直接衰减成数组指针，便于进行之后的代码分析与转换。在此增加对第一维大小的检查虽然合法，但只是凭空增加编译器的编写难度，是意义不大的。