Limbaje formale și tehnici de compilare

generarea de cod

Generarea de cod pentru AtomC se va realiza introducând regulile semantice în interiorul regulilor sintactice, ca și până acum. Pentru simplificarea acestei activități, generarea de cod este minimală, fiind implementate doar operațiile necesare pentru rularea exemplului din laborator.

Peste tot în regulile sintactice unde se face revenire la atomii sintactici precedenți (de exemplu înainte de *"return false;"* folosind *"iTk=startTk;"*), se va face revenire și în lista de instrucțiuni, pentru a se șterge posibilele instrucțiuni generate până atunci. Pentru aceasta, la începutul regulii sintactice putem insera *"Instr *startInstr=owner?lastInstr(owner->fn.instr):NULL;"*, iar lângă *"iTk=startTk;"* se va adăuga *"if(owner)delInstrAfter(startInstr);"*.

În versiunile noi ale fișierelor **vm.h** și **vm.c** instrucțiunile noi necesare mașinii virtuale sunt adăugate la sfârșitul celor vechi, în *enum* și în funcția *run*.

Atenție: în unele reguli semantice, s-au marcat prin comentarii C unde trebuie acestea să fie introduse, relativ la regulile analizei de domeniu și ale analizei de tipuri, care există deja în cod. De exemplu, dacă apare un comentariu /*...*/ într-o regulă semantică, înseamnă că această regulă trebuie introdusă în cod imediat înainte de linia descrisă de acel comentariu.

Altfel, dacă nu există aceste comentarii, regulile semantice de la generarea de cod se vor introduce după toate celelalte reguli care există deja în cod.

Pentru inserarea generării de cod în compilator, în *main* se inserează următorul cod între instrucțiunile specificate prin comentarii. În acest fel se respectă regula din C că u program începe cu execuția funcției *main*.

```
// parse(tokens);

Symbol *symMain=findSymbolInDomain(symTable,"main");
if(!symMain)err("missing main function");
Instr *entryCode=NULL;
addInstr(&entryCode,OP_CALL)->arg.instr=symMain->fn.instr;
addInstr(&entryCode,OP_HALT);
run(entryCode);

// dropDomain();
```

Tratarea valorilor stângi

Valorile stângi (**left-value** sau **lval**) sunt implementate ca fiind adrese de memorie (unde se poate stoca de exemplu rezultatul pentru o atribuire). Când se întâlnește un identificator (variabilă, parametru), se depune pe stivă adresa acestuia.

Adresa unui argument sau a unei variabile locale se depune pe stivă folosind "FPADDR".

Pentru a se extrage valoarea de la o adresă (operația de dereferențiere), se folosește instrucțiunea "LOAD", care preia din stivă adresa și pune în locul ei valoarea de la acea adresă.

Pentru a se depune o valoare la o adresă din stivă, se folosește instrucțiunea "STORE", care are pe stivă atât adresa cât și valoarea.

Funcția *addRVal*, dacă rezultatul curent este o valoare stângă, generează cod care preia din stivă adresa acesteia și depune în stivă valoarea de la acea adresă.

Testarea generării de cod

Pentru a se testa generarea de cod, se va folosi fișierul ”testgc.c”. Se va verifica afișarea valorilor din comentarii.