Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
http://courses.softlab.ntua.gr/pl2/

Γλώσσες Προγραμματισμού II

Οι ασκήσεις πρέπει να παραδοθούν στους διδάσκοντες σε ηλεκτρονική μορφή μέσω του συνεργατικού συστήματος ηλεκτρονικής μάθησης moodle.softlab.ntua.gr. Η προθεσμία παράδοσης θα τηρείται αυστηρά. Έχετε δικαίωμα να καθυστερήσετε το πολύ μία άσκηση.

Άσκηση 7 Εικονικές μηχανές

Προθεσμία παράδοσης: 13/7/2014

Υλοποιήστε μία εικονική μηχανή για τη γλώσσα στοίβας που περιγράφεται στην άσκηση 5, με τις αλλαγές που περιγράφονται παρακάτω. Σας συνιστούμε να χρησιμοποιήσετε τη C ως γλώσσα υλοποίησης του διερμηνέα και να εκμεταλλευτείτε τις επεκτάσεις του GNU C Compiler για την αποδοτική υλοποίηση VM interpreters που αναφέρονται στις διαφάνειες της διάλεξης της 26/2/2014.

Προδιαγραφές της εικονικής μηχανής.

Σε σχέση με τις εντολές της μηχανής στοίβας που περιγράφονται στην άσκηση 5, γίνονται οι εξής τροποποιήσεις που διευκολύνουν την αναπαράσταση των προγραμμάτων σε bytecode:

- Το πρόγραμμα συμβολίζεται από μία ακολουθία n λέξεων των 32 bit (bytecode). Η εκτέλεση ξεκινάει από την αρχή της ακολουθίας (IP=0) και ολοκληρώνεται όταν η εντολή που πρόκειται να εκτελεστεί είναι στη θέση IP=n (δηλαδή μία λέξη μετά την τελευταία εντολή του προγράμματος).
- Καταργούνται οι εντολές `true` και `false`. Οι λογικές τιμές αναπαριστώνται με ακέραιους αριθμούς, σύμφωνα με τη σύμβαση της C.
- Καταργούνται οι εντολές `nop`, `cond` και `loop`, οι οποίες μπορούν να υλοποιηθούν με τη χρήση των νέων εντολών που ακολουθούν.
- Προστίθενται δύο εντολές `jump` και `cjump`, που υλοποιούν άλματα, άνευ συνθήκης ή με συνθήκη, αντίστοιχα. Και οι δύο δέχονται μία ακέραια παράμετρο (θετική, αρνητική ή μηδέν) που δηλώνει την απόκλιση (displacement) του άλματος. Η απόκλιση αυτή προστίθεται στην τρέχουσα τιμή του IP, προτού αυτή αυξηθεί για την εκτέλεση της επόμενης εντολής. Άρα, `jump` με απόκλιση 0 ισοδυναμεί με την (καταργηθείσα) εντολή `nop` ενώ `jump` με απόκλιση -1 ισοδυναμεί με άπειρο βρόχο. Η εντολή `cjump` πραγματοποιεί το άλμα μόνο αν η τιμή στην κορυφή της στοίβας (η οποία αφαιρείται) είναι `true`.
- Προστίθενται δύο εντολές `get` και `put`, που υλοποιούν είσοδο και έξοδο (I/O) αντίστοιχα. Η πρώτη διαβάζει ένα χαρακτήρα από την τυπική είσοδο (stdin) και τοποθετεί τον ASCII κωδικό του στην κορυφή της στοίβας. Σε περίπτωση που η είσοδος έχει εξαντληθεί (EOF), στη στοίβα τοποθετείται ο αριθμός -1. Η δεύτερη αφαιρεί από την κορυφή της στοίβας μία τιμή, που πρέπει να αντιστοιχεί στον ASCII κωδικό ενός χαρακτήρα, και εκτυπώνει τον αντίστοιχο χαρακτήρα στην τυπική έξοδο (stdout). Ο χαρακτήρας πρέπει να εκτυπώνεται άμεσα: καλέστε `fflush(stdout)` στη C.

Κάθε λέξη του bytecode θα αντιστοιχεί σε μία εντολή του προγράμματος. Το LSB της εντολής θα είναι `0` αν πρόκειται για την εντολή αποθήκευσης μη αρνητικής ακέραιας σταθεράς στη στοίβα. Στην περίπτωση αυτή, τα υπόλοιπα bits της λέξης θα περιέχουν τη σταθερά. Για όλες τις υπόλοιπες εντολές, το LSB θα είναι `1` και τα τέσσερα επόμενα bits θα περιέχουν τον κωδικό εντολής (opcode), σύμφωνα με

την εξής αντιστοιχία: $0 \mapsto$ `+`, $1 \mapsto -$, $2 \mapsto$ `*`, $3 \mapsto$ `/`, $4 \mapsto$ `<`, $5 \mapsto$ `=`, $6 \mapsto$ `and`, $7 \mapsto$ `not`, $8 \mapsto$ `dup`, $9 \mapsto$ `pop`, $10 \mapsto$ `swap`, $11 \mapsto$ `swap2`, $12 \mapsto$ `cjump`, $13 \mapsto$ `jump`, $14 \mapsto$ `get`, $15 \mapsto$ `put`. Για τις εντολές `cjump` και `jump`, τα υπόλοιπα bits θα περιέχουν την απόκλιση, σε αναπαράσταση συμπληρώματος ως προς 2.

Είσοδος και έξοδος. Το πρόγραμμά σας θα δέχεται από τη γραμμή εντολών ακριβώς ένα όρισμα (`argv[1]`): το όνομα του αρχείου που περιέχει το πρόγραμμα που θα εκτελέσει η εικονική μηχανή. Οι λέξεις στο bytecode θα δίνονται σε αναπαράσταση big endian.

Κατά τη διάρκεια της εκτέλεσης αυτού του προγράμματος, η εικονική μηχανή πρέπει να διαβάζει από την τυπική είσοδο και να γράφει στην τυπική έξοδο. Θεωρήστε δεδομένο ότι η αναπαράσταση του προγράμματος σε bytecode θα είναι σωστή και ότι η εκτέλεση του προγράμματος δε θα οδηγεί σε σφάλμα (δε χρειάζεται να κάνετε run-time ελέγχους).

```
$ echo "000000680000001F000000640000001F000000140000001F" | xxd -r -p > fortytwo.sbc
$ xxd -ps -u fortytwo.sbc
000000680000001F000000640000001F000000140000001F
$ ./vm fortytwo.sbc
42
```

Πώς να ελέγξετε την εικονική μηχανή σας. Οι λύσεις σας θα βαθμολογηθούν με κριτήριο αφενός την ποιότητα του κώδικα, αφετέρου την απόδοσή του. Στη σελίδα του μαθήματος θα δοθούν λίγα benchmarks για να δοκιμάσετε την εικονική μηχανή σας. Γράψτε και δικά σας και μοιραστείτε τα με τους συναδέλφους σας στο Moodle.