



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 4 Εικονικές μηχανές

Προθεσμία παράδοσης: 9/1/2019

Υλοποιήστε μία εικονική μηχανή για τη γλώσσα που περιγράφεται παρακάτω.

Σας συνιστούμε να χρησιμοποιήσετε τη C/C++ ως γλώσσα υλοποίησης του διεργαστή και να εκμεταλλευτείτε τις επεκτάσεις του GNU C Compiler για την αποδοτική υλοποίηση VM interpreters που αναφέρονται στις διαφάνειες της διάλεξης της 24/10/2018.

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 (0x03): 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).
- swap (0x04): αντιμεταθέτει το στοιχείο που περιέχεται στην κορυφή της στοίβας με αυτό 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 (0x05): αφαιρεί και αγνοεί το στοιχείο που βρίσκεται στην κορυφή της στοίβας.
- push4 (0x06), push2 (0x07) και push1 (0x08): τοποθετούν στη στοίβα τον αριθμό που ακολουθεί (1, 2 ή 4 bytes, αντίστοιχα, με πρόσημο).
- add (0x09): αφαιρεί κατά σειρά από τη στοίβα δύο στοιχεία, b και a , και τοποθετεί στην κορυφή το άθροισμα $a + b$.

- `sub (0x0a)`: ομοίως με την `add` αλλά για τη διαφορά $a - b$.
- `mul (0x0b)`: ομοίως με την `add` αλλά για το γινόμενο $a * b$.
- `div (0x0c)`: ομοίως με την `add` αλλά για το πηλίκο της ακέραιας διαίρεσης a/b .
- `mod (0x0d)`: ομοίως με την `add` αλλά για το υπόλοιπο της ακέραιας διαίρεσης $a \% b$.
- `eq (0x0e)`: αφαιρεί κατά σειρά από τη στοίβα δύο στοιχεία, b και a , και τοποθετεί στην κορυφή τον αριθμό 1 αν $a = b$, διαφορετικά τον αριθμό 0.
- `ne (0x0f)`: ομοίως με την `eq` αλλά για τη συνθήκη $a \neq b$.
- `lt (0x10)`: ομοίως με την `eq` αλλά για τη συνθήκη $a < b$.
- `gt (0x11)`: ομοίως με την `eq` αλλά για τη συνθήκη $a > b$.
- `le (0x12)`: ομοίως με την `eq` αλλά για τη συνθήκη $a \leq b$.
- `ge (0x13)`: ομοίως με την `eq` αλλά για τη συνθήκη $a \geq b$.
- `not (0x14)`: 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 (0x15)`: 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 (0x16)`: 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 (0x17)`: διαβάζει έναν χαρακτήρα `input` και τοποθετεί στην κορυφή της στοίβας τον ASCII κωδικό του.
- `output (0x18)`: 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, αναφορά σε θέση που δεν υπάρχει στη στοίβα, 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 > test.b
H4sICAhmBlwAA3R1c3QuYgCNj80Kg1AQhwcM4+QiaHeXtvURhFbRD/UU1KwqZgS0rJF79u2hZ2kXBRB
3MX83DNnvtG1Q0BgiA0SRNgjR4oTXBSMCeORXRcZDiixY+Y3ipxv0hFrKrYux0hIpNu7XqyX2Rpxa5Qy
jxFiTn1JeUV5LBJAM1CHzbPYcIzXzPet+g1UNR4hVz/XZXT6jZgwi1gH7YRL3Yp1wd9P6DspalLc5L/j
C5IE2H75jNXWmRr19QGi+W0ARwEAAA==
__EOF__

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