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

Γλώσσες Προγραμματισμού II

Αν δεν αναφέρεται διαφορετικά, οι ασκήσεις πρέπει να παραδίδονται 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 6 Εικονικές μηχανές

Προθεσμία παράδοσης: 17/1/2018

Υλοποιήστε μία εικονική μηχανή για τη γλώσσα που περιγράφεται 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 29/11/2017.

Περιβάλλον εκτέλεσης. Η μηχανή διαθέτει μία στοίβα, αρχικά κενή, η οποία μπορεί να περιέχει ακέραιους αριθμούς 32 bit με πρόσημο.

Πρόγραμμα. Ένα πρόγραμμα αποτελείται από μία ακολουθία εντολών bytecode. Το μεγαλύτερο δυνατό πρόγραμμα αποτελείται από $2^{16} = 65536$ bytes εντολών. Οι θέσεις αυτών των bytes μέσα στο πρόγραμμα είναι αριθμημένες από 0 έως και $2^{16} - 1 = 65535$.

Οι εντολές είναι εν γένει μεταβλητού μήκους σε bytes. Κάθε εντολή αρχίζει με ένα op-code, μήκους ενός byte. Σε κάποιες εντολές, το op-code ακολουθείται από ένα ή περισσότερα bytes που περιγράφουν κάποιον ακέραιο αριθμό (με πρόσημο, σε αναπαράσταση συμπληρώματος ως προς δύο, ή χωρίς). Αν είναι περισσότερα του ενός, τα bytes του αριθμού δίνονται σε σειρά little-endian. Οι εντολές που υποστηρίζονται είναι οι ακόλουθες (σε παρένθεση το αντίστοιχο op-code σε δεκαεξαδική μορφή):

- `halt` (`0x00`): τερματίζει την εκτέλεση της μηχανής.
- `jump` (`0x01`): μεταπηδά στην εντολή της οποίας το op-code βρίσκεται στο byte του προγράμματος, η θέση του οποίου δίνεται από τον αριθμό που ακολουθεί (2 bytes, χωρίς πρόσημο).
- `jnz` (`0x02`): αφαιρεί το στοιχείο που βρίσκεται στην κορυφή της στοίβας και, αν δεν είναι μηδέν, μεταπηδά στην εντολή της οποίας το op-code βρίσκεται στο byte του προγράμματος, η θέση του οποίου δίνεται από τον αριθμό που ακολουθεί (2 bytes, χωρίς πρόσημο).
- `dup` (`0x04`): τοποθετεί στην στοίβα ένα αντίγραφο του στοιχείου που περιέχεται στη θέση i της στοίβας (με 0 συμβολίζεται η κορυφή, με 1 το στοιχείο αμέσως κάτω από την κορυφή, κ.ο.κ.), όπου i η τιμή του byte που ακολουθεί (χωρίς πρόσημο).
- `drop` (`0x04`): αφαιρεί και αγνοεί το στοιχείο που βρίσκεται στην κορυφή της στοίβας.
- `push4` (`0x05`): τοποθετεί στη στοίβα τον αριθμό που ακολουθεί (4 bytes, με πρόσημο).
- `push2` (`0x06`): τοποθετεί στη στοίβα τον αριθμό που ακολουθεί (2 bytes, με πρόσημο).
- `push1` (`0x07`): τοποθετεί στη στοίβα τον αριθμό που ακολουθεί (1 bytes, με πρόσημο).
- `add` (`0x08`): αφαιρεί κατά σειρά από τη στοίβα δύο στοιχεία, a και b, και τοποθετεί στην κορυφή το άθροισμα $a + b$.
- `sub` (`0x09`): ομοίως με την `add` αλλά για τη διαφορά $a - b$.

- `mul` (`0x0a`): ομοίως με την `add` αλλά για το γινόμενο $a * b$.
- `div` (`0x0b`): ομοίως με την `add` αλλά για το πηλίκο της ακέραιας διαίρεσης a/b.
- `mod` (`0x0c`): ομοίως με την `add` αλλά για το υπόλοιπο της ακέραιας διαίρεσης $a \% b$.
- `eq` (`0x0d`): αφαιρεί κατά σειρά από τη στοίβα δύο στοιχεία, a και b, και τοποθετεί στην κορυφή τον αριθμό 1 αν $a = b$, διαφορετικά τον αριθμό 0.
- `ne` (`0x0e`): ομοίως με την `eq` αλλά για τη συνθήκη $a \neq b$.
- `lt` (`0x0f`): ομοίως με την `eq` αλλά για τη συνθήκη $a < b$.
- `gt` (`0x10`): ομοίως με την `eq` αλλά για τη συνθήκη $a > b$.
- `le` (`0x11`): ομοίως με την `eq` αλλά για τη συνθήκη $a \leq b$.
- `ge` (`0x12`): ομοίως με την `eq` αλλά για τη συνθήκη $a \geq b$.
- `not` (`0x13`): αφαιρεί το στοιχείο που βρίσκεται στην κορυφή της στοίβας και τοποθετεί στην κορυφή τον αριθμό 1 αν αυτό ήταν μηδέν, διαφορετικά τον αριθμό 0.
- `and` (`0x14`): αφαιρεί κατά σειρά από τη στοίβα δύο στοιχεία και τοποθετεί στην κορυφή τον αριθμό 1 αν και τα δύο ήταν μη μηδενικά, διαφορετικά τον αριθμό 0.
- `or` (`0x15`): αφαιρεί κατά σειρά από τη στοίβα δύο στοιχεία και τοποθετεί στην κορυφή τον αριθμό 0 αν και τα δύο ήταν μηδενικά, διαφορετικά τον αριθμό 1.
- `input` (`0x16`): διαβάζει έναν χαρακτήρα input και τοποθετεί στην κορυφή της στοίβας τον ASCII κωδικό του.
- `output` (`0x17`): αφαιρεί το στοιχείο που βρίσκεται στην κορυφή της στοίβας και εκτυπώνει τον χαρακτήρα που αντιστοιχεί σε αυτόν τον ASCII κωδικό.
- `clock` (`0x2a`): εκτυπώνει στο standard output τον αριθμό των δευτερολέπτων που έχουν περάσει από την έναρξη της εκτέλεσης του προγράμματος, με έξι δεκαδικά ψηφία (format `"%0.6lf\n"`).

Η εκτέλεση του προγράμματος τερματίζεται είτε με την εντολή `halt` είτε όταν εξαντληθούν οι εντολές προς εκτέλεση. Σε περίπτωση άκυρων εντολών ή σφαλμάτων εκτέλεσης (άκυρο op-code, αριθμητική υπερχείλιση, διαίρεση με το μηδέν, άλμα σε μη υπαρκτή εντολή, κ.λπ.) η μηχανή σας είναι ελεύθερη να συμπεριφέρεται όπως θέλετε (undefined behavior).

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.

Κατά τη διάρκεια της εκτέλεσης αυτού του προγράμματος, η εικονική μηχανή πρέπει να διαβάζει (`input`) από την τυπική είσοδο και να εκτυπώνει (`output`) στην τυπική έξοδο.

Παράδειγμα εκτέλεσης. Οι εντολές `echo`, `base64` και `zcat` είναι εντολές κάποιου λειτουργικού συστήματος που σέβεται τον εαυτό του (π.χ., Linux).

```
$ base64 -D << ___EOF___ | zcat > hw.b
H4sICOa7OVoAA3Rlc3QuYgCNj7sOglAQRHcRM4HEwuq22PIJJFbGR/QriBJfCAQhhlha+L+2FjgSpdCY
mFvs487OntWFQOBigDUSbLFHjhQneCgYE8Yjux4yHFBixyxoFDnfuCPWRGydi9GhSNe5XqyX2Qpxa5Qy
jxFhRnlJeUV5LOjDN1CHzbPYcI3fzPes+g1UNR4RVz/XZXT6jZgw27IO2wmPuiXrgr+f0HdS1KS4yX/H
FyQJsfnyGamtUzUa6ANY9UiJRwEAAA==
___EOF___

$ ./vm hw.b
Hello world!
*****************
0.000042
```