Σειρά Εργασιών 4

4.1 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(thread 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 επιμέρους τμήματα μιας εντολής να χωρίζονται με έναν ή περισσότερους κενούς χαρακτήρες. Δεν υποστηρίζονται συναρτήσεις.

Ένα πρόγραμμα μπορεί να έχει τοπικές και καθολικές μεταβλητές, τύπου int (δεν υποστηρίζεται άλλος τύπος). Οι μεταβλητές δηλώνονται αυτόματα, μέσω των εντολών που αναφέρονται σε αυτές (δεν υπάρχουν εντολές για ξεχωριστή/ρητή δήλωση μεταβλητών). Επίσης υποστηρίζονται πίνακες ακεραίων, το μέγεθος των οποίων καθορίζεται αυτόματα και δυναμικά μέσω των εντολών που προσπελάζουν τα 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 την αποθήκευση δεδομένων και προσπελάζονται μόνο μέσω των εντολών `LOAD`/`STORE`. Επίσης, μια καθολική μεταβλητή μπορεί να χρησιμοποιηθεί από το πρόγραμμα και ως ένας γενικός σηματοφόρος μέσω των εντολών `DOWN`/`UP`. Σε αυτό το στάδιο, οι εντολές αυτές μπορεί να ερμηνεύονται απλά ως πράξεις αυξομείωσης ενός ακεραίου.

Ένα πρόγραμμα δέχεται έναν αριθμό ορισμάτων (ακέραιων τιμών) που προσπελάζονται μέσω της προκαθορισμένης μεταβλητής-πίνακα `$argv[]`, ενώ ο αριθμός των ορισμάτων που περνιούνται στο πρόγραμμα αποθηκεύεται στην προκαθορισμένη μεταβλητή `$argc`. Κατά σύμβαση, στο πρώτο στοιχείο του πίνακα ορισμάτων `$argv[0]` αποθηκεύεται αυτόματα από το περιβάλλον εκτέλεσης ένα μοναδικό αναγνωριστικό για το νήμα εφαρμογής που εκτελεί τον κώδικα του προγράμματος.

Ο χρήστης δίνει το όνομα του αρχείου και τα ορίσματα του προγράμματος. Το περιβάλλον εκτέλεσης αρχικοποιεί κατάλληλα τις μεταβλητές `$argv[]` και `$argc`, και αρχίζει την εκτέλεση του κώδικα, εντολή προς εντολή, ελέγχοντας παράλληλα την συντακτική ορθότητα κάθε εντολής (προαιρετικά, σκεφτείτε πως μπορείτε να κάνετε την εκτέλεση πιο αποδοτική, π.χ. αποφεύγοντας συντακτικούς ελέγχους για εντολές που έχουν ήδη ελεγχθεί μια φορά ή/και αποφεύγοντας την αναζήτηση ετικετών στα άλματα). Σε περίπτωση συντακτικού λάθους, πρέπει να εκτυπώνεται μήνυμα και να τερματίζεται η εκτέλεση του προγράμματος.

4.2 Περιβάλλον ταυτόχρονης εκτέλεσης

Επεκτείνετε το περιβάλλον εκτέλεσης έτσι ώστε να υποστηρίζεται η ταυτόχρονη εκτέλεση προγραμμάτων, δηλαδή ο χρήστης να μπορεί να αρχίσει την εκτέλεση ενός προγράμματος μέσα από ένα νέο νήμα εφαρμογής, την στιγμή που ήδη τρέχουν άλλα τέτοια νήματα. Το ίδιο πρόγραμμα μπορεί να εκτελείται ταυτόχρονα πολλές φορές (με τα ίδια ή με διαφορετικά ορίσματα) μέσα από ξεχωριστά νήματα εφαρμογής. Επίσης, πρέπει να παρέχονται εντολές μέσω των οποίων ο χρήστης μπορεί να βλέπει την κατάσταση εκτέλεσης των νημάτων εφαρμογής (αναγνωριστικό νήματος εφαρμογή, όνομα του αρχείου με τον κώδικα του προγράμματος που εκτελεί το νήμα, κατάσταση εκτέλεσης του νήματος), και μπορεί να τερματίσει την εκτέλεση ενός νήματος εφαρμογής.

Στην περίπτωση που τα νήματα εφαρμογής εκτελούν κώδικα που αναφέρεται σε μεταβλητές με το ίδιο όνομα, αυτές οι μεταβλητές είναι αυτομάτως κοινές ανάμεσα τους (δηλαδή η εντολή `LOAD` διαβάζει την τιμή που έχει αποθηκευτεί μέσω της πιο πρόσφατης εντολής `STORE` που εκτελέστηκε μέσα από οποιοδήποτε νήμα για την κοινή μεταβλητή). Ο επιθυμητός συγχρονισμός ανάμεσα στα νήματα εφαρμογής μπορεί να επιτευχθεί χρησιμοποιώντας κάποιες καθολικές μεταβλητές ως σηματοφόρους (μέσω `DOWN/UP`).

Η εκτέλεση των νημάτων εφαρμογής γίνεται μέσω του διερμηνέα που αναπτύξατε στο προηγούμενο στάδιο. Επιπλέον, πρέπει να υλοποιηθεί αυτόματη εναλλαγή ανάμεσα στα νήματα εφαρμογής, υπό τον έλεγχο του περιβάλλοντος εκτέλεσης. Αυτό θα πρέπει να υλοποιηθεί σταδιακά, ως εξής.

Σε πρώτη φάση, υλοποιήστε τα νήματα εφαρμογής μέσα από ένα μοναδικό νήμα σε επίπεδο συστήματος. 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 νήματα εφαρμογής μπορεί να γίνεται ανά συγκεκριμένο αριθμό εντολών ή με βάση τον πραγματικό χρόνο ή κάθε φορά που εκτελείται μια από τις εντολές `DOWN`, `UP`, `SLEEP` και `RETURN`.

Σε δεύτερη φάση, υλοποιήστε την ταυτόχρονη εκτέλεση των νημάτων εφαρμογής μέσα από N ξεχωριστά νήματα συστήματος (το N δίνεται κατά την εκκίνηση του περιβάλλοντος εκτέλεσης). Το περιβάλλον εκτέλεσης πρέπει να κατανέμει ομοιόμορφα τα νήματα εφαρμογής στα νήματα συστήματος. Οι εντολές `LOAD/STORE/DOWN/UP` πάνω στις κοινές/καθολικές μεταβλητές πρέπει να λειτουργούν σωστά ακόμα και όταν αυτές καλούνται από νήματα εφαρμογής που εκτελούνται μέσα από ξεχωριστά νήματα συστήματος (που θα πρέπει να συγχρονίζονται σωστά για αυτό το σκοπό).

Δοκιμάστε το περιβάλλον εκτέλεσης γράφοντας προγράμματα που λύνουν κάποια από τα κλασικά προβλήματα συγχρονισμού με σηματοφόρους. 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. Εναλλακτικά, μπορείτε να γράψετε την υλοποίηση σας σε Java ή Python, χρησιμοποιώντας τις βασικές λειτουργίες πολυνηματικής εκτέλεσης και συγχρονισμού της εκάστοτε γλώσσας.