Μεταγλωττιστές 2019-2020
Tony Compiler
Αναφορά εργασίας

Μάμαλη Κατερίνα 03116017
Πετρακόπουλος Βασίλης 03116105

Γλώσσα - εργαλεία - βιβλιοθήκες

Ο compiler είναι γραμμένος σε OCaml και έχουν χρησιμοποιηθεί τα εργαλεία ocamllex και ocamlyacc για τη λεκτική και συντακτική ανάλυση αντίστοιχα.
Χρειάζεται να υπάρχουν τα παρακάτω προκειμένου να λειτουργεί:

- ocamlbuild
- ocamlfind
- Ocaml (version 4.08.0)
- nasm
- llvm (έχει χρησιμοποιηθεί version 6.0, αλλά ενδεχομένως παίζουν και μεταγενέστερες)
- clang (ομοίως version 6.0, αλλά ενδεχομένως παίζουν και μεταγενέστερες)
- llvm ocaml bindings
- libgc-dev (για τον garbage collector)

Επίσης, έχει χρησιμοποιηθεί η βιβλιοθήκη edsger_lib, που χρησιμοποιήθηκε και στις εργαστηριακές διαλέξεις του μαθήματος.

Bonus τμήματα

Έχουμε ενσωματώσει και δοκιμάσει τον garbage collector των Boehm-Demers-Weiser για τον οποίο θα χρειαστεί να έχει εγκατασταθεί κάποιο πακέτο, όπως το libgc-dev (για Ubuntu) που αναφέρεται παραπάνω.
Για την εφαρμογή των βελτιστοποιήσεων χρησιμοποιείται ο LLVM optimizer με παράμετρο -Oz:
opt -Oz [options] <input bitcode file>.

Λειτουργικό σύστημα

Ο compiler έχει δοκιμαστεί και λειτουργεί σε Ubuntu 18. Πιθανώς να παίζει και σε άλλα συστήματα. Σημειώνεται ότι σε μια δοκιμή σε Ubuntu 20 εμφανιζόταν κάποιο πρόβλημα κατά την κλήση των συναρτήσεων της edsger_lib που οδηγούσε σε segmentation fault, το οποίο δε λύθηκε και δε γνωρίζουμε αν ήταν πρόβλημα του συγκεκριμένου συστήματος μόνο ή αν υπάρχει γενικά κάποια ασυμβατότητα με Ubuntu 20.

Εγκατάσταση, μεταγλώττιση και εκτέλεση compiler

Για την εγκατάσταση και μεταγλώττιση του πηγαίου κώδικα του Compiler αρκεί η εκτέλεση του make όπως περιγράφεται παρακάτω. Η μεταγλώττιση προγραμμάτων Tony γίνεται όπως

περιγράφεται στην ενότητα 4 της εκφώνησης. Υπάρχει δυνατότητα εισαγωγής παραμέτρων -O, -f, -i με τη λειτουργικότητα που περιγράφεται στην ενότητα 4. Σημειώνουμε ότι για τη μεταγλώττιση ενός προγράμματος θα πρέπει να διαθέτουμε δικαίωμα διαγραφής στο φάκελο που γίνεται η μεταγλώττιση. Αυτό είναι απαραίτητο καθώς για την υλοποίηση των -f, -i που απαιτούν η έξοδος να γίνεται μόνο στο standard output χρειάζεται να διαγράφουμε το αρχείο a.ll που δημιουργείται κατά τη μεταγλώττιση. Γνωρίζουμε ότι αυτές οι λειτουργίες θα μπορούσαν να υλοποιηθούν ορθότερα με χρήση της συνάρτησης του llvm, llvm::cl::ParseCommandLineOptions() αλλά δεν καταφέραμε να την εφαρμόσουμε σε κώδικα Ocaml.

1. Σε οποιοδήποτε directory:
   `git clone https://github.com/MamaliKaterina/Compilers.git`
2. `cd Compilers/compiler`
3. Εφόσον έχουν εγκατασταθεί όλα όσα αναφέρονται παραπάνω: `make`
4. Παράδειγμα μεταγλώττισης ενός προγράμματος:
   `./tony ../tony_programs/hello_world.tony`
5. Παράγεται το εκτελέσιμο *a.out* τού προγράμματος, το οποίο είναι έτοιμο να τρέξει:
   `./hello_world.out`

Κώδικας

Ο φάκελος “compiler” είναι ουσιαστικά ο μόνος που χρειάζεται για τη μεταγλώττιση και εκτέλεση του compiler μας (και ο φάκελος “tony_programs” για δοκιμές με έτοιμα προγράμματα). Οι υπόλοιποι φάκελοι είναι κάποια μέρη του μεταγλωττιστή ή/και κάποιες δοκιμές που δημιουργήθηκαν στην πορεία υλοποίησης της εργασίας.

Στον φάκελο αυτό βρίσκονται τα παρακάτω αρχεία. Τα χωρίσουμε σε δύο κατηγορίες:

1. πηγαίος κώδικας του compiler,
2. βοηθητικά αρχεία για το make και την εκτέλεση της μεταγλώττισης.

Θα αναπτύξουμε σύντομα τη λειτουργία κάθε αρχείου και τη θέση του στη διαδικασία μεταγλώττισης.

1. Πηγαίος Κώδικας Μεταγλωττιστή

tony_lexer.mll/.mli:
Αυτοματοποιημένο εργαλείο για λεκτική ανάλυση του πηγαίου κώδικα της Tony. Αρκεί να ορίσουμε κατάλληλα τις λέξεις κλειδιά της γλώσσας καθώς και κάποιους κανόνες προτεραιότητας και προσεταιριστικότητας.

Helping_types.ml:
Ορίζουμε datatypes που θα χρησιμοποιηθούν από τον parser για την παραγωγή του ast. Είναι σημαντικό ότι ορίζουμε και τα βασικά datatypes που αντιστοιχούν στους βασικούς τύπους της γλώσσας Tony και αποτελούν τη βάση της σημασιολογικής ανάλυσης. Σε αυτό το πλαίσιο ορίζουμε διάφορες συναρτήσεις σύγκρισης και διαχείρισης (π.χ. εκτύπωσης) αυτών που αξιοποιούνται στα επόμενα κομμάτια.

tony_parser.mly:
Αυτοματοποιημένο εργαλείο για συντακτική ανάλυση του πηγαίου κώδικα της Tony. Γράφουμε τους κανόνες γραμματικής της γλώσσας και ο parser χρησιμοποιώντας και τα τερματικά σύμβολα που προκύπτουν από τον lexer παράγει το ast σύμφωνα με τα datatypes που ορίσαμε στο Helping_types.ml.

tony_symbol.ml:
Ορίζεται μια σειρά από συναρτήσεις που παράγουν το symbol table.
Επιπλέον, ορίζονται κάποιες δομές που παίζουν ρόλο των εγγραφών δραστηριοποίησης. Κάθε φορά που ορίζεται μια συνάρτηση δημιουργείται μια τέτοια δομή για τη διατήρηση πληροφοριών για την εμβέλεια, τις συναρτήσεις, τις μεταβλητές κλπ. Οι δομές αυτές διαγράφονται μετά τη μεταγλώττιση κάθε συνάρτησης-scope.
Τέλος, ορίζουμε μια δομή που περιλαμβάνει όλες τις μεταβλητές του προγράμματος, καταλληλα εμφωλευμένες σύμφωνα με τα scopes τους. Αυτή η δομή είναι πολύ σημαντική για την υλοποίηση της σωστής εμβέλειας των μεταβλητών σύμφωνα με τους κανόνες της Pascal. Για να την πετύχουμε, περνάμε σε κάθε συνάρτηση ως “κρυφή” παράμετρο μια δομή με δείκτες σε όλες τις μεταβλητές που είναι σε εξωτερικά blocks αλλά είναι ορατές από τη συνάρτηση άμεσα ή έμμεσα (αν είναι ορατές σε συνάρτηση που μπορεί να καλέσει η εν λόγω συνάρτηση και άρα θα

πρέπει να τους περνάει τις αντίστοιχες παραμέτρους, φαινόμενο που προκύπτει λόγω της αμοιβαίας αναδρομής). Τελικά, για να περάσουμε τις σωστές παραμέτρους χρειάζεται ο σημασιολογικός αναλυτής να παράγει αυτή τη δομή ώστε ο κώδικας παραγωγής ενδιάμεσου κώδικα (tony_llvm) να μπορεί να δημιουργήσει τα σωστά structs. Χωρίς αυτή τη δομή δε θα μπορούσαμε να ξέρουμε ποιες από τις **επόμενες** μεταβλητές θα πρέπει να περάσουμε σε μια συνάρτηση. Αυτές μπορεί να ορατές στην παρούσα συνάρτηση λόγω της αμοιβαίας αναδρομής.

tony_sem.ml:
Αναλαμβάνει τη σημασιολογική ανάλυση. Διασχίζει το ast που δημιουργήθηκε από τον parser και δημιουργεί ένα symbol table για τις μεταβλητές. Ελέγχει την ορθότητα των τύπων, της εμβέλειας των μεταβλητών, του ορθού ορισμού συναρτήσεων κλπ. Σε περίπτωση σφάλματος, δημιουργεί κατάλληλο TypeError με μήνυμα που περιλαμβάνει και τη γραμμή που βρέθηκε το σφάλμα.

tony_llvm.ml:
Παραγωγή του ενδιάμεσου κώδικα του llvm. Διασχίζει και πάλι το ast και για κάθε εντολή παράγει τις αντίστοιχες του ενδιάμεσου κώδικα. Μπορεί να πιάσει κάποια επιπλέον σφάλματα που δεν φαίνονται στην σημασιολογική ανάλυση και να δημιουργήσει κατάλληλες εξαιρέσεις.

Main.ml:
Πρόκειται για την κυρίως συνάρτηση του μεταγλωττιστή. Διαβάζει το αρχείο εισόδου και τρέχει όλα τα στάδια της μεταγλώττισης διαδοχικά. Επιπλέον αναλαμβάνει να πιάνει τις εξαιρέσεις και να εκτυπώνει κατάλληλα μηνύματα σφάλματος στον standard output.

Error.ml/.mli:
Αρχείο που πήραμε από τη σελίδα του μαθήματος και δεν μεταβάλλαμε. Ορίζονται διάφορα είδη σφαλμάτων (errors, warnings, etc) καθώς και συναρτήσεις για την εκτύπωση των μηνυμάτων που αυτά προκαλούν.

Extend.ml, Hashcons.ml/.mli, Identifier.ml/.mli:
Τεχνικά αρχεία που πήραμε από τη σελίδα του μαθήματος και δεν μεταβάλλαμε. Χρησιμεύουν στον ορισμό του hash table που χρησιμοποιείται για το symbol table.

2. Βοηθητικά αρχεία

Makefile:
Αναλαμβάνει τη μεταγλώττιση των αρχείων Ocaml που αποτελούν τον πηγαίο κώδικα του compiler.
Με make clean διαγράφονται όλα τα αρχεία .ll, .s, .out καθώς και το εκτελέσιμο του compiler Main.native.

tony:

Bash file που παίζει το ρόλο του μεταγλωττιστή. Για τη μεταγλώττιση ενός προγράμματος το τρέχουμε με παράμετρο το πρόγραμμα και παράγει το εκτελέσιμο αρχείο. Δέχετα παραμέτρους σύμφωνα με την ενότητα 4 της εκφώνησης.
Κατά τη μεταγλώττιση ενός αρχείου .tony, το μέρος του compiler που γράψαμε και που το εκτελέσιμό του θα βρίσκεται στο αρχείο Main.native μετά το make, μετατρέπει τον κώδικα tony σε llvm code. Στη συνέχεια, με το εργαλείο llc παράγουμε την assembly και το clang αναλαμβάνει να συνδέσει με τον κώδικα της βιβλιοθήκης και παράγει το τελικό εκτελέσιμο.

run_tests.sh:
Bash file που δημιουργήσαμε για τη μαζική εκτέλεση των test προγραμμάτων. Συγκεκριμένα, η εκτέλεσή του, χωρίς παραμέτρους, εκτελεί όλα τα προγράμματα .tony που βρίσκονται στους φακέλους ../tony_programs/testing_functionality, ../tony_programs/testing_error_catching.

libtony.a:
Ο κώδικας των συναρτήσεων βιβλιοθήκης που περιλαμβάνει η γλώσσα μας όπως τον πήραμε από τη βιβλιοθήκη του Αχιλλέα Μπενετόπουλου. Παραπάνω αναφέρεται ως edsger_lib.