Σειρά Εργασιών 4

Στάδιο Α: Διερμηνέας για μια απλή γλώσσα προγραμματισμού

Υλοποιήστε έναν διερμηνέα για προγράμματα που γράφονται με την παρακάτω σύνταξη:

Σύνταξη	**Σημασία** (σε συμβολικό κώδικα)
`Program = Tag {InstructionLine}.`	
`InstructionLine = [Label] Instruction.`	
`Instruction = LOAD Var GlobalVar` |	`Var = GlobalVar`
`STORE GlobalVar VarVal` |	`GlobalVar = VarVal`
`SET Var VarVal` |	`Var = VarVal`
`ADD Var VarVal1 VarVal2` |	`Var = VarVal1 + VarVal2`
`SUB Var VarVal1 VarVal2` |	`Var = VarVal1 - VarVal2`
`MUL Var VarVal1 VarVal2` |	`Var = VarVal1 * VarVal2`
`DIV Var VarVal1 VarVal2` |	`Var = VarVal1 / VarVal2`
`MOD Var VarVal1 VarVal2` |	`Var = VarVal1 % VarVal2`
`BRGT VarVal1 VarVal2 Label` |	`if (VarVal1 > VarVal2) goto Label`
`BRGE VarVal1 VarVal2 Label` |	`if (VarVal1 >= VarVal2) goto Label`
`BRLT VarVal1 VarVal2 Label` |	`if (VarVal1 < VarVal2) goto Label`
`BRLE VarVal1 VarVal2 Label` |	`if (VarVal1 <= VarVal2) goto Label`
`BREQ VarVal1 VarVal2 Label` |	`if (VarVal1 == VarVal2) goto Label`
`BRA Label` |	`goto Label`
`DOWN GlobalVar` |	`semaphore_down(GlobalVar)`
`UP GlobalVar` |	`semaphore_up(GlobalVar)`
`SLEEP VarVal` |	`sleep(VarVal)`
`PRINT String {VarVal}` |	`print(id,String,{VarVal}}`
`RETURN.`	`return()`
`Tag = #PROGRAM`	
`VarVal = Var | IntVal.`	
`Var = VarName | Varname [ VarVal ].`	
`VarName = $ Letter {Letter | Digit}.`	
`Label = L {Letter | Digit}.`	
`IntVal = [-] Digit {Digit}.`	
`String = " {Character} ".`	

Ο κώδικας ενός προγράμματος δίνεται σε ASCII, με κάθε εντολή σε ξεχωριστή γραμμή και τα μέρη μιας εντολής να χωρίζονται με έναν ή περισσότερους κενούς χαρακτήρες. Δεν υποστηρίζονται συναρτήσεις.

Ένα πρόγραμμα μπορεί να χρησιμοποιεί τοπικές καθώς και καθολικές μεταβλητές. Ο τύπος όλων των μεταβλητών είναι int. Οι μεταβλητές δηλώνονται αυτόματα, μέσω των εντολών που αναφέρονται σε αυτές (δεν υπάρχουν ξεχωριστές εντολές για την ρητή δήλωση μεταβλητών). Επίσης υποστηρίζονται πίνακες ακεραίων, το μέγεθος των οποίων καθορίζεται αυτόματα μέσω των εντολών που προσπελάζουν τα επιμέρους στοιχεία τους.

Οι εντολές που αφορούν τις βασικές λειτουργίες αριθμητικής και ελέγχου/αλμάτων εφαρμόζονται αποκλειστικά σε τοπικές μεταβλητές (βλέπε κανόνες/σημασία σύνταξης). Οι καθολικές μεταβλητές χρησιμεύουν για αποθήκευση δεδομένων και προσπελάζονται μέσω των εντολών `LOAD`/`STORE`. Επίσης, μέσω των εντολών `DOWN`/`UP`, μια καθολική μεταβλητή μπορεί να χρησιμοποιηθεί από το πρόγραμμα και ως ένας γενικός σηματοφόρος (βλέπε και επόμενο στάδιο). Σε αυτό το στάδιο, αυτές οι εντολές μπορεί απλά να αγνοούνται από τον διερμηνέα.

Ένα πρόγραμμα μπορεί να δεχτεί έναν αριθμό ορισμάτων (ακέραιων τιμών) που προσπελάζονται μέσω της προκαθορισμένης μεταβλητής `$argv[]`, ενώ ο αριθμός των ορισμάτων που περνιούνται στο πρόγραμμα αποθηκεύεται στην προκαθορισμένη μεταβλητή `$argc`. Κατά σύμβαση, στο πρώτο στοιχείο του πίνακα ορισμάτων `$argv[0]` αποθηκεύεται ένα μοναδικό αναγνωριστικό με βάση το οποίο μπορεί να γίνει αναφορά στην συγκεκριμένη εκτέλεση του προγράμματος (βλέπε και επόμενο στάδιο).

Το περιβάλλον εκτέλεσης πρέπει να δέχεται τον κώδικα ενός προγράμματος μέσω αρχείου, το όνομα του οποίου δίνεται από τον χρήστη, μαζί με τα ορίσματα του προγράμματος. Στην συνέχεια, πρέπει να αρχικοποιεί κατάλληλα τις μεταβλητές `$argv[]` και `$argc`, και να καλεί τον διερμηνέα που αναλαμβάνει να εκτελέσει το πρόγραμμα, εντολή προς εντολή, ελέγχοντας παράλληλα την συντακτική ορθότητα κάθε εντολής (προαιρετικά, προσπαθήστε να αποφύγετε περιττούς συντακτικούς ελέγχους για εντολές που έχουν ήδη ελεγχθεί μια φορά). Σε περίπτωση συντακτικού λάθους, πρέπει να εκτυπώνεται κατάλληλο μήνυμα και να τερματίζεται η εκτέλεση του προγράμματος.

Στάδιο Β: Περιβάλλον ταυτόχρονης εκτέλεσης

Επεκτείνετε το περιβάλλον εκτέλεσης έτσι ώστε να υποστηρίζει την ταυτόχρονη εκτέλεση προγραμμάτων γραμμένων στην παραπάνω γλώσσα. Σε αυτή την περίπτωση, οι καθολικές μεταβλητές είναι κοινές ανάμεσα σε όλα τα προγράμματα, και αυτές που χρησιμοποιούνται ως σηματοφόροι μπορεί να διευκολύνουν τον συγχρονισμό μεταξύ των προγραμμάτων.

Το περιβάλλον εκτέλεσης πρέπει, μέσω κατάλληλης εντολής, να επιτρέπει στον χρήστη να αρχίσει την εκτέλεση ενός προγράμματος, ενώ πιθανώς να εκτελούνται παρασκηνιακά ήδη κάποια προγράμματα. Επίσης, μέσω κατάλληλων εντολών, ο χρήστης πρέπει να μπορεί να βλέπει την κατάσταση εκτέλεσης των προγραμμάτων (μοναδικό αναγνωριστικό εκτέλεσης του προγράμματος, όνομα του αρχείου με τον κώδικα του προγράμματος, κατάσταση εκτέλεσης του προγράμματος), καθώς και να τερματίζει την εκτέλεση ενός προγράμματος. Να σημειωθεί πως το ίδιο πρόγραμμα μπορεί να εκτελείται ταυτόχρονα πολλές φορές, με τα ίδια ή με διαφορετικά ορίσματα (κάθε εκτέλεση λαμβάνει διαφορετικό αναγνωριστικό).

Η εκτέλεση κάθε προγράμματος γίνεται μέσω του διερμηνέα που αναπτύξατε στο προηγούμενο στάδιο. Επιπλέον, πρέπει να υλοποιηθεί αυτόματη εναλλαγή ανάμεσα στα προγράμματα, υπό τον έλεγχο του περιβάλλοντος εκτέλεσης. Αυτό θα πρέπει να υλοποιηθεί σταδιακά, όπως περιγράφεται παρακάτω.

Σε πρώτη φάση, υλοποιήστε την εκτέλεση των προγραμμάτων με ένα μοναδικό νήμα «εργάτη». Οι εντολές `DOWN/UP` θα πρέπει να υλοποιηθούν έτσι ώστε να επιτυγχάνεται η λειτουργικότητα γενικών σηματοφόρων, χωρίς να χρησιμοποιούνται λειτουργίες συγχρονισμού σε επίπεδο συστήματος. Η εναλλαγή ανάμεσα στα προγράμματα μπορεί να γίνεται ανά συγκεκριμένο αριθμό εντολών, με βάση τον πραγματικό χρόνο ή/και κάθε φορά που εκτελείται μια από τις εντολές `DOWN`, `UP`, `SLEEP` και `RETURN`.

Σε δεύτερη φάση, υλοποιήστε την εκτέλεση των προγραμμάτων με N νήματα «εργάτες». Η τιμή του N δίνεται κατά την εκκίνηση του περιβάλλοντος εκτέλεσης, και τα προγράμματα προς εκτέλεση πρέπει να ισοκατανέμονται στους διαθέσιμους «εργάτες». Επίσης, οι εντολές πάνω στις (κοινές) καθολικές μεταβλητές `LOAD/STORE/DOWN/UP` πρέπει να λειτουργούν σωστά ακόμα και όταν αυτές καλούνται από προγράμματα που εκτελούνται από ξεχωριστά νήματα «εργάτες». Τέλος, προαιρετικά, αλλάξτε την υλοποίηση σας έτσι ώστε ο αριθμός των νημάτων να αυξομειώνεται δυναμικά, με στόχο να μην υπάρχουν αδρανείς «εργάτες» που δεν εκτελούν κάποιο πρόγραμμα.

Δοκιμάστε το περιβάλλον εκτέλεσης γράφοντας προγράμματα που λύνουν κάποια από τα κλασικά προβλήματα συγχρονισμού με σηματοφόρους. Επίσης, στο site του μαθήματος θα βρείτε κάποια απλά προγράμματα που μπορεί να χρησιμοποιήσετε για δοκιμές στο πρώτο και δεύτερο στάδιο ανάπτυξης.

Σημείωση: Η υλοποίηση μπορεί να γίνει σε C με την βιβλιοθήκη pthreads, χρησιμοποιώντας mutexes και conditions, στο πνεύμα ενός ελεγκτή (monitor). Εναλλακτικά, μπορείτε να γράψετε την υλοποίηση σας σε Java, χρησιμοποιώντας τις λειτουργίες συγχρονισμού της γλώσσας synchronized, wait και notify.