Πανεπιστήμιο Θεσσαλίας
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τμήμα Πληροφορικής

*Μεταγλωττιστές*

## Στοίβα Εκτέλεσης και Εγγραφήματα Δραστηριοποίησης

Σε όλες σχεδόν τις μοντέρνες γλώσσες προγραμματισμού, μια συνάρτηση[1] μπορεί να έχει τοπικές μεταβλητές που δημιουργούνται με την είσοδο στη συνάρτηση. Πολλές κλήσεις της συνάρτησης αυτής μπορούν να είναι ταυτόχρονα ενεργές, και κάθε κλήση δημιουργεί τα δικά της αντίγραφα των τοπικών μεταβλητών.
Για παράδειγμα, στη πιο κάτω συνάρτηση της γλώσσας C:

```
int f (int x) {
   int y = x+x;
   if (y < 10) return f(y) else return y-1;
}
```

ένα νέο αντίγραφο της τοπικής μεταβλητής x (που ταυτόχρονα είναι και τυπική παράμετρος της συνάρτησης) δημιουργείται κάθε φορά που καλείται η συνάρτηση f, το οποίο και αρχικοποιείται από την καλούσα συνάρτηση. Εξ αιτίας των αναδρομικών κλήσεων της f, πολλά αντίγραφα της x υπάρχουν την ίδια στιγμή. Επίσης, ένα νέο αντίγραφο της τοπικής μεταβλητής y δημιουργείται με κάθε είσοδο στο σώμα της συνάρτησης f.
Σε πολλές γλώσσες προγραμματισμού οι τοπικές μεταβλητές καταστρέφονται με την έξοδο από μια συνάρτηση. Επειδή η έξοδος από μια συνάρτηση συνεπάγεται προηγούμενη έξοδο από όλες τις συναρτήσεις που καλούνται από αυτήν, μπορούμε να συμπεράνουμε ότι οι κλήσεις συναρτήσεων έχουν συμπεριφορά LIFO (last-in-first-out). Εάν λοιπόν οι τοπικές μεταβλητές μιας συνάρτησης δημιουργούνται με την είσοδο και καταστρέφονται με την έξοδο μιας συνάρτησης, τότε μπορούμε να χρησιμοποιήσουμε μια δομή *στοίβας* για να τις αποθηκεύουμε στη μνήμη.
Μερικές γλώσσες προγραμματισμού από την άλλη μεριά, δεν καταστρέφουν τις τοπικές μεταβλητές με την έξοδο από μια συνάρτηση. Κάτι τέτοιο συμβαίνει όταν μια γλώσσα υποστηρίζει ταυτόχρονα (α) φωλιασμένες συναρτήσεις και (β) συναρτήσεις που επιστρέφουν συναρτήσεις. Τέτοιες γλώσσες είναι για παράδειγμα οι ML και η Scheme. Για το υπόλοιπο της ανάλυσης δε θα ασχοληθούμε με τέτοιες γλώσσες.

### Η στοίβα εκτέλεσης

Η απλούστερη έννοια της στοίβας είναι ως μία δομή δεδομένων που υποστηρίζει δύο λειτουργίες, τη λειτουργία *push* και τη λειτουργία *pop*. Στην περίπτωσή μας όμως, οι τοπικές μεταβλητές δεν εισάγονται στην, ούτε εξάγονται από τη στοίβα μεμονωμένα, αλλά όλες μαζί. Επίσης, μετά την εισαγωγή των τοπικών μεταβλητών στη στοίβα, θέλουμε να μπορούμε να προσπελάσουμε τοπικές μεταβλητές που βρίσκονται κάτω από την κορυφή της στοίβας, είτε από αυτές που εισάγαμε στην τελευταία, είτε από αυτές που εισάγαμε σε προηγούμενη εισαγωγή. Έτσι, το απλό μοντέλο των δύο λειτουργιών δεν είναι κατάλληλο για το σκοπό μας.
Αντίθετα, θεωρούμε τη στοίβα σαν έναν πίνακα, που προσπελαύνεται με τη βοήθεια ειδικού καταχωρητή, του *δείκτη στοίβας* (stack pointer) που η τιμή του δείχνει σε κάποια θέση στον πίνακα αυτόν. Οι θέσεις πέρα από την τιμή του δείκτη στοίβας περιέχουν τυχαίες άγνωστες

[1] Με τον όρο συνάρτηση εννοούμε κάθε μονάδα ενός προγράμματος που ενεργοποιείται με κάποιο μηχανισμό κλήσης, με πιθανό πέρασμα παραμέτρων, και απενεργοποιείται με κάποιον αντίστοιχο μηχανισμό επανόδου στη μονάδα που την κάλεσε, με πιθανή επιστροφή αποτελέσματος.

τιμές, ενώ οι θέσεις πίσω από την τιμή του δείκτη στοίβας περιέχουν τιμές ενεργών μεταβλητών. Η στοίβα αυτή, που ονομάζεται *στοίβα εκτέλεσης* (run-time stack), συνήθως αυξάνεται μόνο στην κλήση μιας συνάρτησης, κατά μέγεθος τέτοιο, ώστε ο νέος χώρος που δημιουργείται να χωράει όλες τις τοπικές μεταβλητές της συνάρτησης, και μειώνεται ακριβώς πριν την έξοδο από τη συνάρτηση, κατά ίσο μέγεθος. Η περιοχή στη στοίβα που δεσμεύεται για τις τοπικές μεταβλητές μιας συνάρτησης ονομάζεται *εγγράφημα δραστηριοποίησης* (ΕΔ - activation record) της συνάρτησης. Εκτός των τοπικών μεταβλητών (και παραμέτρων) της συνάρτησης, το ΕΔ συνήθως αποθηκεύει βοηθητικές τιμές, όπως τη διεύθυνση επανόδου, την τιμή αποτελέσματος, το σύνδεσμο προσπέλασης, καθώς και προσωρινές μεταβλητές της συνάρτησης. Για ιστορικούς λόγους, οι στοίβες εκτέλεσης ξεκινούν σε υψηλές διευθύνσεις μνήμης και αυξάνονται προς τα κάτω, προς μικρότερες διευθύνσεις μνήμης.
Η σχεδίαση του ΕΔ μιας συνάρτησης λαμβάνει υπ' όψη τα ιδιαίτερα χαρακτηριστικά του συνόλου εντολών της τελικής γλώσσας, αλλά και την αρχική γλώσσα. Όμως, ο κατασκευαστής ενός επεξεργαστή συχνά προδιαγράφει τη μορφή των ΕΔ που πρέπει να υιοθετούνται από τους μεταγλωττιστές όλων των γλωσσών που χρησιμοποιούν τη στοίβα εκτέλεσης στον επεξεργαστή αυτόν. Μερικές φορές αυτή η μορφή δεν είναι και η καλύτερη για κάποιον συγκεκριμένο μεταγλωττιστή ή κάποια συγκεκριμένη γλώσσα, αλλά η χρήση αυτής της μορφής διευκολύνει στη σύνδεση ενός κώδικα με κάποια βιβλιοθήκη, ή με κώδικα που έχει γραφτεί σε άλλη αρχική γλώσσα.
Ανεξάρτητα από τη μορφή του ΕΔ, η δέσμευση των θέσεων σε αυτό γίνεται από το μεταγλωττιστή, ανάλογα με τις ανάγκες μετάφρασης της κάθε συνάρτησης. Συνήθως, πρώτα δεσμεύονται οι θέσεις των τυπικών παραμέτρων της συνάρτησης, της διεύθυνσης επανόδου, της τιμής αποτελέσματος και του συνδέσμου προσπέλασης. Στη συνέχεια δεσμεύονται οι θέσεις των τοπικών μεταβλητών, με τη σειρά που εμφανίζονται στο σώμα της συνάρτησης, ή με όποια σειρά βολεύει το μεταγλωττιστή. Στο τέλος δεσμεύονται θέσεις για προσωρινές μεταβλητές, που συνήθως εξυπηρετούν ανάγκες διάχυσης καταχωρητών, και οι οποίες εμφανίζονται στα τελευταία στάδια μετάφρασης, αφού δηλαδή έχει γίνει η δέσμευση όλων των άλλων θέσεων στο ΕΔ.
Για κάθε θέση που δεσμεύεται από το μεταγλωττιστή, η μετατόπισή της από την αρχή του ΕΔ είναι σταθερή και γνωστή από τη στιγμή της δέσμευσης. Άσχετα αν η θέση του ΕΔ στη μνήμη δε μπορεί να είναι γνωστή στο μεταγλωττιστή, οι εντολές προσπέλασης θέσεων του ΕΔ μπορούν να παραχθούν, με διευθυνσιοδότηση σχετική με το δείκτη στοίβας. Ο δείκτης στοίβας λαμβάνει τιμές κατά την εκτέλεση του παραγόμενου κώδικα, από εντολές που παράγει ο μεταγλωττιστής στην είσοδο και στην έξοδο μιας συνάρτησης. Έτσι, στην είσοδο μιας συνάρτησης, ο δείκτης στοίβας μειώνεται κατά το μέγεθος του ΕΔ της συνάρτησης, ώστε να δείχνει πάντα στην κορυφή της στοίβας, και η στοίβα να μπορεί να δεχτεί το ΕΔ κάποιας άλλης συνάρτησης που καλείται από αυτήν. Αντίθετα, στην έξοδο από τη συνάρτηση, ο δείκτης στοίβας αυξάνεται αντίστοιχα. Η αρχική τιμή του δείκτη στοίβας δίνεται από το σύστημα κατά τη φόρτωση και εκκίνηση ενός προγράμματος.

Καταχωρητές και στοίβα εκτέλεσης

Κάθε μοντέρνος επεξεργαστής έχει ένα μεγάλο αριθμό *καταχωρητών* (συνήθως 32). Για την αύξηση της ταχύτητας των προγραμμάτων, είναι σκόπιμο να διατηρούμε τις τοπικές μεταβλητές, όπως και άλλες ενδιάμεσες τιμές, σε καταχωρητές αντί της στοίβας. Στους περισσότερους σύγχρονους επεξεργαστές, οι αριθμητικές εντολές προσπελαύνουν άμεσα τους καταχωρητές τους και όχι τη μνήμη. Η προσπέλαση της μνήμης απαιτεί ειδικές εντολές *φόρτωσης* και *αποθήκευσης*. Ακόμα και για αρχιτεκτονικές που υποστηρίζουν προσπέλαση της μνήμης μέσα από αριθμητικές εντολές, η προσπέλαση καταχωρητών είναι πιο γρήγορη.
Κάθε επεξεργαστής έχει συνήθως ένα μοναδικό σύνολο καταχωρητών. Από την άλλη μεριά, πολλαπλές μονάδες ενός προγράμματος μπορούν να επιθυμούν ταυτόχρονη χρήση των καταχωρητών αυτών. Έστω για παράδειγμα μια συνάρτηση f που χρησιμοποιεί τον καταχωρητή r για να αποθηκεύσει μια τοπική μεταβλητή, και η οποία καλεί μια άλλη συνάρτηση g, η οποία επίσης χρησιμοποιεί τον ίδιο καταχωρητή για δικούς της υπολογισμούς. Τότε, η τιμή του r πρέπει να αποθηκευτεί στη στοίβα εκτέλεσης, πριν η g τον χρησιμοποιήσει, και να επανα-

φορτωθεί από τη στοίβα, αφού η g ολοκληρώσει τη χρήση του. Το ερώτημα που γεννάται είναι: ποιος κώδικας θα αποθηκεύσει και θα επαναφορτώσει τον r, ο κώδικας της f ή της g; Στην πρώτη περίπτωση λέμε ότι ο r είναι στην ευθύνη της *καλούσας* συνάρτησης, ενώ στη δεύτερη λέμε ότι ο r είναι στην ευθύνη της *καλούμενης* συνάρτησης.
Στις περισσότερες αρχιτεκτονικές η παραπάνω διάκριση δε γίνεται μέσα από το υλικό, αλλά είναι αποτέλεσμα παραδοχής που ορίζεται μέσα από το εγχειρίδιο χρήσης του επεξεργαστή. Στους επεξεργαστές MIPS για παράδειγμα, η παραδοχή είναι ότι οι καταχωρητές 16-23 πρέπει να διατηρούν την τιμή τους μέσα από κλήσεις συναρτήσεων (κι επομένως είναι στην ευθύνη της καλούμενης συνάρτησης), ενώ οι υπόλοιποι μπορούν να αλλάζουν τιμή μέσα από κλήσεις συναρτήσεων (κι επομένως είναι στην ευθύνη της καλούσας συνάρτησης).
Η αποθήκευση και επαναφόρτωση καταχωρητών δεν είναι πάντα απαραίτητη. Εάν στο πιο πάνω παράδειγμα η f – δηλαδή ο μεταγλωττιστής που μεταφράζει την f – γνωρίζει ότι η τιμή κάποιας μεταβλητής x δε θα χρησιμοποιηθεί μετά την κλήση της g, μπορεί να την τοποθετήσει σε έναν καταχωρητή της ευθύνης της f, χωρίς να την αποθηκεύσει πριν την κλήση. Παρόμοια, αν η f έχει μια μεταβλητή i, η οποία χρησιμοποιείται ξανά μετά την κλήση της g, και πιθανά μετά από άλλες κλήσεις άλλων συναρτήσεων, μπορεί να τοποθετήσει την τιμή της σε κάποιον καταχωρητή της ευθύνης της καλούμενης συνάρτησης, οπότε αρκεί να αποθηκεύσει την αρχική τιμή του πριν την πρώτη χρήση του, και να την επαναφορτώσει μετά την τελευταία χρήση του. Η τιμή αυτή είναι η τιμή που ο καταχωρητής έχει από άλλη συνάρτηση που καλεί την f. Η νέα τιμή που αποδίδεται σε αυτόν από την f αποθηκεύεται και επαναφορτώνεται από όποια συνάρτηση καλείται μέσα από την f, αν και εφ' όσον αυτή θέλει να τον χρησιμοποιήσει. Από τα παραπάνω καταλαβαίνουμε ότι η επιλογή του καταχωρητή για κάποια μεταβλητή μπορεί να επηρεάσει το συνολικό αριθμό των εντολών προσπέλασης μνήμης κάποιου προγράμματος. Ο αλγόριθμος δέσμευσης καταχωρητών πρέπει να διακρίνει τις δύο κατηγορίες καταχωρητών, ώστε να κάνει τη βέλτιστη δέσμευση.

Πέρασμα παραμέτρων

Από τη δεκαετία του 1970, οπότε καθιερώθηκε η χρήση της στοίβας εκτέλεσης για την τοποθέτηση των ΕΔ των συναρτήσεων, οι παραδοχές για το μηχανισμό κλήσης συναρτήσεων υπαγορεύουν οι παράμετροι των συναρτήσεων να περνάνε μέσα από τη στοίβα. Κάτι τέτοιο όμως αυξάνει τις προσπελάσεις της μνήμης. Μελέτες αντιπροσωπευτικών προγραμμάτων έχουν δείξει ότι πολύ σπάνια μια συνάρτηση έχει πάνω από τέσσερις παραμέτρους, και σχεδόν ποτέ πάνω από έξι. Έτσι, στους σύγχρονους επεξεργαστές, οι παραδοχές αυτές έχουν τροποποιηθεί, και καθορίζουν ότι οι πρώτες k παράμετροι μιας συνάρτησης (όπου k είναι συνήθως 4 ή 6) περνάνε μέσα από καταχωρητές, και μόνο οι υπόλοιπες μέσα από τη στοίβα.
Βέβαια, δημιουργείται το ερώτημα, τι θα γίνει αν μια συνάρτηση f που δέχτηκε μια παράμετρο μέσω ενός καταχωρητή r καλέσει άλλη συνάρτηση g. Τότε η παράμετρος που δέχτηκε θα πρέπει να αποθηκευτεί στη στοίβα. Έτσι, πώς μειώθηκε ο αριθμός προσπελάσεων στη μνήμη; Στο παραπάνω ερώτημα μπορούν να δοθούν τέσσερις διαφορετικές απαντήσεις, που μπορούν να ισχύσουν ανάλογα με την περίπτωση:

1. Κάποιες συναρτήσεις δεν περιέχουν άλλες κλήσεις συναρτήσεων. Τέτοιες συναρτήσεις, που μάλιστα καλούνται πολύ συχνά – ίσως πιο συχνά από συναρτήσεις που καλούν άλλες συναρτήσεις, δε χρειάζεται να αποθηκεύσουν καμία παράμετρο στη στοίβα. Ακόμα περισσότερο, αν οι τοπικές μεταβλητές μιας τέτοιας συνάρτησης αντιστοιχούνται όλες σε καταχωρητές, τότε αυτή δε δημιουργεί καν ΕΔ στη στοίβα εκτέλεσης, κάτι που οδηγεί σε σημαντική εξοικονόμηση εντολών και προσπελάσεων μνήμης.
2. Κάποιοι μεταγλωττιστές δεσμεύουν καταχωρητές λαμβάνοντας υπ' όψη τις κλήσεις συναρτήσεων, ώστε να δεσμεύουν διαφορετικούς καταχωρητές σε συναρτήσεις που καλούνται μέσα από άλλες. Τότε, η συνάρτηση f μπορεί να αποφύγει τη χρήση της στοίβας για την αποθήκευση των παραμέτρων της πριν την κλήση της g.
3. Πολλές φορές, όταν έρθει η στιγμή της κλήσης, η συνάρτηση f έχει τελειώσει με τη χρήση της παραμέτρου που πέρασε μέσα από τον καταχωρητή r. Έτσι δε θα χρειαστεί να την

αποθηκεύσει στη στοίβα, και οποιαδήποτε επόμενη χρήση του r δε θα σχετίζεται με την παράμετρο αυτή.

4. Μερικές αρχιτεκτονικές υποστηρίζουν *παράθυρα καταχωρητών*, που αλλάζουν με κάθε κλήση συνάρτησης, ώστε να μην απαιτείται χρήση της στοίβας για αποθήκευση των παραμέτρων με κάθε κλήση.

Πέρα από την ανάγκη αποθήκευσης της τιμής του r εξ αιτίας της κλήσης της συνάρτησης g, η f αποθηκεύει τον r στη στοίβα εκτέλεσης και όταν το πρόγραμμα χρησιμοποιεί τη διεύθυνση της παραμέτρου. Οι καταχωρητές δεν έχουν διεύθυνση μνήμης, κι επομένως, αν ο κώδικας της f περιέχει κάποιο δείκτη (pointer) στην παράμετρο που πέρασε μέσα από τον καταχωρητή r, η f θα πρέπει να αποθηκεύσει την τιμή της παραμέτρου στη στοίβα, απ' όπου θα μπορεί να διαβαστεί με αποδεικτοδότηση του αντίστοιχου δείκτη. Για εύκολη επίλυση του παραπάνω προβλήματος, μπορούν όλες οι παράμετροι να δεσμεύουν διαδοχικές θέσεις στο ΕΔ της συνάρτησης, άσχετα αν οι θέσεις αυτές τελικά χρησιμοποιούνται από τον κώδικα ή όχι. Κάτι τέτοιο κάνει η γλώσσα C, η οποία εκτός από δείκτες, υποστηρίζει και το μηχανισμό μεταβλητού αριθμού παραμέτρων, ο οποίος για να λειτουργήσει σωστά, προϋποθέτει ότι διαδοχικές παράμετροι βρίσκονται σε διαδοχικές θέσεις μνήμης.
Πάντως, ένας καλύτερος τρόπος χρήσης της διεύθυνσης μιας παραμέτρου είναι με το μηχανισμό περάσματος *κατ' αναφορά*. Με το μηχανισμό αυτό, το πρόγραμμα δεν έχει τη δυνατότητα άμεσης προσπέλασης της διεύθυνσης της παραμέτρου, κάτι που γίνεται στη C με τη βοήθεια δεικτών, αλλά μόνο έμμεσης προσπέλασης στη διεύθυνση αυτή μέσα από αναθέσεις στην παράμετρο. Το πέρασμα κατ' αναφορά υλοποιείται απλά με πέρασμα της διεύθυνσης της πραγματικής παραμέτρου – που προφανώς πρέπει να είναι τιμή αριστερής προσπέλασης. Κάθε αναφορά στην παράμετρο ως τιμή δεξιάς προσπέλασης απαιτεί μια παραπάνω προσπέλαση μνήμης για την αποδεικτοδότησή της, ενώ κάθε ανάθεση στην παράμετρο γίνεται με αποθήκευση στη διεύθυνση που πέρασε κατά την κλήση. Με το μηχανισμό αυτό εξασφαλίζεται ότι μετά την επιστροφή από τη συνάρτηση, δε θα παραμένει καμία *ξεκρέμαστη αναφορά* σε παράμετρό της.

Διεύθυνση επανόδου

Όταν μια συνάρτηση f ολοκληρώνει την εκτέλεσή της, πρέπει να επιστρέψει στη συνάρτηση που την κάλεσε. Έτσι, πρέπει να γνωρίζει σε ποια διεύθυνση κώδικα θα μεταφέρει τον έλεγχο ροής. Αν η διεύθυνση από την οποία έγινε η κλήση της f ήταν η διεύθυνση A, τότε η ζητούμενη διεύθυνση είναι συνήθως η διεύθυνση A+1, και λέγεται *διεύθυνση επανόδου*.
Παλαιότερα, οι διευθύνσεις επανόδου αποθηκεύονταν στη στοίβα εκτέλεσης κατά τις κλήσεις συναρτήσεων. Σήμερα είναι πιο αποδοτικό κάτι τέτοιο να γίνεται μέσα από καταχωρητές. Εκτός από τη μείωση του αριθμού προσπελάσεων μνήμης, αυτό εξυπηρετεί και το υλικό, μια που ο μηχανισμός *σύνδεσης*, δηλαδή της αποθήκευσης της διεύθυνσης επανόδου, υλοποιείται ευκολότερα σε κάποιον καταχωρητή, παρά σε κάποια διεύθυνση μνήμης.
Έτσι, οι σύγχρονοι επεξεργαστές, με κάθε εντολή κλήσης συνάρτησης, αποθηκεύουν τη διεύθυνση επανόδου σε κάποιον καταχωρητή. Από εκεί και πέρα, ισχύουν όσα αναφέρθηκαν πιο πάνω, για οποιαδήποτε ανάγκη αποθήκευσης της τιμής του καταχωρητή στη στοίβα. Ειδικότερα, αν ο καταχωρητής είναι προκαθορισμένος, όπως για παράδειγμα συμβαίνει με την εντολή jal της αρχιτεκτονικής MIPS, τότε κάθε κλήση συνάρτησης μέσα από την f απαιτεί την αποθήκευση της τιμής του καταχωρητή πριν την κλήση, μια που διαφορετικά η τιμή του θα χαθεί. Μετά την επιστροφή στην f, η διεύθυνση επανόδου πρέπει να επαναφορτωθεί από τη στοίβα, ώστε να χρησιμοποιηθεί ως διεύθυνση προορισμού με την ολοκλήρωση της f για την επιστροφή στη συνάρτηση που την έχει καλέσει.

Μεταβλητές του ΕΔ

Σύμφωνα με όσα έχουμε πει πιο πάνω, οι μοντέρνες παραδοχές που χρησιμοποιούνται σε μια κλήση συνάρτησης απαιτούν οι παράμετροι να περνάνε μέσω καταχωρητών, η διεύθυνση επανόδου να αποθηκεύεται σε καταχωρητή, αλλά και η τιμή αποτελέσματος να αποθηκεύεται

σε καταχωρητή. Επίσης, πολλές από τις τοπικές μεταβλητές αντιστοιχούνται σε καταχωρητές. Τελικά, έχουμε πραγματική χρήση του ΕΔ στη στοίβα εκτέλεσης, μόνο αν είναι απαραίτητο, και αυτό θα συμβαίνει σε μια από τις ακόλουθες περιπτώσεις:

- Η τοπική μεταβλητή περνάει σε άλλη συνάρτηση κατ' αναφορά, οπότε πρέπει να έχει διεύθυνση μνήμης.
- Η διεύθυνση μιας τοπικής μεταβλητής χρησιμοποιείται άμεσα από τον κώδικα, όπως για παράδειγμα με τον τελεστή & της γλώσσας C.
- Η τοπική μεταβλητή προσπελαύνεται από κάποια φωλιασμένη συνάρτηση, και ο μεταγλωττιστής δε λαμβάνει υπ' όψη τις κλήσεις συναρτήσεων στη δέσμευση καταχωρητών.
- Η τιμή της μεταβλητής δε χωράει σε καταχωρητή, και ο μεταγλωττιστής δε χρησιμοποιεί πολλαπλούς καταχωρητές για αποθήκευση μιας τέτοιας τιμής.
- Η μεταβλητή δεν είναι βαθμωτή, οπότε είναι απαραίτητο να γίνουν υπολογισμοί για την εύρεση της διεύθυνσης κάποιου στοιχείου ή πεδίου αυτής, με βάση την αρχική διεύθυνση της μεταβλητής.
- Ο καταχωρητής που περιέχει την τιμή της μεταβλητής θα χρειαστεί για κάποια άλλη λειτουργία, όπως για παράδειγμα το πέρασμα κάποιας παραμέτρου, οπότε η τιμή αυτή πρέπει είτε να μεταφερθεί σε άλλον καταχωρητή, είτε να αποθηκευτεί στη στοίβα.
- Η παρούσα συνάρτηση έχει τόσες πολλές τοπικές μεταβλητές που είναι ταυτόχρονα ενεργές και δε χωράνε σε καταχωρητές. Τότε, ο αλγόριθμος δέσμευσης καταχωρητών δημιουργεί προσωρινές μεταβλητές που τις διαχέει στη στοίβα.

Ας σημειωθεί ότι οι πιο πολλές από τις παραπάνω περιπτώσεις παρουσιάζονται μετά από τη δήλωση της αντίστοιχης τοπικής μεταβλητής. Έτσι, είναι αδύνατο να ξέρουμε τη στιγμή που συναντάμε μια δήλωση, αν η μεταβλητή που δηλώνεται θα χρησιμοποιήσει τη στοίβα εκτέλεσης ή όχι. Επομένως, είναι βολικό να δεσμεύουμε χώρο στο ΕΔ με κάθε δήλωση μεταβλητής, άσχετα αν αργότερα ο αλγόριθμος δέσμευσης καταχωρητών την αντιστοιχήσει σε κάποιον καταχωρητή και η θέση που δεσμεύτηκε δε χρησιμοποιηθεί.

Σύνδεσμοι προσπέλασης

Σε γλώσσες που επιτρέπουν φωλιασμένες δηλώσεις συναρτήσεων, είναι σύνηθες εσωτερικά δηλωμένες συναρτήσεις να αναφέρονται σε μη τοπικές μεταβλητές που είναι δηλωμένες σε εξωτερικές συναρτήσεις.
Για παράδειγμα, στον κώδικα Pascal:

```
type tree = record
    key: string; left: ^tree; right: ^tree;
    end;
procedure prettyprint(tree : tree);
    var output: string;
    procedure write(s : string);
        begin
        output := concat(output,s);
        end;
    procedure show(n : integer; t : tree);
        procedure indent(s : string);
            var i: integer;
            begin
            for i := 1 to n do write("\t");
            output := concat(output,s);
            write("\n");
            end;
        begin
        if t = nil then indent(".");
        else
            begin
            indent(t.key);
            show(n+1,t.left);
```

```
            show(n+1,t.right);
            end;
        end;
    begin
    output := "";
    show(0,tree);
    print(output);
    end;
```

η διαδικασία write αναφέρεται στη μη τοπική μεταβλητή, ενώ η διαδικασία indent αναφέρεται στις μη τοπικές μεταβλητές n και output. Για να επιτευχθεί τέτοια αναφορά, πρέπει οι φωλιασμένες διαδικασίες να έχουν προσπέλαση όχι μόνο στο δικό τους ΕΔ, αλλά και στα ΕΔ άλλων διαδικασιών.

Μία μέθοδος που χρησιμοποιείται για την επίλυση αναφοράς μη τοπικών μεταβλητών είναι βασισμένη στους *συνδέσμους προσπέλασης*. Αν η γλώσσα προγραμματισμού υποστηρίζει στατικό δέσιμο στις κλήσεις συναρτήσεων, όπως συμβαίνει στην Pascal, ο σύνδεσμος προσπέλασης για μια συνάρτηση f είναι η διεύθυνση στη στοίβα εκτέλεσης στην οποία αρχίζει το ΕΔ της συνάρτησης που περικλείει την f. Εφ' όσον η f είναι φωλιασμένη στη συνάρτηση αυτή, δεν είναι ορατή από άλλες συναρτήσεις, κι επομένως αν η f είναι ενεργοποιημένη, τότε θα είναι ενεργοποιημένη και η συνάρτηση που την περικλείει. Για την επίλυση αναφοράς από την f μιας μη τοπικής μεταβλητής που είναι δηλωμένη εξωτερικά της περικλείουσας συνάρτησης, θα πρέπει η κώδικας να ακολουθήσει επαναληπτικά τους συνδέσμους προσπέλασης, μέχρι να φτάσει στο επίπεδο φωλιάσματος όπου δηλώνεται η μεταβλητή. Ο τελευταίος σύνδεσμος προσπέλασης που διαβάζεται δίνει τη διεύθυνση στην οποία αρχίζει το ΕΔ στο οποίο είναι αποθηκευμένη η μεταβλητή.

Έτσι, στον πιο πάνω κώδικα θα έχουμε για παράδειγμα:

1. Η διαδικασία prettyprint καλεί τη διαδικασία show, οπότε αποθηκεύει την αρχική διεύθυνση του δικού της ΕΔ στη θέση του συνδέσμου προσπέλασης της δεύτερης.
2. Η διαδικασία show καλεί τη διαδικασία indent, οπότε αποθηκεύει την αρχική διεύθυνση του δικού της ΕΔ στη θέση του συνδέσμου προσπέλασης της indent.
3. Η διαδικασία show καλεί τον εαυτό της αναδρομικά, οπότε αντιγράφει το δικό της σύνδεσμο προσπέλασης στη θέση του συνδέσμου προσπέλασης στο ΕΔ της αναδρομικής κλήσης.
4. Η διαδικασία indent αναφέρεται στη μη τοπική μεταβλητή n, η οποία είναι δηλωμένη στη διαδικασία show. Η αναφορά γίνεται σε διεύθυνση σχετική με την τιμή του δικού της συνδέσμου προσπέλασης, ο οποίος δείχνει στο ΕΔ της διαδικασίας show. Η μετατόπιση στην προσπέλαση είναι γνωστή στο μεταγλωττιστή από τη μετάφραση της τελευταίας.
5. Η διαδικασία indent καλεί τη διαδικασία write. Εφ' όσον η write περιέχεται στην prettyprint και όχι στην indent, ο σύνδεσμος προσπέλασής της πρέπει να δείχνει στο ΕΔ της prettyprint. Ο κώδικας θα πρέπει να ακολουθήσει το σύνδεσμο προσπέλασης της indent στο ΕΔ της show, ώστε να αντιγράψει στη θέση του συνδέσμου προσπέλασης της write τον σύνδεσμο προσπέλασης της show, που είναι στο ίδιο βάθος φωλιάσματος με την write. Και η show και η write περικλείονται από την ίδια διαδικασία prettyprint, αλλιώς η write δε θα ήταν ορατή στην indent.
6. Η διαδικασία indent αναφέρεται στη μη τοπική μεταβλητή output, η οποία είναι δηλωμένη στη διαδικασία prettyprint. Για να βρει τη μεταβλητή αυτή, ακολουθεί το σύνδεσμο προσπέλασής της στο ΕΔ της show, απ' όπου ακολουθεί το σύνδεσμο προσπέλασης της show στο ΕΔ της prettyprint. Η αναφορά τώρα μπορεί να γίνει σε διεύθυνση σχετική με την τιμή του τελευταίου συνδέσμου προσπέλασης που διάβασε.

Από τα παραπάνω συμπεραίνουμε ότι σε γλώσσες που υποστηρίζουν φωλιασμένες συναρτήσεις με στατικό δέσιμο, σε κάθε κλήση συνάρτησης και σε κάθε αναφορά σε μη τοπική μεταβλητή ο κώδικας πρέπει να υλοποιεί μια αλυσίδα από μηδέν ή περισσότερες αναγνώσεις από τη στοίβα εκτέλεσης διαδοχικών συνδέσμων προσπέλασης. Το μήκος της αλυσίδας είναι η διαφορά στο βάθος φωλιάσματος των συναρτήσεων που εμπλέκονται στην κλήση ή στην αναφορά στη μεταβλητή.

Αν από την άλλη μεριά η γλώσσα προγραμματισμού υποστηρίζει δυναμικό δέσιμο στις κλήσεις συναρτήσεων, η επίλυση αναφοράς σε μη τοπικές μεταβλητές είναι παρόμοια με την παραπάνω, μόνο αν για κάθε τέτοια αναφορά υπάρχει μόνο μία πιθανή αλυσίδα αναγνώσεων που να οδηγούν στο ΕΔ όπου επιλύεται η αναφορά. Διαφορετικά, και επειδή το δέσιμο γίνεται στο χρόνο εκτέλεσης, ο μεταγλωττιστής δε μπορεί να ξέρει ποιο δρόμο θα ακολουθήσει στις διαδοχικές αναγνώσεις, κι επομένως απαιτείται άλλη, περισσότερη πολύπλοκη μέθοδος για την επίλυση της αναφοράς, η οποία επιδρά σε ολόκληρο το μηχανισμό κλήσης συναρτήσεων. Στην απλή περίπτωση μίας πιθανής αλυσίδας αναγνώσεων, μπορούμε να χρησιμοποιήσουμε τη μέθοδο του συνδέσμου προσπέλασης, με τη διαφορά ότι η ενημέρωση του συνδέσμου σε κάθε κλήση δε γίνεται με βάση το βάθος φωλιάσματος, αλλά με βάση το βάθος κλήσης. Από τη στιγμή που η αλυσίδα που πρέπει να ακολουθηθεί είναι μοναδική, ο μεταγλωττιστής ξέρει πόσες εντολές ανάγνωσης πρέπει να παράγει, ώστε ο κώδικας να φτάσει το ζητούμενο ΕΔ.