Βελτιστοποιητικός μεταγλωττιστής από υποσύνολο της ANSI C σε MIPS32

Μάθημα: Προηγμένα Θέματα Θεωρητικής Πληροφορικής (CST319 / W10)
Διδάσκων: Νικόλαος Καββαδίας
nkavv@uop.gr
20/03/2010

Αντικείμενο της εργασίας

Αντικείμενο αυτής της εργασίας είναι η ανάπτυξη και σχεδιασμός μεταγλωττιστή από υποσύνολο της γλώσσας ANSI C σε κώδικα συμβολομεταφραστή για την αρχιτεκτονική MIPS32. Ο μεταγλωττιστής θα πρέπει να μπορεί να χρησιμοποιηθεί σε εκτελέσιμη μορφή από τη γραμμή εντολών του κατάλληλου τερματικού σε Cygwin/Windows και Linux. Ο μεταγλωττιστής θα πρέπει να σχεδιαστεί σύμφωνα με τις αρχές του δομημένου προγραμματισμού για την ευκολότερη συντήρηση και επέκταση του κώδικα.

Το υποσύνολο της ANSI C δίνεται στο Σχήμα 1 σε μορφή BNF (Backus-Naur Form). Σε σχέση με το πρότυπο της ANSI C, το θεωρούμενο υποσύνολο καθορίζεται σύμφωνα με τους παρακάτω περιορισμούς:

- τύποι δεδομένων: char, int (προσημασμένοι ακέραιοι των 8- και 32-bit)
- δεν υποστηρίζονται struct, union και δείκτες
- υποστηρίζονται πίνακες με μόνο μία διάσταση
- δεν υποστηρίζονται συναρτήσεις και κλήσεις συναρτήσεων εκτός και αν πρόκειται για προκαθορισμένες συναρτήσεις του συστήματος χρόνου εκτέλεσης. Κάθε πρόγραμμα ορίζεται από την προκαθορισμένη συνάρτηση `main`
- οι προκαθορισμένες συναρτήσεις που μπορεί να κληθούν σε μία εφαρμογή στο υποσύνολο της C είναι οι εξής: `scanc, scani, printc, printi.`

```
<program> ::= "main" "(" ")" "{" <globals> <statement_list> "}"
<statement> ::= "if" <paren_expr> <statement> |
            "if" <paren_expr> <statement> "else" <statement> |
            "for" "(" <paren_expr> ";" <paren_expr> ";" <paren_expr> ")" <statement> |
            "while" <paren_expr> <statement> |
            "do" <statement> "while" <paren_expr> ";" |
            "{" <statement_list> "}" |
            <expr> ";" |
            ";"
<globals> ::= <type_def> <var_def_list> ";" | <globals> <type_def> <var_def_list> ";"
<statement_list> ::= <statement> | <statement_list> <statement>
<item> ::= <int> | <char>
<item_list> ::= <item> | <item_list> "," <item>
<var_def> ::= <id> | <id> "=" <expr> | <id> "[" <int> "]" "=" "{" <item_list> "}"
<var_def_list> ::= <var_def> | <var_def> "," <var_def>
<type_def> ::= "int" | "char"
<sum> ::= <term> | <sum> <binop> <term>
<paren_expr> ::= "(" <expr> ")"
```

```
<expr> ::= <test> | <id> "=" <expr> | <unop> <expr> | <id> "[" <paren_expr> "]"
<test> ::= <sum> | <sum> <condop> <sum>
<sum> ::= <term> | <sum> <binop> <term>
<term> ::= <id> | <int> | <char> | <paren_expr>
<unop> ::= "+" | "-"
<binop> ::= "+" | "-" | "*" | "/" | "%"
<condop> ::= "==" | "!=" | "<=" | "<" | ">=" | ">"
<id> ::= <a legal C string>
<int> ::= <an unsigned decimal integer>
<char> ::= <an ASCII character>
```

Σχήμα 1: Γραμματική για το υποσύνολο της ANSI C.

Η ενδιάμεση αναπαράσταση του μεταγλωττιστή θα πρέπει να είναι αρχικά κάποιας μορφής αφηρημένο συντακτικό δένδρο (AST). Ως ενδεικτική υλοποίηση ενός AST και των βοηθητικών ρουτινών για τη διαχείρισή του αναφέρεται το [treeir]. Στη συνέχεια θα πρέπει να παράγεται γραμμικοποιημένος κώδικας ο οποίος θα χρησιμοποιεί εντολές της αρχιτεκτονικής MIPS32, αλλά θα διαθέτει απεριόριστους καταχωρητές. Οι καταχωρητές αυτοί θα έχουν τη μορφή $x<num> όπου <num> ακέραιος αριθμός μεγαλύτερος ή ίσος του 1. Στο επίπεδο της τελευταίας αυτής αναπαράστασης (MIPS32 linear IR ή MIPS32-LIR) θα εφαρμόζονται οι περισσότεροι μετασχηματισμοί όπως η επιλογή κώδικα (code selection) και ο καταμερισμός καταχωρητών (register allocation). Μετά την εφαρμογή και του καταμερισμού καταχωρητών, παράγεται ο τελικός κώδικας συμβολομεταφραστή για την αρχιτεκτονική MIPS32, στον οποίο χρησιμοποιούνται μόνο οι προβλεπόμενοι φυσικοί καταχωρητές που διαθέτει αυτή (από $0 ως $31).

Ο μεταγλωττιστής θα υποστηρίζει τις εξής βαθμωτές βελτιστοποιήσεις:

- διάδοση σταθεράς (επιλογή `-cnstprop`)
- δίπλωση σταθεράς (επιλογή `-cnstfold`)
- εξουδετέρωση νεκρού κώδικα (επιλογή `-dce`). Η βελτιστοποίηση DCE θα πρέπει να εφαρμόζεται μετά από κάθε πέρασμα βελτιστοποίησης
- εξουδετέρωση κοινής υποεκφράσεως (επιλογή `-cse`)

Για την επιλογή κώδικα να χρησιμοποιηθεί αλγοριθμική τεχνική η οποία να καλύπτει με βέλτιστο τρόπο τα δένδρα της ενδιάμεσης αναπαράστασης. Τέτοιες τεχνικές είναι τύπου tree covering με δυναμικό προγραμματισμό όπως για παράδειγμα η τεχνική BURS ή σε άλλη περίπτωση η τεχνική maximal munch ("μέγιστη μπουκιά"). Για την υλοποίηση της διαδικασίας της επιλογής κώδικα μπορεί να χρησιμοποιηθεί κάποιο γνωστό εργαλείο όπως είναι τα OLIVE, IBURG, και LBURG. Η αντίστοιχη επιλογή θα ονομάζεται `-inst-sel`.

Για τον καταμερισμό καταχωρητών θα πρέπει να υλοποιηθεί ο αλγόριθμος γραμμικής σάρωσης (linear-scan register allocation) των Poletto και Sarkar, ο οποίος αποτελεί καλό συμβιβασμό ανάμεσα στην ευκολία ανάπτυξης, την ταχύτητα εκτέλεσής του και τις επιδόσεις του τελικού προγράμματος συμβολομεταφραστή. Η αντίστοιχη επιλογή θα ονομαστεί `-linear-scan-regalloc`. Η απουσία αυτής της επιλογής, ή η εφαρμογή της `-no-regalloc`, θα αποτρέπει τον καταμερισμό καταχωρητών, με αποτέλεσμα όλες οι μεταβλητές να αποθηκεύονται και να φορτώνονται πάντα από τη μνήμη δεδομένων.

Επίσης η αρχιτεκτονική θεωρείται ότι δεν διαθέτει θυρίδες καθυστέρησης (delay slots). Η επιλογή -spim θα παράγει κώδικα συμβατό με τον spim, τοποθετώντας μία ψευδοεντολή nop μετά από κάθε εντολή αλλαγής ροής ελέγχου (jump ή branch). Επίσης, η επιλογή `-spim` θα αντικαθιστά τη χρήση των προκαθορισμένων συναρτήσεων read και write με τις συνήθεις

κλήσεις συστήματος (system calls) του διερμηνευτικού προσομοιωτή SPIM.
Προαιρετικά ο μεταγλωττιστής (π.χ. ως αντικείμενο διπλωματικής εργασίας) θα μπορούσε να επεκταθεί ως εξής:

- προσθήκη της δήλωσης switch-case
- ρύθμιση των παραμέτρων της αρχιτεκτονικής (π.χ. παραλληλία σε αριθμό ALU, πολλαπλασιαστών, ολισθητών κ.λ.π.)
- χρονοπρογραμματισμός κώδικα με τεχνική λίστας (list-based instruction scheduling)
- υποστήριξη εντολών του χρήστη για την επέκταση της αρχιτεκτονικής MIPS32
- εξαγωγή ενδιάμεσης αναπαράστασης τύπου στατικής ανάθεσης (SSA)

Βιβλιογραφικές αναφορές

[bison] Bison homepage. http://www.gnu.org/software/bison/bison.html

[flex] Flex (The Fast Lexical Analyzer) homepage. http://flex.sourceforge.net

[GCC] The GNU Compiler Collection homepage. http://gcc.gnu.org

[IBURG] IBURG, a tree parser generator. http://www.cs.princeton.edu/software/iburg/

[Ker90] Brian Kernighan and Dennis Ritchie, The C Programming Language, 2nd edition, Prentice Hall, 1990.

[Lev09] John Levine, Flex & Bison, O'Reilly Publishing, 2009.

[MIPS] MIPS Technologies Inc.. http://www.mips.com (Manuals για την αρχιτεκτονική MIPS32)

[Nie10] Tom Niemann, "A Compact Guide to Lex & Yacc."

http://www.epaperpress.com/lexandyacc/

[Pat04] D.A. Patterson and J.L. Hennessy, Computer Architecture: the Hardware-Software Interface Approach, 3rd edition, Morgan Kaufmann Publishers, San Fransisco, CA, 2004.

[treeir] Intermediate representation trees. http://code.google.com/p/drhanson/

Παράδοση και βαθμολόγηση της εργασίας

Στην εργασία του μαθήματος, ο φοιτητής καλείται να παραδώσει

- τον πηγαίο κώδικα του μεταγλωττιστή
- παραδείγματα δοκιμής της λειτουργίας του μεταγλωττιστή
- τεχνική αναφορά η οποία θα τεκμηριώνει τόσο τη διαδικασία της ανάπτυξης όσο και του σχεδιασμού του μεταγλωττιστή
- συνιστώμενες προδιαγραφές συστήματος του τελικού χρήστη για τη δημιουργία του μεταγλωττιστή με κτίσιμο (make) από τον πηγαίο κώδικα

Η εργασία παραδίδεται σε ηλεκτρονική μορφή (PDF της εργασίας + αρχεία κώδικα) στο email του διδάσκοντα. Οι φοιτητές μπορούν να παραδώσουν τις εργασίες τους το αργότερο μέχρι και την ημερομηνία διεξαγωγής των εξετάσεων περιόδου Ιουνίου-Ιουλίου 2010 για το μάθημα. Εργασία η οποία θα παραδοθεί μετά το πέρας αυτής της ημερομηνίας, θα βαθμολογηθεί ώστε να ληφθεί υπόψη για τις εξετάσεις της επόμενης περιόδου.

Μια εργασία βαθμολογείται με άριστα το δέκα (10), ο οποίος βαθμός και προστίθεται στο βαθμό της γραπτής εξέτασης πολλαπλασιασμένος με τον παράγοντα 0.3.

Η συγκεκριμένη προγραμματιστική εργασία είναι ατομική.